Foundations of Ad Hoc Wireless Networks

Andrea Cerone

July 29, 2012

Declaration

This thesis has not been submitted as an exercise for a degree at this or any other university. It is entirely the candidate's own work. The candidate agrees that the Library may lend or copy the thesis upon request. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

_____

Andrea Cerone

Summary

In this thesis we implement different process calculi to model *ad hoc wireless networks*. Each of these calculi considers different features of wireless systems, which are selected by focusing on the kind of applications or protocols that need to be analysed.

In particular, we decided to take into account the following features:

local broadcast a wireless network is represented as a collection of agents, called *nodes* (or *stations*); each node has the capability to broadcast messages and to wait for messages to be received. A graph is used to describe the topology of the network; a message broadcast by a node can be detected by all, and only all, its neighbours. All the neighbours because I wish to model ad hoc wireless networks, which employ broadcast communication. Only all, because every node has a range of transmission; the neighbours of a node, in the graph representing the network topology, are intuitively exactly those nodes which lie in its range of transmission, and have therefore the chance to receive the message. In this scenario, I make the assumption that communication is perfect; every time a message is broadcast by a node, all the nodes in its range of transmission can detect it correctly. As I shall note later, this amounts to assume that perfect communication is achieved by means of some protocol at the *MAC sub-layer* of the *ISO/OSI reference model* [69].

probabilistic behaviour the calculus described above is extended so that nodes exhibit random behaviour. There are several motivations for this extension. First, in the last decades there has been a growing research interest in the use of probabilistic protocols in wireless networks, which have been applied in different situations; see for example [14] or [59]. Second, ad hoc wireless networks are multi-agent systems. For such systems, there exist problems of high relevance that cannot be solved by deterministic protocols, but for which unbounded time probabilistic protocols have been exhibited and proved correct. One of the most famous problems in this family is distributed consensus [6]

Time and collisions the calculi discussed above are suitable for the analysis of high level networks, specifically for the description of protocols and applications which lie at the *internet layer*, or at a higher one, in the *ISO/OSI* reference model. However, since the birth of wireless systems, protocols at lower layers, such as the MAC sub-layer, have been defined and analysed in detail; see [40] for a survey. To accomodate the needs of the research community, I decided to develop a process calculus in which the requirement that communication is perfect is dropped. In this calculus, broadcast of messages is modeled as a timed activity; they have a start phase and a end phase, during which time can pass. This constraint allows the modelling of situations in which a broadcast is performed while another one is in progress, causing a collision; that is, in this scenario a receiver detects a corrupted (noisy) message. I stress here that in this calculus the communication topology is flat; a broadcast can be detected by every station in the network. This is because the literature about collisions in wireless networks is relatively poor; in our own opinion, it is best to analyse an easier framework in order to understand in depth the semantics of broadcast processes subject to collision, rather than trying to accomplish the same task over frameworks which are more complicated both to define and analyse.

For each of these calculi, the problem of defining a behavioural compositional theory is tackled. Compositional reasoning is very useful; in few words, it allows the analysis of the behaviour of complex (i.e. very

large) networks by looking at the behaviour of two simpler networks, selected in a way that their composition coincides with the former one.

However, it is important here to remark that the notion of *composition of networks* has to be stated precisely. In process calculi, such as CCS and CSP, composing two processes is a simple problem, as it is sufficient to define a set of structural operational semantics rules that state how two processes P and Q interact in their parallel composition $P \mid Q$.

When dealing with networks the situation becomes more complicated. This is because networks are equipped with a topological structure by definition; processes are associated to nodes (or *locations*), and a notion of neighborhood between nodes is provided by a *digraph*. When composing two networks, we want at the same time to ensure that the topological structure of them is both *preserved* and *non-corrupted*. Preservation of the topological structure corresponds to requiring that two neighbouring nodes in a network are still neighbours when the latter is composed with another network; also, the process associated with a location has to be preserved. Non-corruption (or *integrity*) corresponds to requiring that no additional neighbours are introduced for any node in a network when it is composed with another one.

In order to ensure these two properties, network composition has to be defined as a partial operator. For example, it is not possible to compose two networks which have different processes associated with the same location, as code preservation would not be ensured.

While it is possible to define an operator that ensures preservation and integrity of networks, this approach has a severe limitation. Behavioural compositional preorders induced by this operator (such as the testing preorders [17]) turn out to be degenerate. That is, networks cannot be distinguished from each other.

An alternative approach is that of weakening the constraints required by composition operator; given two networks $\mathcal{M}$ and $\mathcal{N}$, we allow the structure of the latter to be corrupted in their composition $\mathcal{M} \natural \mathcal{N}$. The main drawback of this weakening is that the composition operator is not symmetric anymore, so that compositional reasoning becomes more complex; however, the behavioural preorders it induces do not lead to a degenerate theory, and relating networks with different topological structure is possible.

The study of behavioural preorders for networks constitutes the main contribution of this thesis. Testing behavioural preorders in the style of Hennessy and de Nicola are defined and analysed for each of the calculi above, and proof techniques for them are exhibited. This allows to demonstrate whether two networks cannot be distinguished by any test. Also, it is possible to check if a network satisfies a specification. This is done by exhibiting a simple network, for which it is straightforward to prove that it satisfies a desired (extensional) property; then, for any other network, it is sufficient to show that it is equivalent to the former one in order to ensure that it also satisfies the required property.

We provide a wide range of applications that show how the proof techniques can be used to analyse practical situations and problems of interest. In this case, we decided to follow two different strands:

Wireless Networks problems which are particular to (wireless) systems are taken into account; specifically we take into account the analysis of as routing protocols and connection protocols.

Distributed Systems in this case both theoretical and practical problems in distributed systems are analysed; examples include the implementation of a virtual shared memory and the problem of consensus.

Acknowledgements

First of all, I want to thank my supervisor, *prof. Matthew Hennessy*, who has supported my research throughout these three years and helped me understood and overcome the challenges that arise when making research in the area of formal methods. Also, I have to say that I am very happy to have him as a supervisor, since he was very patient and he explained carefully all the topics on which I needed help.

I want also to thank *Prof. Massimo Merro*, who helped me understand the semantics of broadcast and wireless systems via a lot of phone calls on skype.

Thanks also to *Giovanni Bernardi* and *Carlo Spaccasassi*, for the interesting discussions we had on the topic and for helping me solving some problems on the whiteboard of our lab. And to to *Riccardo Bresciani* for the help he gave me on Latex.

Finally, thanks to the *Science Foundation Ireland*, which financially supported both my research and my education.

That was all for what it concerns acknowledgments related to the world of research. However, I want also to thank other people that helped me survive peacefully during these three years.

Obviously, I want to thank my parents *Giorgio Cerone* and *Susanna Scocchera* for the moral support they gave me when I mostly needed it (especially in my first year abroad). Also thanks to *Irene Scocchera* and *Francesco Tagliaferri* for their support.

Thanks also to my brother, *Luca Cerone*, who was living with me here in Dublin during this time without being too much demanding; and thanks to him for taking care of the house more than I did. Another thank goes to his girlfriend, *Florence de Filippis*, for making me laugh most of the time when I was at home.

I want to thank the *players of the Fitzwilliam Card Club*, for during my first year they contributed to financially support my survival here in Dublin (even if I am not sure that they wanted to).

Quoting *Paul Erdos*, who said that *A mathematician is a device for turning coffee into theorems* (though I have to say that I am not a mathematician), I am sure that I would have not been able to complete this Thesis (especially in this last month) without a massive amount of caffeine. Thanks then to the staff of the *Science Gallery Cafe* for preparing me coffee (and sometimes offering it) every day. Hoping I do not forget anyone (a lot of people has worked there during these three years). *Costantino*, *Flavia*, *Angela*, *Lisa della Gassa*, *Leandro*, *Valeria Colnaghi*, *Vanesa Marioni*, *Alessandro*, *Arianna*, *Timeh*, *Agatha*, *Nicky*, *Marisol*, *Norita*, *Lucy*, *Izabela*, *Francesco Formica*.

It would have been difficult to survive in Dublin without having friends; therefore, thanks to *Maria Clara di Biase*, *Arisa Simazaki*, *Prysca de Yp*, *Livia Rossi* and *Dario* (dude, what's your surname?), *Eliseo del Forno* and way too many other people, for their friendship during my third year in Dublin and all the nights out. I had fun with you, too bad all of you left Ireland.

I would like to thank also *Alessandro Vaccaro* and *Cristina de Persis* for their friendship during my third year here, and for all the coffee breaks.

Thanks to the people of the town of *Carrito* (and acquitances) in Italy for all the fun we had during the Summer and whenever I was in Italy (a very special thank among these people goes to *Maurizio Rossi*, *Patrizia Eramo*, *Natalie Marra*, *Veronica* and *Ilaria Almonte*, *Danilo* and *Alberto di Nicola*, *Laura Cerone*). Also thanks to the people of *Piazza Fiume* for their friendship when I was back in Rome (guys, you are too many, and there is no one among you which I want to thank more than the others; so no names here :P).

There are two pubs to which I become affectionated during all this time; the first one is *Dicey's Garden*, here in Dublin, while the other one is the *Stone Age* in Rome. To all the people working, who have worked, or which I knew there, thanks a lot.

Other people I want to thank are *Victoria Cristina Vize*, for all the conversations we had on MSN (I'm still sorry); thanks also to *Alessandro Formosa* for suggesting me I had to watch Game of Thrones and for being one of the best people I have known. I also want to thank *Samantha Santamaria Rodriguez* for showing me the city of Barcelona last year. Thanks also to *Matteo Cimini* for coming to visit me in Dublin during St. Patrick's day; when I have time I'll go to visit you in Iceland. Thanks to *Giulio Gazzola* too, who helped me developing a less anti-social personality; to him I wish all the best luck for his marriage this year.

Last of All, I want to thank *Alessandra Petrecca* for too many reasons; the first one is that of designing the logo of my Thesis, which in the end I decided not to include. But mostly, thanks for taking care of me, being supportive, bringing me Tequila chocolates (can I have others?), the patience she had while I was writing the Thesis (other people would have gotten really angry at me), and most of all thanks for her love.

There are a lot of other people that I would like to thank, but I'm afraid that if I started I would end up having the acknowledgments being more than a half of the whole thesis; I am very sorry guys.

Addio, addio e un bicchiere levato
al cielo d'Irlanda e alle nuvole gonfie.
Un nodo alla gola ed un ultimo sguardo
alla vecchia Anna Liffey e alle strade del porto.
Un sorso di birra per le verdi brughiere
e un altro ai mocciosi coperti di fango,
e un brindisi anche agli gnomi a alle fate,
ai folletti che corrono sulle tue strade.

Modena City Ramblers,
In un giorno di Pioggia

Contents

Chapter 1

Introduction

In this thesis we develop a formal approach to the analysis and verification of ad-hoc wireless networks and distributed systems. We propose different process calculi to describe in a rigorous way the behaviour of wireless systems. These calculi differ both in the computational power we grant to wireless stations and in the level of abstraction used to model wireless communication between stations.

For each of these calculi we propose different notions of behavioural equivalences, based on the idea that two networks are deemed to be equivalent whenever they cannot be distinguished by any agent which is able to observe their behaviour and to interact with them. Here much depends both on the notion of observation performed by an agent over a wireless system as well as on the power that we grant to agents for interacting with a network. These two parameters lead to a wide variety of behavioural equivalences; most of them are described in [67, 68].

For each notion of behavioural equivalence we propose we exhibit illuminating examples of networks which are equivalent, aiming to convince the reader that such behavioural equivalences capture the intuitive notion of two wireless networks *having the same behaviour*.

We also develop proof methodologies to check with mathematical rigour if two networks are deemed equivalent with respect to the proposed notions of behavioural equivalence. Such proof techniques can be used to analyse rather complicated situations, such as proving that a network protocol behaves correctly with respect to a specification.

The rest of this Chapter is organised as follows: in Section 1.1 we first describe, at a very high level, how wireless systems work; then we focus on how wireless networks can interconnect by giving a brief overview of the *TCP/IP reference model*. This can be thought of as a stack of layers organised in a hierarchical way. Each of these layers contain protocols to accomplish a dedicated task; here the main idea is that a protocol contained in a given layer provides services to protocols at the upper level layer, and requests services from protocols at the lower level layer. Most of the time we refer to [65] to illustrate the role of each layer in the *TCP/IP protocol stack*.

In Section 1.2 we emphasise the need for formal methods in the analysis of wireless systems. As we will see, wireless networks are rather complicated systems, mainly due to broadcast communication and the fact that wireless stations are spatially distributed, giving rise to the concept of a network topology. The development of mathematical tools for the analysis of wireless networks greatly helps in the verification of the behaviour of network protocols.

Section 1.3 contains a literature review of related work. We describe what research has already been carried out for the formal analysis of wireless and distributed systems, and we emphasise the most important results that have been achieved in the last decade.

In Section 1.4 we describe in more details the contents of this thesis. We give a brief overview of the contents of each chapter of the thesis, specifying both its topic and the results it contains.

1.1 Wireless Networks and the TCP/IP standard

Since the first experiments on wireless systems, in the *18th century*, lots of progress have been made. Nowadays wireless stations are able to interconnect with each other in a network, providing services to users such as connections to the internet and mobile telephony.

At the very basic level, the operations that can be performed by a wireless station can be summarised as *broadcasting a message*, in the form of an electromagnetic wave, and *detecting a message* and using an antenna to convert it to electric current via the use of an antenna.

Messages, or electromagnetic waves, broadcast by a network propagate through a unique medium of communication, the *ether*. Since the signal broadcast by a network is an electromagnetic wave, it has a frequency, an amplitude and a power; the last determines the range of transmission of the signal [1].

Once a message has been broadcast by a wireless station with a given power, it can be detected by other stations in the sender's transmission range; communication is broadcast, in the sense that a message sent from a station can be detected by any other station in the transmission range of the former.

The broadcast nature of wireless communication gives rise to technical problems; for example, it is possible for a wireless station to be exposed to signals coming from different stations at the same time. Since these signals are electromagnetic waves, the receiving station will detect the sum (or superposition) of the two waves originated by each of the broadcasting stations. Unless some multiple access mechanism is used, it would not be possible for the station to reconstruct the two different signals which have been broadcast. In this case we say that a collision has happened.

Another feature of wireless network is that communication is *half-duplex*. At any given time, a wireless station can either transmit or receive some message, but it cannot receive and transmit messages at the same instant of time. However, recent research has shown that it is possible to obtain *full-duplex* communication in wireless stations [12].

In order to allow wireless stations to interconnect with each other in a network, providing remote services to users, a series of protocols which enable them to accomplish more and more complicated tasks have to be defined; these include how to represent a bit as an electromagnetic wave, how to avoid stations being exposed to different transmissions at any given instant of time, and how to establish a connection between two stations which are not in each other's range of transmission.

We describe the protocol suite known as the *TCP/IP standard*, or *TCP/IP reference model*. It consists of a description of a series of protocols linked to each other in a hierarchical fashion, and it is the basis of the *internet*; a protocol implemented in the n-th layer in this hierarchical structure provides services for protocols implemented at the $n+1$-th layer, often exploiting the services made available by protocols implemented at the $n-1$-th layer. The protocol stack of the *TCP/IP standard* is depicted in Figure 1.1.

Starting from the first level, the *physical layer*, in which the only task that a wireless station can perform is that of sending and receiving electromagnetic waves, the protocols at each layer solve more and more difficult problems. At the end of the protocol hierarchy there is the *application layer*, which provides the primitives to the end-user to access remote services, such as checking its own e-mail account or browsing the world-wide web.

We remark that the *TCP/IP* protocol stack does not contain the description of services to be implemented in wireless stations; rather, they contain the guidelines, expressed in terms of primitives that protocols at a given layer should provide to protocols at the upper layer (or to the end user, in the case of the *application layer*), that should be followed by any group of (either wired or wireless) stations to be able to interconnect each other in a network. The description of how services offered by the lowest layers of the *TCP/IP reference model* in a wireless network are described in either the *IEEE 802.11* standard or one of its more modern extensions.

We also point out that the *TCP/IP* standard is not the only practical possibility which is available to establish a connection between two or more wireless stations. For example, in the *GSM standard* protocols to

[1] Other factors, such as the diffraction index of the ambiental environment, also play a significant role in determining the range of transmission of a signal

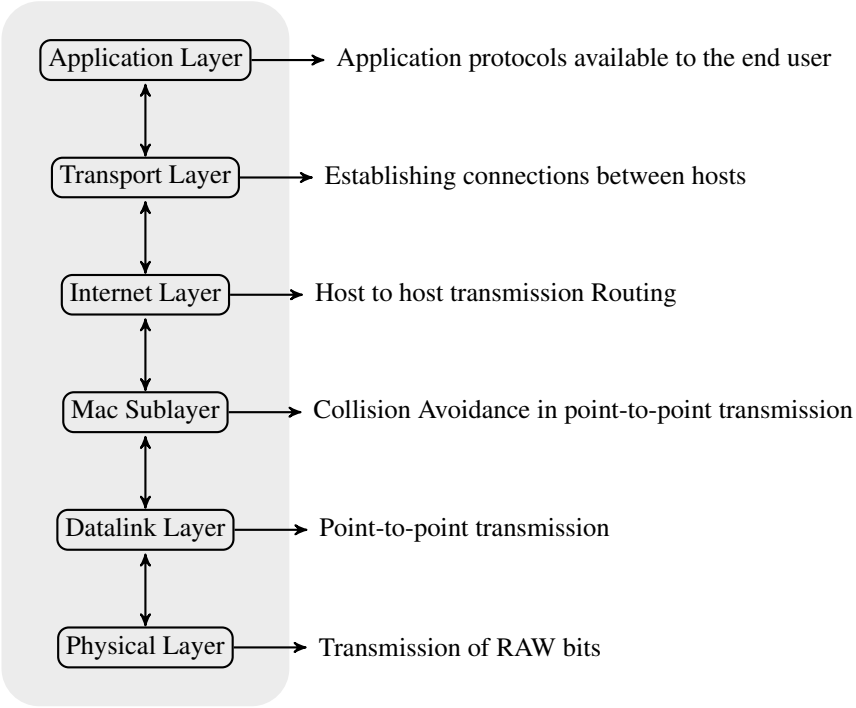


Figure 1.1: The TCP/IP Protocol Stack

be used by mobile phones to connect to each other to communicate with an *Access Point* are defined, as well as protocols to be used by access points to establish a connection between two different mobile phones or to provide multimedia services to a mobile phone. See [31] for details.

In the rest of this Section we briefly describe the services provided by protocols implemented at a given level of the *TCP/IP* standard. This description is taken from [65].

The Physical Layer *The physical layer is concerned with transmitting raw bits over a communication channel. The design issues have to do with making sure that when one side sends a* 1 *bit, it is received by the other side as a* 1 *bit, not as a* 0 *bit. [..] The design issues here largely deal with mechanical, electrical, and timing interfaces, and the physical transmission medium, which lies below the physical layer.* We remark that, in the case of wireless stations, the physical transmission medium is the *ether*.

The Datalink Layer *The main task of the data link layer is to transform a raw transmission facility into a line that appears free of undetected transmission errors to the network layer. It accomplishes this task by having the sender break up the input data into data frames [..] and transmit the frames sequentially. If the service is reliable, the receiver confirms correct receipt of each frame by sending back an acknowledgement frame. [..] Broadcast networks have an additional issue in the data link layer: how to control access to the shared channel. A special sublayer of the data link layer, the medium access control sublayer, deals with this problem.*

We recall here that wireless stations are indeed broadcast stations, that is the transmission of a wireless station can be detected by all the stations in the sender's range of transmission. The most important task that the *Medium access control sublayer* (or MAC sublayer) has to accomplish is that of avoiding a collision to happen; that is, a wireless station should not be exposed to different transmissions at any given time. While *collision avoidance* is in general an unsolvable task, techniques such as *collision detection* have been developed to reduce the amount of collisions that can take place in a wireless station [40].

The Internet Layer *The internet layer defines an official packet format and protocol called IP (Internet Protocol). The job of the internet layer is to deliver IP packets where they are supposed to go. Packet routing*

is clearly the major issue here, as is avoiding congestion.

The Transport Layer *The layer above the internet layer in the TCP/IP model is now usually called the transport layer. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer. Two end-to-end transport protocols have been defined here. The first one, TCP (Transmission Control Protocol), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in the internet. It fragments the incoming byte stream into discrete messages and passes each one on to the internet layer. At the destination, the receiving TCP process reassembles the received messages into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.*

The second protocol in this layer, UDP (User Datagram Protocol), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own. It is also widely used for one-shot, client-server-type request-reply queries and applications in which prompt delivery is more important than accurate delivery, such as transmitting speech or video.

The Application Layer *On top of the transport layer is the application layer. It contains all the higher-level protocols. The early ones included virtual terminal (TELNET), file transfer (FTP), and electronic mail (SMTP).[..] Many other protocols have been added to these over the years: the Domain Name System (DNS) for mapping host names onto their network addresses, NNTP, the protocol for moving USENET news articles around, and HTTP, the protocol for fetching pages on the World Wide Web, and many others.*

1.2 The Need for Formal Methods

Wireless networks are complicated systems. Starting from the broadcast of an electromagnetic wave, there are lots of technical details to be considered, and protocols to be implemented, before wireless stations can interconnect to each other in a network and provide services to end users.

One of the most important questions that needs to be addressed is to establish if the behaviour of a wireless network accomplishes the task for which it was designed; in other words, we want to check whether the behaviour of a wireless network is consistent with respect to a specification. For example, we want to check if in a wireless network any two stations can connect each other to exchange data, and if such data are delivered correctly from the source to the destination.

Due to the complicated nature of wireless networks, in which communication is broadcast and the range of transmission of a station is limited, an informal analysis of their behaviour often leads to neglect errors in protocols which compromise the correct functioning of the overall network.

For example, imagine that we want to analyse a network in which two different nodes, n_1 and n_2, are not in each other's range of transmission. However, node n_1 can broadcast messages to other two nodes n_3, n_4, and the latter nodes can broadcast messages to node n_2. Imagine also that we have the following informal specification, in terms of pseudo-code, of the behaviour of each of the nodes.

- **Node** n_1:
  broadcast a stream of frames $f_1, \cdots f_j$
- **Node** n_2:
  upon receiving a frame f*:*
  Broadcast the frame f
- **Node** n_3:
  upon receiving a frame f*:*
  Broadcast the frame f

- **Node** n_4:
  upon receiving a frame f*: Do Nothing*

The behaviour of this rather simple network is very intuitive. Nodes n_3, n_4 act as forwarders of the frames sent by n_1, making them available at node n_4. However, the question *is* n_2 *able to detect all the frames sent by the station* n_1*?* has not an easy answer. In our naive design of the network, several technical details were neglected. Among others, we emphasise the following:

- Is the communication reliable? Whenever a node broadcasts a frame f, are we sure that the same frame will be detected by the stations in the sender's range of transmission?
- Is the communication subject to collisions? If both n_3 and n_4 broadcast a frame at the same time, what will be detected at node n_2?
- What happens if node n_1 broadcasts a second frame before the first one has not been forwarded by either n_3 or n_4? In other words, do we wait for the code being run by such nodes to be executed before processing another request (and in which case, are these nodes equipped with an internal memory to store the received frames which have not yet been forwarded), or does the broadcast of the second frame take place immediately after it has been received?

As it is easy to see, the informal pseudo-code we have provided for the nodes in the network we want to analyse does not allow us to infer any information about the behaviour of the network; this is because several technical details have been neglected.

In the example above the design of the network is very naive; in general, it is often the case that the behaviour of stations in a network is well-known, even at an informal level. For example, in the analysis of protocols at the *MAC sub-layer* it is assumed that stations are subject to collisions, while this assumption is dropped in protocols running either at the *Internet Layer* or above.

However, in many cases protocols consist of several thousands lines of codes, and many assumptions have to be made on the network topology in order to ensure their correct behaviour. Techniques such as testing and simulation of the protocol in different network environments do not provide a definite proof of its correctness. In fact, it is often the case that pathological situations which cause the malfunctioning of the protocol are discovered. In these cases there is the possibility that the error in the protocol is not discovered for years after its first implementation.

On the other hand, formal methods have been often used to verify, with mathematical rigour, the correct behaviour of large programs and complicated system. During the last years the research community of formal methods has also become interested in wireless systems [47, 46, 48, 52].

The main idea lying at the basis of formal methods for the analysis of (wireless) systems is that of defining a language (or calculus) for describing, at some level of abstraction, the behaviour of the components of the system to be modelled. The language usually includes a syntax, in which the primitives that can be used by the system are defined, and an operational semantics, which describes how the system evolves after performing an action, such as executing a primitive. This approach enables the possibility to analyse formally the behaviour of a system being modelled.

In the case of wireless networks, there are many questions to be addressed for defining a formal language for describing their behaviour. These include the following:

- Which kind of mathematical structure do we use to represent a wireless network? Since wireless stations are spatially distributed, there is the need to include the topology of a network in its formal description. Graphs and metric spaces have been often used to solve this problem [52, 48].
- What is the expressive power of a wireless station, and what are the primitives that such a station can use? For example, we could limit the capacity of a wireless station to broadcast and receive messages, or we could also add other features like storing received messages in the internal memory of the node, or allowing the code running at a wireless station to behave probabilistically.

- How is communication between stations in a wireless network modelled? This question is very important, for it establishes the level of abstraction (in the *TCP/IP* reference layer) which we use to define the formal behaviour of wireless stations. For example, we can assume broadcast and receive primitives to be reliable, meaning that errors cannot occur during a transmission. Intuitively speaking, this assumption corresponds to modelling wireless networks at a level of abstraction corresponding to the *Internet Layer* of the *TCP/IP model*. Another possibility would that of assuming communication to be unreliable and collision prone, in which case the calculus is modelled at the level of abstraction of the *MAC Sublayer*.

Different answers to such questions lead to different process calculi, each of which has its own strengths and weaknesses. For example, it would not be possible to formalise the behaviour of a protocol at the *MAC sublayer* if the calculus is designed by assuming that communication between nodes is reliable; however, we point out that it would be extremely difficult to check the correct behaviour of routing protocols in a setting where communication is assumed to be collision-prone, while the same task could be accomplished rather easily in a calculus where communication between different stations is assumed to be reliable.

Note that if we define a calculus to model protocols at the n-th layer of the *ISO/OSI reference model* and we provide an implementation of its transmission primitives in a calculus defined for modelling protocols at a lower level of the protocol stack (e.g. at the $n-1$-th layer), then we can encode any implementation P of a protocol in the former calculus to an implementation Q in the latter; in fact Q can be defined by substituting any occurrence of a transmission primitive in P with its implementation defined in the lower level calculus.

Once a process calculus has been chosen to define wireless network, the main question to be addressed is how to determine if its behaviour is correct with respect to a given specification. This task is not easy to accomplish, for specification of services are usually given in an informal way.

One of the possibilities, which is the one we follow in this thesis, is that of establishing a notion of *behavioural equivalence* between networks. As the name suggests, two networks are behaviourally equivalent if they exhibit the same behaviour. However, there are different notions of behavioural equivalence which have been proposed in the literature for process calculi, such as *bisimulation*, *testing equivalences* and *failure models*; changing the notion of behavioural equivalence we use in our calculus also change the underlying theory. For example, two networks could be deemed to be equivalent with respect to some notion of behavioural equivalence, but they could be distinguished if a different behavioural equivalence is used. See [67] for a detailed description.

Having a process calculus and a notion of behavioural equivalence to relate its terms, we can check whether the behaviour of a network is consistent with an informal specification as follows:

- As the very first step, we define a network as a *model* of the informal specification of the required behaviour. The behaviour of the model should be consistent with such an informal specification. For it is not possible to prove formally this statement, the model should be as simple as possible. In this way it can be easily checked that, at least informally, its behaviour is consistent with the informal specification.

- We can now check whether the behaviour of a (usually complicated) network, formalised as a term of our process calculus, is consistent with the informal specification by checking if it is behaviourally equivalent to the network we have defined as a model of the specification.

The last step in the procedure above is a mathematical proof that the network behaves (in the established framework) as the model used to represent an informal specification. That is, it is a proof that the behaviour of the network is correct with respect to its informal specification, under the assumption that the model has been defined correctly.

Since wireless networks often contain thousands of wireless stations, each of which run a protocol consisting of several thousands lines of code, it would be rather complicated, if not impossible, to certify its correct behaviour without using formal methods. Quoting *Edsger W. Dijkstra*, *program testing can be used to show the presence of bugs, but never to show their absence*. Formal methods, on the other hand, can be used both

to prove mathematically that a network behaves as expected, or they can help to discover errors in ill-behaved networks.

1.3 Literature Review

In this Section we describe briefly the papers that inspired our research, as well as other papers reporting recent results in the area of formal methods for broadcast systems and formal methods. To avoid any confusion, we grouped the works that we describe into three different strands; the first one describes techniques and results for the formal analysis of concurrent systems. The second strand focuses on the work that has been carried out by the research community in the area of probabilistic concurrent systems, while the last one focuses entirely on the results that have been obtained for broadcast calculi, with particular emphasis on calculi for describing wireless network.

1.3.1 Behavioural Equivalences

In this thesis we develop behavioural equivalences for relating wireless networks. As we have already mentioned, different notions of behavioural equivalences for concurrent systems have been developed in the literature. Here we describe some of the most important works that inspired the contents of this thesis.

A technical report from *Van Glabbeek* [67] provides a very detailed description of such behavioural equivalences by putting emphasis on the difference between linear time and branching time theories; the author also provides a detailed comparison between the various notions of behavioural equivalences described in the paper in terms of their distinguishing power. In a sequel to the paper [68] the introduced behavioural equivalence are analysed in a setting in which processes can perform silent moves; intuitively, these are moves which cannot be detected from an agent which observes a system from an outside point of view.

The notions of behavioural equivalences we will need in this thesis correspond to the testing preorders of *de Nicola* and *Hennessy* and to the reduction barbed congruence of *Milner* and *Sangiorgi*.

Testing preorders were first introduced in [17], and their theory is developed in a more detailed fashion in [33]. The main idea is that of allowing an external agent called test, equipped with a success predicate ω, to interact with a concurrent system (or process) to check whether the latter satisfies a property for which the test was designed for; in this case the test reaches a successful configuration, that is one in which the success predicate ω is satisfied. Due to the non-deterministic nature of processes, two different notions of *passing a test* are developed; the first one, process P may-pass the test T, if the experiment obtained by letting P interact with T leads to a successful configuration. The second one, P must-pass T, holds whenever every computation of the interaction between P and T leads to a successful configuration. Each of these relations induces a *testing preorder* for relating processes. Process P is may-related to process Q, written $P \sqsubseteq_{\text{may}} Q$ if whenever P may-pass T for some test T, then Q may-pass T; the preorder $\sqsubseteq_{\text{must}}$ is defined similarly, using the must-pass testing relation. Each of the developed preorder induces a behavioural equivalence in the standard way. The authors provide a characterisation result for each of the proposed testing preorders; specifically, they prove that the may-testing preorder is captured by trace inclusion, while the must-testing preorder is captured by acceptance sets.

The other notion of behavioural equivalence we will use in this thesis corresponds to reduction barbed congruence. This has been proposed for the first time in [49] with respect to the process calculus *CCS*. The main idea is that two processes are equivalent if they have the same observations under all possible reductions in all possible contexts. Here much depends from the notion of observation which is provided; in the paper an observation on a channel corresponds to a CCS process which can synchronise along such a channel. In this paper it is proved that reduction barbed congruence is compositional with respect to all the CCS operators, with the only exception being the non-deterministic choice operator +.

Other works focusing on concurrency theory that have been proved useful while developing this thesis are [1, 16]. The first work presents the basic notions for the analysis of concurrent systems, putting particular

emphasis on the process calculus *CCS* and its timed variant, and introducing the reader to both strong and weak bisimulation and their modal characterisation. The second work introduces the reader to bisimulation and concurrent techniques; a survey of other behavioural equivalences, such as the testing equivalences, is also given in a dedicated chapter.

1.3.2 Probabilistic Concurrent Systems

Probabilistic concurrent systems have been widely analysed in the last two decades, and several models have been proposed in the literature to describe them formally.

In [61] probabilistic automata are introduced; these can be viewed as input/output automaton in which the transitions performed by a state are described as probability distributions over couples of actions and states. Different notions of (both weak and strong) probabilistic simulations and bisimulations are introduced as extensions of their non-probabilistic counterparts, and a comparison between a probabilistic version of the temporal logic CTL is made.

In [60] an extension of the testing preorders of *de Nicola* and *Hennessy* to probabilistic automata is defined; the author prove a characterisation result for each preorder, although he remarks that for their approach to work it is necessary to include multiple success predicates in the testing framework. They note that, due to the both non-deterministic and probabilistic nature of their model, each computation of a probabilistic automaton interacting with a test gives rise to a success probability; automata are then compared in terms of the different probabilities with which they pass a test. Another approach for extending the testing preorders to probabilistic automata is proposed in [39].

In [13] a different, generative model for formalising probabilistic processes is considered, and a testing preorder in the style of *Hennessy* and *de Nicola* for such processes is defined defined. In the proposed model, for any given state s of a process and an action α, either s can perform the action α, in which case the resulting state is chosen according to some probability distribution, or it cannot perform the action at all. The authors provide a characterisation result in terms of probabilistic traces.

In [38] a third model, *probabilistic labelled transition system* (pLTSs) are used to model probabilistic processes; in this case a transition maps a state into a distribution of states, thus distinguishing completely probabilistic choices from non-deterministic ones. The authors extend the definition of the may-testing preorder to pLTSs, by comparing processes according to the maximal probabilities of success they induce when composed with a test. Probabilistic simulations are defined for states of pLTSs; the authors show that these are sound and complete with respect to the proposed testing preorder.

In [20, 19] testing theories for pLTSs are also considered; here the authors define the may-testing preorder by comparing the set of all the possible success probabilities that are generated by a process interacting with a test; they also define the must-testing preorder accordingly. In the papers a mathematical theory for describing the behaviour of a pLTS in the long run is developed; this is done by lifting in a monadic fashion the transition relations of states in a pLTS to distribution of states. The paper focuses on the characterisation of both the may and must testing preorders, respectively in terms of simulations and probabilistic simulations.

The theory developed in [20, 19] has been widely used in the last years for reasoning over pLTSs. Among other works, we cite [21, 18, 22, 35]. Further, pLTSs have been also employed to define wireless networks which exhibit both non-deterministic and probabilistic behaviour [44, 9, 8]. We defer the description of the contents of these works to the next Section.

1.3.3 Broadcast Calculi and Wireless Networks

In this Section we describe the work that has been performed in the field of formal methods for wireless networks and broadcast communication. Most of the papers described in this section have been proved to be very useful in establishing a basis for the research carried out in this thesis.

The first paper describing a process calculus for broadcast systems, **CBS**, is [55]. In this paper the author presents a simple process calculus in which a synchronisation between a sender and a receiver is modelled

as an output action, rather than an internal activity as in standard process calculi such as CCS. This allows multiple receivers to detect a message sent by a sender, thus implementing broadcast communication. In [32] different notions of barbed congruence for a variant of CBS are introduced; these correspond to strong barbed congruence and weak barbed congruence. For each of them, a characterisation result in terms of strong and weak bisimulation, respectively, is proved.

Another calculus to model broadcast systems known as the $b\pi$-calculus, inspired by both CBS and the π-calculus [58], is introduced in [23]; as the author points out, broadcast communication is modelled in the same style of CBS. In this paper the authors define three different behavioural equivalences, corresponding to barbed congruence, step equivalence and labelled bisimilarity. The author proves that such behavioural equivalences coincide.

In [24] the authors define both the may and must testing preorders for processes of the $b\pi$-calculus, and they prove a characterisation result for each of them. The main contribution here lies in the characterisation of the must-testing preorder; as the authors point out, in fact, broadcast communication leads to a non-standard characterisation of the latter. In particular, the non-blocking nature of broadcast actions does not allow acceptance sets to be used in their characterisation result.

In the last decade, broadcast calculi have been modified in several ways by equipping processes with a topological structure, thus modelling wireless networks; the idea is that of representing a process as a set of locations, running different code for broadcasting and receiving messages; the topology defined for a process establishes how communication is modelled, for example by letting only some locations being able to detect the messages broadcast at another one.

In [53] the authors propose to model the topological structure of a network by using a connectivity graph; a process is viewed as a set of locations running code, while a graph whose vertices are locations is used to determine how communication is carried out. Intuitively, a transmission originated at a given location can only be detected by those vertices which are connected to the former. The transition relation of processes is defined as parametric in a connectivity graph. This framework has been proposed by the authors as a basis for the analysis of security protocols in wireless networks.

In [52] an allocation environment is used to represent the topological structure of a wireless networks. A wireless network is intended as a parallel composition of processes, each of which is associated with a set of locations to which the process belongs and a probability distribution over locations; intuitively, the latter describes the probability with which a message broadcast by the process is detected at a given location.

In [26] the authors propose a *restricted broadcast process theory* to model wireless networks. Here a network consists of a parallel composition of different processes; each process is associated with a location name, and a function between locations to sets of locations is used to represent the network topology. The author propose the standard notion of weak bisimulation as the behavioural equivalence to be used to relate networks and they show a case study in which they prove the correctness of a routing protocol.

In [27] an extension of the restricted broadcast process theory, the *Computed Network Theory*, is proposed; here the expressive power of a network is augmented through different operators. For the resulting calculus, a variant of strong bisimulation is defined and proved to be a congruence. The main result in the paper is a sound axiomatisation of the strong bisimulation, thus enabling equational reasoning for wireless networks. The authors also show that the proposed axiomatisation is complete in a setting where only non-recursive networks are considered. The *Computer Network Theory* framework is also used in [28] to check properties of mobile networks; the author show how both the equational theory and model checking can be used to verify the correctness of a routing protocol.

In [62] the authors view a network as a collection of processes, each of which is associated with one or more groups. Processes which belong to the same group are assumed to be neighbours; as a consequence, a broadcast performed by a process can be detected by all the processes which belong to at least one group of the broadcaster. The authors show that in their framework state reachability is a decidable problem; further, they introduce different notions of behavioural equivalences, based on late bisimilarity and its weak variant, and they show that such equivalences are in fact congruences. Finally, they apply their calculus by formalising

and analysing the behaviour of a leader election protocol and a routing protocol.

In [48, 43] the authors describe wireless networks by using metric spaces; they assume that a network consists of a set of processes, each of which has an associated location and a radius of transmission; a metric distance over the set of locations is assumed to determine how communication is modelled. The authors describe the behaviour of a wireless network in terms of both a reduction semantics and a labelled transition semantics. These two semantics are proved to be equivalent up-to a notion of structural congruence. We remark that in their paper the authors assume that a communication between two stations consists of two phases, one for the beginning and one for termination. These allows the authors to model collision-prone communication.

Another calculus for wireless networks in which collision-prone behaviour is taken into account is described in [47]. In their work, the authors describe a network as a set of processes running in parallel, each of which has a location name and a semantic tag associated with it; the latter consists of a set of locations names and it corresponds to the set of locations which can detect messages broadcast by the process. The calculus includes a notion of discrete-time, in the style of [36], and broadcasts of messages start and end at different time slots. The authors develop a notion of barbed congruence for wireless systems and they propose a sound, but not complete, characterisation result in terms of weak bisimulation.

In recent years, a particular effort has been made to formalise probabilistic wireless systems. In [63] the authors propose a model in which the topological structure of a network is represented as a graph whose vertices are locations; further, they assign to each edge in the graph a (possibly unknown) probability as a likelihood estimate of whether a message broadcast by a location at the starting end-point of an edge will be delivered to the location at the terminal end-point of the same. The proposed model also allows the network topology of a system to evolve according to a probabilistic mobility function. The authors prove that, in the proposed calculus, the logical equivalence defined over a variant of PCTL coincides with weak bisimulation.

In [44] the authors define a language for wireless networks in which the code running at network locations contains both non-probabilistic and non-deterministic behaviour. The topological structure of a network is defined in the same way of [47]; the authors introduce a notion of simulation, parametrised in a probability value, in order to capture the concept of two networks exhibiting the same behaviour up-to such a probability. The model used to represent wireless networks and define their formal behaviour is that of a pLTS.

In [29] a different approach is made to formalise a wireless network. The authors identify a network as a set of processes associated with a location address and a queue, representing the data at the datalink layer that a station has not yet broadcast. The calculus they use is a probabilistic generalisation of the restricted broadcast process theory of [27]; here the sending primitive consists of a message to be broadcast and a probability rate, representing the likelihood that such a message will be sent. The model used to describe the behaviour of a system is that of **Continuous Time Markov Automata**.

In [8, 9] the authors define a network in terms of a connectivity graph, whose vertices are wireless nodes, in a style similar to [53], and the code running at any node contains both non-deterministic and probabilistic behaviour; the model used to describe the behaviour of a network is again that of a pLTS. The authors suggest that some nodes have no code associated, for they can be used by external agents to interact with the network. They define both the may and must testing preorder and they exhibit sound, but not complete, proof techniques for both of them. The contents covered in [8, 9] are analysed in detail in Part II of this thesis.

1.4 Overview of The thesis

In this thesis we develop different process calculi for wireless networks; each of them differs from the other in either the level of abstraction at which communication primitives in broadcast systems are defined, or in the computational power granted to wireless stations. The thesis is divided in three parts, one for each of the calculi we develop. We give a brief summary of the contents of each part.

1.4.1 Part I - High level Wireless Networks

The first calculus we develop is very simple, and it can be used to model networks at a high level. The calculus relies on the assumption that communication is perfect; whenever a node broadcasts a message and a neighbouring node is waiting for a message to be received, then it is ensured that the latter will receive the message originally broadcast by the sending node. In practice, this corresponds to assuming that nodes use some protocol at both the Datalink and MAC layers to achieve point-to-point reliable communication.

Also, when developing the calculus, we assume an enumerable set of channels that nodes can use to transmit a value. This could seem in contrast with the fact that the communication medium is unique in wireless networks. However, there are several multiple access techniques that can be used to create virtual channels, such as FDMA, TDMA and CDMA [40]. Introducing multiple channels corresponds to assuming that the nodes agree on some multiple access mechanism.

The development of this calculus is carried out in **Chapter 2 - A simple language for networks**. Here we define the mathematical structures that will be used to represent wireless networks; also, we define how such structures evolve after performing some activity (such as broadcasting a message or receiving one) through SOS inference rules.

In **Chapter 3 - Behavioural theories for networks** we address the topic of composing wireless networks; this task is not easy to accomplish. Due to the presence of a topological structure in wireless networks, it is necessary to impose some constraints to be satisfied by networks which are being composed; for example, we do not want to compose two networks which have both code running at a given wireless station, for it would not be clear which would be the code running at such station in the composed network. Further, the way in which the topological structure of a composed network has to be be inferred from its components also has to be defined.

As we will see, we solve the problem of compositionality by using *interfaces*. These are stations which run no code, connected to other stations in the network (i.e. they can either receive or send messages to a subset of the stations which do run code in the network) and which can be used by external agents to join the network.

The proposed compositional theory is then exploited to define a testing framework in the style of Hennessy and de Nicola [17] and the induced may and must testing preorders. Testing preorders and their induced equivalences constitute the observational theories we use to relate networks in this part of the thesis. To the best of our knowledge, testing theories have never been defined for wireless networks.

In **Chapter 4 - Characterisation of the testing preorders** we focus on full abstraction results for the testing preorders introduced in Chapter 3. This amounts to elucidating a set of activities performed by a network which can be detected by its external environment; that is, we define an extensional semantics for networks, together with its weak variation. We provide a characterisation for both the may and must testing preorder, respectively in terms of traces and deadlock traces. We remark here that the proof methods we present are sound and complete; however, completeness comes at the price of introducing a non-standard version of weak transitions in the extensional semantics. Further, our results hold only for *finitary networks* and, in the case of the must-testing preorder, the additional hypothesis that a network is *strongly convergent* has to be included.

In **Chapter 5 - Applications** we show how the developed theory can be applied to the study of practical situations. We consider different problems which are particular to wireless networks and distributed systems; starting from an informal specification, we build a simple network for which it is easy to check that its behaviour is consistent with such a specification. Then we consider more complicated networks and we prove that their behaviour is correct with respect to the informal specification considered by showing its testing equivalence with the network we proposed as a model. In other words, the proposed networks constitute an implementation of the model.

The practical situations we consider in Chapter 4, include

- Connection-less routing, for which an implementation at the *Internet layer* of the TCP/IP is proposed and proved correct

- Connection-oriented communication, for which two different implementations are proposed and proved correct; the first one is an implementation at the *Internet Layer* which abstracts from distance-vector routing protocols, while the second one is an implementation at the *Transport Layer* which can be seen as an abstraction of the *TCP* protocol.
- Multicast communication, for which a very simple implementation is proposed and proved correct
- Virtual Shared Memory in a distributed system with two users. This is probably the most complicated application we propose; again, an implementation is proposed and proved correct.

In all the cases, the implementation we propose for a given model is parametric, in the sense that we define the properties to be satisfied by a network for our equivalence result to hold rather than defining the network as a term of our language.

1.4.2 Part II - Probabilistic Wireless Networks

In this part of the thesis we extend the calculus developed in Part I to allow nodes to exhibit probabilistic behaviour. Again, we define behavioural theories based on testing and we provide proof techniques to compare networks with respect to the defined behavioural preorders. The contents of this Part are an extension of the results presented in [8, 9].

In **Chapter 6 - Probabilistic Wireless Networks**, we first provide the mathematical basis needed to model systems which exhibit probabilistic behaviour. The formal model used is that of *probabilistic Labelled Transition Systems* (pLTSs), and the theory presented is just a review of the contents discussed in [19, 20]. Then we focus on the probabilistic extension of the calculus for wireless networks. In the final part of the Chapter we introduce the testing framework and the induced testing preorders in the probabilistic setting.

In **Chapter 7 - Proof Methods for Probabilistic Networks**, we develop proof techniques to establish whether two networks are related with respect to one of the testing preorders. In particular, we show that probabilistic simulations are sound with respect to the may-testing preorder, and deadlock simulations are sound with respect to the must-testing preorder. Again, for our results to hold it is necessary to restrict our attention to a finitary setting.

We also show that our proof techniques are not complete. This result is rather surprising; we put particular emphasis on the reasons that cause the introduced proof techniques to fail being complete, and we prove that the current state of the art in the theory of probabilistic system is inadequate to provide a characterisation of the two testing preorders. In particular, simulations can not be revised to obtain a complete characterisation of the testing preorders.

In **Chapter 8 - Applications for Probabilistic Networks** we address the same topics as in Chapter 5 in the probabilistic setting. Specifically, probabilistic routing and probabilistic reliable connections analysed; in these cases, we relate (via the developed proof techniques) probabilistic networks to the respective model defined in Chapter 5.

1.4.3 Part III - Time and Collisions

In the last part of the thesis we develop a simple timed calculus which deals with unreliable communications; in this calculus, communication is timed; each transmission begins and ends at different time instants. While communication along a channel is in progress, we say that the channel is exposed. A station which is exposed to more than one transmission in a given instant of time is not able to infer the message associated with such transmissions. In few words, the calculus introduces the possibility for collisions to happen as the result of multiple broadcasts along the same channel.

In **Chapter 9 - A Timed Calculus for Collisions** we first develop the calculus, putting an emphasis on the behaviour of collisions in practice to provide evidence that the semantics of the calculus reflects the practical behaviour of wireless networks.

In Chapter **Chapter 10 - Barbed Equivalence and Full Abstraction** we define a behavioural equivalence based on the notion of *reduction barbed congruence* [49] and we provide a full abstraction result for it in terms of weak bisimulation. Finally, we provide some simple applications in which we employ our full abstraction results to show that two networks are barbed congruent.

In **Chapter 11 - Conclusions** we give a brief description of the future directions of research and of possible applications of the theory developed in this thesis.

Part I

High level Wireless Networks

Chapter 2

A Simple Language for Networks

In this chapter we present a basic process language for modelling wireless networks. The topology of a wireless network is assumed to be static; that is, it never changes as the consequence of some activity of the network. A *connectivity graph* is used to represent the topology of the network; intuitively speaking, vertices of a connectivity graph represent nodes (or locations) of wireless networks, while a directed edge from one vertex to another models the possibility for the latter to detect messages transmitted by the former. The idea of using mathematical structures to represent the topology of wireless networks has been already used, among others, in [52, 53, 48, 47, 46, 30].

Communication in our calculus is *broadcast*. The transmission of a message is a non-blocking action, and a message broadcast by a node can be detected only by those nodes which are connected to it in the connectivity graph. Broadcast communication has been first introduced in [55], and it has been studied in deeper detail in [23].

In this chapter we first define formally the concept of wireless networks on which our calculus is based. This is done in Section 2.1. Here we first present a grammar whose terms define the code that nodes in a wireless networks are running; then we integrate the information provided by such terms with the connectivity graph of the network. As we already mentioned, this is needed to model how nodes communicate each other in our calculus. As we will see, we impose some natural requirements on the structure of terms and connectivity graphs, which correspond to our intuition of what a wireless network is. For example, we do not focus on networks in which a node has some code associated to it, but it is not defined as a vertex in the connectivity graph of the network.

In Section 2.2 we present a *reduction semantics* for wireless networks. This defines the dynamics of wireless networks in our calculus. In Section 2.3 we present a labelled transition semantics for wireless networks; then we show in Section 2.4 that there is a close relationship between these two semantics.

The Chapter ends with a brief comparison with related work in Section 2.5.

All the topics presented in this chapter are analysed thoroughly, often by providing simple examples to illustrate the intuitive meaning of our definitions and the importance of our results.

2.1 The Calculus

The calculus we present is designed to model broadcast systems, particularly wireless networks, at a high level. We do not deal with low level issues, such as collisions of broadcast messages or multiplexing mechanisms [65]; instead, we assume that network nodes use protocols to achieve both perfect and dedicated communication between nodes [40, 56].
Basically, the language will contain both primitives for sending and receiving messages, and it will enjoy the following features:

(i) communication can be obtained through the use of different channels; though the physical medium for

M, N	::=		**Systems**
		$n[\![P]\!]$	Nodes
		$M \mid N$	Composition
		$\mathbf{0}$	Identity

P, Q	::=		**Processes**
		$c!\langle e\rangle.P$	broadcast
		$c?(x).P$	receive
		$P + Q$	choice
		if b then P else Q	branch
		$\tau.P$	pre-emption
		$A(\tilde{x})$	definitions
		ω	Success
		$\mathbf{0}$	terminate

Figure 2.1: Syntax

exchanging messages in wireless networks is unique, it is reasonable to assume that network nodes use some multiple access technique, such as *TDMA* or *FDMA* [65], to setup and communicate through virtual channels,

(ii) communication is broadcast; whenever a node of a given network sends a message, it will be detected by all the nodes in its range,

(iii) communication is perfect: whenever a node broadcasts a message and a neighbouring node (that is, a node in the sender's range) is waiting to receive a message on the same channel, then the message will be delivered to the receiver. This is not ensured if low level issues are considered, for problems such as message collisions [40] and nodes synchronisation [56] arise .

The language for system terms, ranged over by $M, N, L, \cdots$ is given in Figure 2.1. Basically a system term consists of a collection of named nodes, ranged over $m, n, l, \cdots$, at each of which there is some running code (or process) . The syntax for this code is a straightforward instance of a standard process calculus.

Process $c?(x)$ waits to receive some value along channel c; when a value v is received, the process evolves in $\{v/x\}P$; the latter is defined as process P, where all the free occurrences of variable x have been replaced by value v. Therefore, the input operator $c?(x)$ acts as a binding operator for the variable x. This induces the standard notions of closed system terms, closed expressions and α-conversion [37].

Process $c!\langle e\rangle.P$ first evaluates a closed expression e to some value v; then this value is sent along channel c, and the process above evolves in P. Here e is some expression from a decidable theory. We assume an evaluation function $[\![\cdot]\!]$ which maps closed expressions to values.

Process $\tau.P$ performs some internal activity, thus evolving in P.

Process $A(\tilde{x})$ corresponds to process definition. Here $\tilde{x}$ is a list of formal parameters for the process definition A. We use the notation $A(\tilde{x}) \Leftarrow P$ to associate the definition $A(\tilde{x})$ with process P. A list of expressions $\tilde{e}$ can be used to instantiate this process definition, provided that $\tilde{e}$ and $\tilde{v}$ have the same length. If $\tilde{e} = \langle e_1, \cdots, e_n\rangle$ for some $n > 0$, and $\tilde{v} = \langle [\![e_1]\!], \cdots [\![e_n]\!]\rangle$ is the list of values obtained by evaluating each of the expressions in $\tilde{e}$, then we can define the process $A\langle\tilde{e}\rangle$ to be equal to $\{\tilde{v}/\tilde{x}\}P$. With an abuse of notation, in the following we use $[\![\tilde{e}]\!]$ to denote the lifting of $[\![\cdot]\!]$ to lists of expressions. One could think of a process definition $A(\tilde{x}) \Leftarrow P$ as a function declaration whose formal parameters are the variables contained in $\tilde{x}$; similarly, the construct $A(\tilde{v})$ used for processes corresponds to a call to the defined function A, where $\tilde{e}$ corresponds to the list of actual parameters used for invoking such a function. Note that we stated that the actual parameters have to be evaluated to values when invoking a function call, thus giving rise to an *eager evaluation* of process invocation. On the other hand, we could have required expressions not to be evaluated, thus implementing a *lazy evaluation* strategy for

process invocation. In practice, whether eager or lazy evaluation is used for invoking a process definition does not influence any of the topics which we cover in this thesis.

The construct if b then P else Q is the standard *if-then-else* construct; again, b is a boolean statement, and we assume that there exists an evaluation function $[\![\cdot]\!]$ which maps them in the set $\{\text{true}, \text{false}\}$. Note that, with an abuse of notation, we used the symbol $[\![\cdot]\!]$ to evaluate both expressions and boolean statements. However, this does not cause any confusion, as it will always be clear from the environment which of these functions is being considered.

Finally, we have a clause in the Grammar of Figure 2.1 which states that ω is a process; such processes are not used in this chapter, and the discussion about the role it plays in the calculus is deferred until Chapter 3.

We assume that the constructors $c!\langle v\rangle$ and $c?(x)$ bind stronger than non-deterministic choice; also, the latter binds stronger than the matching construct. As an example, process if $x = v$ then $c!\langle x\rangle$ else $c?(x).P + d?(x).Q$ stands for if $x = v$ then $c!\langle x\rangle$ else $((c?(x).P) + d?(x).\mathrm{Q})$.

We assume the following countable sets: a set **Nodes** for node names, a set **Ch** for channels, a set **Val** for values, and a set **Var** for variables. The parallel composition operator $(\cdot \mid \cdot)$ is used to compose different nodes running some code;

We only consider the sub-language of *well-formed system terms* in which all node names have at most one occurrence. We use sSys to range over all closed well-formed terms. A well-formed system term can be viewed as a mapping that assigns to node names the code they are executing. A subterm $n[\![P]\!]$ appearing in a system term M represents node n running code P. We define the set nodes(M) to contain exactly those nodes which have some code associated in the system term M; formally, this set can be defined by the following inductive definition

$$\begin{aligned} \mathrm{nodes}(\mathbf{0}) &= \emptyset \\ \mathrm{nodes}(n[\![P]\!]) &= \{n\} \\ \mathrm{nodes}(M \mid N) &= \mathrm{nodes}(M) \cup \mathrm{nodes}(N) \end{aligned}$$

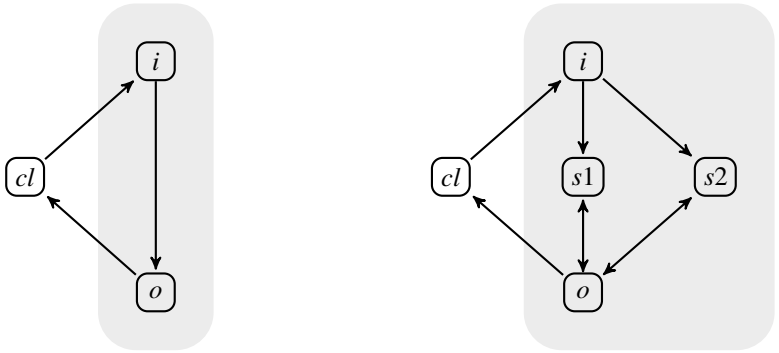
Additional information such as the connections between nodes of a network is needed to formalise communications between nodes. The network connectivity is represented by a directed graph $\Gamma = \langle \Gamma_V, \Gamma_E \rangle$; here Γ_V is a finite set of nodes and $\Gamma_E \subseteq (\Gamma_V \times \Gamma_V)$. Henceforth we use the term *connectivity graph* in lieu of directed graph, and the term *connection* for edges of directed graphs.

We use the more graphic notation $\Gamma \vdash m$ in lieu of $v \in \Gamma_V$ and $\Gamma \vdash m \to n$ for $(m,n) \in \Gamma_E$. Intuitively $\Gamma \vdash m \to n$ means that messages broadcast from node m can be received by node n[1]. Similarly, we use $\Gamma \vdash n \leftarrow m$ for $\Gamma \vdash m \to n$, and $\Gamma \vdash m \leftrightarrow n$ if both $\Gamma \vdash m \to n$ and $\Gamma \vdash m \leftarrow n$ are true. Also, we define $\Gamma \vdash m \rightleftarrows n$ to be true if either $\Gamma \vdash m \leftarrow n$ or $\Gamma \vdash m \to n$ hold. Finally, we use $\Gamma \vdash m \nrightarrow n$, $\Gamma \vdash m \nleftarrow n$, $\Gamma \vdash m \nleftrightarrow n$ and $\Gamma \vdash m \not\rightleftarrows n$ for the negations of $\Gamma \vdash m \to n$, $\Gamma \vdash m \leftarrow n$, $\Gamma \vdash m \leftrightarrow n$ and $\Gamma \vdash m \rightleftarrows n$, respectively.

A *network* consists of a pair $\langle \Gamma, M \rangle$, representing the system term M, from sSys, executing relative to the connectivity graph Γ. Henceforth we will use the notation $\Gamma \rhd M$ to denote a network $\langle \Gamma, M \rangle$.

Note that we assume that the topology of a network (i.e. its connectivity graph) is static; this is emphasised in Section 2.2, where we will see that reductions for networks do not affect their topologies. However, we point out that our framework can be extended to deal with mobile networks, that is networks whose topology changes according to some *mobility policy*; this can be done by specifying a *mobility relation* that describes how connectivity graphs can evolve in a computation; then, when defining the reduction semantics of networks (see Section 2.2), it is sufficient to introduce a rule that allows the topology of a network to evolve according to the considered mobility relation. Dealing with mobile networks is a topic outside the scope of this thesis; throughout part I we will only deal with networks whose topology is assumed to be static; this is because our goal is that of establishing a foundation theory for wireless networks, and in our own point of view we believe it better to accomplish this task for a simple framework rather than trying to work directly with complicated

[1] Note that no information is given whether the opposite is also true. The notation $\Gamma \vdash m \to n$ does not include any information on whether $\Gamma \vdash n \to m$ or not.



$\mathcal{M} = \Gamma_M \rhd i[\![A_i]\!] \mid o[\![A_o]\!] \mid m[\![A_m]\!]$ $\quad$ $\mathcal{N} = \Gamma_N \rhd i[\![A_i]\!] \mid o[\![B_o]\!] \mid s1[\![A_1]\!] \mid s2[\![A_2]\!]$

Figure 2.2: Example networks

objects such as mobile networks.

Despite our definition of network being rather simple and intuitive, it is easy to provide examples of networks which are not consistent with the intuitive idea of what a wireless network is. For example, we can exhibit a network $\Gamma \rhd M$ where an arbitrary node m has some code associated to it, but for which no connectivity information is provided in Γ; that is, there exists a node m such that $m \in \mathrm{nodes}(M)$, but $\Gamma \not\vdash m$. Therefore, we restrict our attention only to a specific class of networks, whose instances we denote as *well-formed networks*.

Definition 2.1.1. The network $\Gamma \rhd M$ is well-formed if:

(i) $M \in \mathsf{sSys}$

(ii) for any node name m, $\Gamma_E \vdash m \leftrightarrow m$

(iii) $\mathrm{nodes}(M) \subseteq \Gamma_V$

□

Requirement (i) says that each node in a system term can have only one code associated. Requirement (ii) establishes that self loops are not allowed in connectivity graphs. This corresponds to the intuition that, in wireless systems, a node cannot listen to its own transmission. In fact, in wireless networks, communication is *half-duplex* [57]: *at any given time, a node can only transmit or receive information*. Requirement (iii) imposes that, whenever a node appears in a system term, then information for it has to be provided in the connectivity graph.

We use Nets to denote the set of well-formed networks, and in the sequel we will assume that networks are well-formed. We use the symbols $\mathcal{M}, \mathcal{N}, \mathcal{L}$ to range over arbitrary (well-formed) networks, and apply operations such as $\mathrm{nodes}(\mathcal{M})$ in the obvious manner.

Note that in a network $\Gamma \rhd M$ there can be nodes which do not have any code associated. Intuitively, this means that there is no knowledge about the code which is being run by such nodes. However, such information can be provided by considering a (well-formed) network of the form $\Gamma \rhd M|N$, where the system term N defines the code for (not necessarily all) those nodes which do not appear in $\mathrm{nodes}(M)$. Given a network $\Gamma \rhd M$, we use the term *external node* to refer to a node which has no code associated. The set of all external nodes of $\Gamma \rhd M$, given by $\Gamma_V \setminus \mathrm{nodes}(M)$, is called the *interface* of the network, and it is denoted by $\mathsf{Int}(\Gamma \rhd M)$. In contrast, all the nodes included in a system term M take the name of *internal nodes* of the network $\Gamma \rhd M$.

Example 2.1.2. Consider the network $\mathcal{M}$ described in Figure 2.2. Here and henceforth we use shading to distinguish internal nodes from external nodes in a network. For the network $\mathcal{M}$, let

$$\begin{aligned} A_i &\Leftarrow c?(x).c!\langle x\rangle.A_i \\ A_o &\Leftarrow c?(x).d!\langle h(f(x),g(x))\rangle.A_o \end{aligned}$$

There are three nodes in this network: cl, i, o. However, only the code for two of them, i and o, is specified, hence we have $\mathsf{nodes}(\mathcal{M}) = \{i, o\}$ and $\mathsf{Int}(\mathcal{M}) = \{cl\}$. The connectivity graph Γ_M specifies that node cl can send values to i, and it can receive them from o. Further, node i can broadcast values to node o.

Intuitively, the behaviour of this network is as follows; node i waits to receive some value v from cl, then it forwards it to node o; at this point, node o applies two different transformations to the received value, $f(\cdot)$ and $g(\cdot)$, then it uses the values obtained to perform a binary operation $h(\cdot)$. The value obtained as a result of this operation, which is $h(f(v), g(v))$ is then broadcast along channel d. The only node that can detect this broadcast is cl, as specified by Γ_M. However, no information about the code run by cl is given in the system term M, so that we do not know if this node is actually waiting to receive some message along channel d.

Now, in Figure 2.2 consider the network $\mathcal{N}$. Here let

$$\begin{aligned} A_1 &\Leftarrow c?(x).c_1!\langle f(x)\rangle.d?(x).A_1 \\ A_2 &\Leftarrow c?(x).c_2!\langle g(x)\rangle.d?(x).A_2 \\ B_o &\Leftarrow (c_1?(x).c_2?(y).d!\langle h(x,y)\rangle.B_o) + c_2?\langle y\rangle.c_1?\langle x\rangle.d!\langle h(x,y)\rangle \end{aligned}$$

In this network we have five nodes, of which only cl is external. In the connectivity graph Γ_N we have that node i can broadcast messages to both s_1 and s_2. Node o can receive messages from s_1 and s_2; further, it can also broadcast messages to these nodes and to the external node cl.

The behaviour of $\mathcal{N}$ can be summarised as follows; node i waits to receive some value v from node cl, then it forwards it along channel c. Such a value will be detected by both s_1 and s_2; the former applies the transformation $f(\cdot)$ to value v, then it broadcasts it along a channel c_1. Once the message has been broadcast, node s_1 waits to receive an input along channel d before evolving in its original configuration. The code in node s_2 is similar to the one of s_1, but the transformation $g(\cdot)$ is applied to v, instead of $f(\cdot)$; also, a different channel c_2 is used to broadcast the value $f(v)$ to node o.

Node o waits to receive two different messages, one along channel c_1 and the other along channel c_2, not necessarily in this order. The former one is the value $f(v)$ broadcast by s_1, while the other one is the value $g(v)$ broadcast by s_2. Once these values have been received, node o applies the binary operator h to them, then it broadcasts the value $h(f(v), g(v))$ along channel d.

This value can be detected by the external node cl; further, nodes s_1 and s_2 detect the transmission of this value too, after which they evolve in their original configurations, A_1 and A_2 respectively. □

We conclude this section by introducing a number of conventions that will be used in the following. Given a finite index set $I = \{i_1, \cdots, i_k\}$, a collection of nodes $\{m_i\}_{i\in I}$ and a collection of processes $\{P_i\}_{i\in I}$, the notation $\prod_{i\in I} m_i[\![P_i]\!]$ is used for the system term $\left(m_{i_1}[\![P_{i_1}]\!] \mid \cdots \mid m_{i_k}[\![P_{i_k}]\!]\right)$. If the index set I is empty, then both $\{m_i\}_{i\in I}$ and $\{P_i\}_{i\in I}$ are equal to the empty set $\emptyset$, and $\prod_{i\in I} n_i[\![P_i]\!]$ is defined to be exactly the system term $\mathbf{0}$. We assume that the operator $\prod_{i\in I} \cdot$ binds stronger than parallel composition, so that $\prod_{i\in I} m_i[\![P_i]\!] \mid M$ is actually the system term $(\prod_{i\in I} m_i[\![P_i]\!]) \mid M$.
The terms $c!\langle e\rangle$ and τ are used to denote $c!\langle e\rangle.\mathbf{0}$ and $\tau.\mathbf{0}$, respectively.

Sometimes we will identify networks, system terms and processes modulo *structural congruence*. For processes, structural congruence is the smallest equivalence relation which is a commutative monoid with respect to the choice operator $\cdot + \cdot$ and the empty process $\mathbf{0}$; further, it is preserved by unfolding process definitions and by evaluating boolean statements in branching constructs. For system terms, structural congruence is the smallest equivalence relation which is a commutative monoid with respect to the composition operator $(\cdot \mid \cdot)$ and the empty system term $\mathbf{0}$, and which is preserved by $(\cdot \mid \cdot)$.

Definition 2.1.3 (Structural Congruence)**.** 1. The binary relation $\equiv$ between processes is defined to be the smallest equivalence relation such that, for all processes P, Q, R, list of expressions $\tilde{e}$ and process definition $A(\tilde{x})$ for which $\tilde{x}$ and $\tilde{e}$ have the same length, it holds

(i) $P \equiv P + \mathbf{0}$,

(R-BCAST)
$$\frac{[\![e]\!] = v \quad \forall i \in I.\Gamma \vdash m \to n_i \quad \neg \mathrm{rcv}(M,c) \quad \forall n \in \mathrm{nodes}(N).\Gamma \vdash m \nrightarrow n}{m[\![c!\langle e\rangle.P + Q]\!] \mid \prod_{i\in I} n_i[\![(c?(x).P_i) + Q_i]\!] \mid M \mid N \multimap m[\![P]\!] \mid \prod_{i\in I} n_i[\![\{v/x\}P_i]\!] \mid M \mid N}$$

(R-TAU)
$$\frac{}{\Gamma \rhd m[\![\tau.P + Q]\!] \mid M \multimap m[\![P]\!] \mid M}$$

(R-STRUCT)
$$\frac{M \equiv N \quad \Gamma \rhd N \multimap N' \quad N' \equiv M'}{\Gamma \rhd M \multimap M'}$$

Figure 2.3: Reduction Semantics for (high level) networks

(ii) $P + Q \equiv Q + P$

(iii) if $[\![b]\!]$ = true, then if b then P else $Q \equiv P$

(iv) if $[\![b]\!]$ = false, then if b then P else $Q \equiv Q$

(v) if $A\langle\tilde{x}\rangle \Leftarrow P$ and $[\![\tilde{e}]\!] = \tilde{v}$, then $A\langle\tilde{e}\rangle \equiv \{\tilde{v}/\tilde{x}\}P$

2. The binary relation $\equiv$ between system terms is defined to be the smallest equivalence relation such that, for any node m, processes P, Q and system terms M, N, L

(i) if $P \equiv Q$ then $m[\![P]\!] \equiv m[\![Q]\!]$

(ii) $M \equiv M \mid \mathbf{0}$

(iii) $M \mid N \equiv N \mid M$

(iv) if $M \equiv N$ then $M \mid L \equiv N \mid L$

Given two networks $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$, we say that $\mathcal{M} \equiv \mathcal{N}$ iff $\Gamma_M = \Gamma_N$ and $M \equiv N$.

Finally, given a system term M and a channel c, the predicate $\mathrm{rcv}(M,c)$ is defined to be true if $M \equiv m[\![c?(x).P + Q]\!]|N$ for some processes P, Q, node m and system term N; false otherwise. In few words, the predicate $\mathrm{rcv}(\cdot, c)$ is true only for those system terms which contain at least one node which is waiting to receive a value along channel c. We also define, with an abuse of notation, the predicate $\mathrm{rcv}(\cdot, c)$ for processes, by letting $\mathrm{rcv}(P,c)$ be true iff $P \equiv c?(x).P' + Q$ for some P', Q. In the future it will always be clear from the context whether we are applying the predicate $\mathrm{rcv}(\cdot, c)$ to system terms or to processes.

2.2 Reduction Semantics

In this Section we develop a *reduction semantics* for our calculus. In Concurrency Theory, reduction semantics is a kind of *operational semantics* which is used to model the behaviour of a system which is isolated from the external environment [58, 34]. This is in contrast with another kind of operational semantics, the *labelled transition semantics*, which is discussed in Section 2.3.

The reduction semantics for networks is defined by focusing only on those nodes whose code has been provided. In other words, external nodes do not affect the behaviour of a network. Judgements take the form $\Gamma \rhd M \multimap N$ and can be inferred by using the rules provided in Figure 2.3, which are explained below.

Rule (R-BCAST) models local broadcast communication. When a node m broadcasts a value v along channel c in a network $\Gamma \rhd M$, then the only nodes which are affected by the communication are only those which

(i) are in the sender's range and

(ii) are waiting to receive a value along channel c.

The code for such nodes has the form $c?(x).P + Q$ for some processes P,Q. As each of these nodes can detect that value v has been broadcast, the code they are running evolves to $\{v/x\}P$, meaning that value v has been

received and all the free occurrences of variable x in P have been replaced with v. All the other nodes in the network are not affected by the broadcast performed by node m. This is because either

(i) they are not waiting to detect a value along channel c, or

(ii) they are not in the sender's transmission range

In particular, in Rule (R-BCAST) we identify the sub-system term referring to those nodes which cannot detect a value along channel c as M, while N denotes the sub-system term which refers to those nodes which are not in the sender's range of transmission. Note that in general, for a given network of the form $\Gamma \rhd L$, where $L \equiv m[\![c!\langle e\rangle.P]\!] \mid L'$, there are several ways to apply Rule (R-BCAST); in fact, nodes in nodes(L') which are not waiting to receive a value along channel c, and for which $\Gamma \vdash m \twoheadrightarrow n$, can be part of either the subsystem term M or N in a derivation performed by using rule (R-BCAST). However, since the code running at such nodes is not affected by the broadcast performed by node m, we are ensured that all these derivations lead to the same reduction. Further, distinguishing between nodes that cannot detect a value along channel c and nodes which are not in the sender's transmission range will help in the proof of some statements, such as Proposition 2.4.9.

Rule (R-TAU) is trivial; it models the capability of a node to perform an internal action without affecting any other node in the network.

Finally, rule (R-STRUCT) establishes that reductions are defined modulo structural equivalence.

Henceforth we will use the symbol $\rightarrowtriangle^*$ to denote the reflexive transitive closure of $\rightarrowtriangle$; thus $\Gamma \rhd M \rightarrowtriangle^* M'$ if the network $\Gamma \rhd M$ can evolve in network $\Gamma \rhd M'$ after performing a finite number (including 0) of reductions.

Let us look at how our reduction semantics can be used to infer the behaviour of networks.

Example 2.2.1. Consider the network $\mathcal{M}$ of Example 2.1.2. It is straightforward to note that this network cannot perform any reduction. This is because no node in the system term M can perform a broadcast nor an internal activity. Instead, consider the network $\Gamma_M \rhd M \mid L$, where $L = cl[\![c!\langle v\rangle.d?\langle x\rangle]\!]$. It is straightforward to show that

$$M|L \equiv cl[\![c!\langle v\rangle.d?\langle x\rangle]\!] \mid i[\![c?(x).c!\langle x\rangle.A_i]\!] \mid o[\![c?(x).d!\langle h(f(x),g(x))\rangle.A_o]\!] \tag{2.1}$$

This can be proved by applying definitions 2.1.3(1) and 2.1.3(2).

Since we have $\Gamma_M \vdash cl \rightarrow i$ and $\Gamma_M \vdash cl \twoheadrightarrow o$ We can apply rule (R-BCAST) to the right hand side of Equation (2.1) to infer

$$\begin{gathered}\Gamma_M \rhd cl[\![c!\langle v\rangle.d?\langle x\rangle]\!] \mid i[\![c?(x).c!\langle x\rangle.A_i]\!] \mid o[\![c?(x).d!\langle h(f(x),g(x))\rangle.A_o]\!] \\ \rightarrowtriangle \\ cl[\![d?\langle x\rangle]\!] \mid i[\![c!\langle v\rangle.A_i]\!] \mid o[\![c?(x).d!\langle h(f(x),g(x))\rangle.A_o]\!]\end{gathered} \tag{2.2}$$

The last system term is structurally equivalent to $M' \mid L'$, where

$$\begin{aligned} M' &= i[\![c!\langle v\rangle.A_i]\!] \mid o[\![c?(x).d!\langle h(f(x),g(x))\rangle.A_o]\!] \\ L' &= cl[\![d?\langle x\rangle]\!] \end{aligned}$$

Now it is possible to apply rule (R-STRUCT) to the reduction inferred in (2.2) to obtain $\Gamma_M \rhd M \mid L \rightarrowtriangle M' \mid L'$. In a similar way we can show that $\Gamma_M \rhd M' \mid L' \rightarrowtriangle M'' \mid L'$, where $M'' = i[\![A_i]\!] \mid o[\![d!\langle h(f(v),g(v))\rangle.A_o]\!]$. Finally, we also have $\Gamma_M \rhd M''|L' \rightarrowtriangle M|L_0$; here $L_0 = cl[\![\mathbf{0}]\!]$. Now it is easy to note that such a network has no possible reduction; we reached a deadlocked state.

Consider now the network $\Gamma_N \rhd N|L$, where $\Gamma_N \rhd N$ is defined in Example 2.1.2. For this network we have $\Gamma_N \rhd N|L \rightarrowtriangle \Gamma_N \rhd N'|L'$, where

$$N' = i[\![c!\langle v\rangle.A_i]\!] \mid s_1[\![c?(x).c_1!\langle f(x)\rangle.d?(x).A_1]\!] \mid s_2[\![c?(x).c_2!\langle g(x)\rangle.d?(x).A_1]\!] \mid o[\![B_o]\!]$$

For $\Gamma_N \rhd N'$, note that we have both $\Gamma_N \vdash i \rightarrow s_1$ and $\Gamma_N \vdash i \rightarrow s_2$; as node i can broadcast message v in N',

we have that $\Gamma_N \triangleright N'|L' \twoheadrightarrow N''|L'$, where

$$\begin{array}{rcl} N'' & = & i[\![A_i]\!] \mid s_1[\![c_1!\langle f(v)\rangle .d?(x).A_1]\!] \mid s_2[\![c_2!\langle g(v)\rangle .d?(x).A_2]\!] \\ & \mid & |o[\![(c_1?(x).c_2?(y).d!\langle h(x,y)\rangle .A_o) + (c_2?\langle y\rangle .c_1?\langle x\rangle .d!\langle h(x,y)\rangle .A_o)]\!] \end{array}$$

The network $\Gamma_N \triangleright N''|L'$ now has two possible reductions modulo structural equivalence. In fact

1. either node s_1 broadcasts value $f(v)$, or
2. node s_2 broadcasts value $g(v)$.

Both these derivations can be inferred by using rules (R-BCAST) and (R-STRUCT). In the first case we have $\Gamma_N \triangleright N''|L' \twoheadrightarrow N_1''|L'$, where

$$N_1'' = i[\![A_i]\!] \mid s_1[\![d?(x).A_1]\!] \mid s_2[\![c_2!\langle g(v)\rangle .d?(x).A_2]\!] \mid o[\![c_2?(y).d!\langle h(f(v),y)\rangle .A_o]\!]$$

while in the second one we have $\Gamma_N \triangleright N''|L' \twoheadrightarrow N_2''|L'$, where

$$N_2'' = i[\![A_i]\!] \mid s_1[\![c_1!\langle f(v)\rangle .d?(x).A_1]\!] \mid s_2[\![d?(x).A_2]\!] \mid o[\![c_1?(x).d!\langle h(x,g(v))\rangle .A_o]\!]$$

Both these networks have a single reduction; for $j = 1,2$, $\Gamma_N \triangleright N_j''|L' \twoheadrightarrow N'''|L'$; here

$$N''' = i[\![A_i]\!] \mid s_1[\![d?(x).A_1]\!] s_2[\![d?(x).A_2]\!] \mid d!\langle h(f(v),g(v))\rangle .A_o o.$$

Finally, as $\Gamma_N \vdash o \to s_1$, $\Gamma_N \vdash o \to s_2$ and $\Gamma_n \vdash o \to cl$, we have the reduction $\Gamma_N \triangleright N'''|L' \twoheadrightarrow N|L_0$. Again, this network is deadlocked, that is it has no possible reduction. □

2.3 Labelled Transition Semantics

As we pointed out in Section 2.2, reduction semantics for networks only models the interactions among nodes for which the code has been provided. No information about how these nodes interact with the external environment (that is, with nodes in the interface of the network they belong to) is given.

Another kind of operational semantics is the *labelled transition semantics*. This operational semantics provides a more general description of the behaviour of a network, as it models how internal nodes interact with both internal and external nodes. The labelled transition semantics strongly relies on the notion of *Labelled Transition System* (**LTS**) [50], whose definition is provided below. In a process calculus, terms are interpreted as states of an LTS, while the transitions that can be inferred in the labelled transition semantics coincide with the actions performed by the state of the LTS in which such a term is interpreted.

Definition 2.3.1 (Labelled Transition System)**.** An LTS is a triple $\langle S, Act, \longrightarrow\rangle$ where

1. S is a countable set of *states*,
2. Act is a countable set of *actions*
3. $\longrightarrow \subseteq S \times Act \times S$ is the transition relation.

Given an LTS $\langle S, Act, \longrightarrow\rangle$, two states $s, t \in S$ and an action $\lambda \in Act$, we use the more classical notation $s \xrightarrow{\lambda} t$ in lieu of $(s,\lambda,t) \in Act$.

The LTS in which networks are interpreted is given by $\langle \mathsf{Nets}, Act, \longrightarrow\rangle$, where $Act = \{m.c!v, m.c?v, m.\tau \mid m \in \mathrm{Nodes}, c \in \mathrm{Ch}, v \in \mathrm{Val}\}$. Henceforth we will use λ to range over elements of Act. The action $m.c!v$ corresponds to node m broadcasting value v along channel c, $m.c?v$ to a node (different from m) reacting to some input broadcast by m along channel c, and $m.\tau$ to some internal activity performed by node m. The transition relation

$$\text{(B-BROAD)}\quad \frac{P \xrightarrow{c!v} Q}{\Gamma \rhd n[\![P]\!] \xrightarrow{n.c!v} n[\![Q]\!]}$$

$$\text{(B-REC)}\quad \frac{P \xrightarrow{c?v} Q}{\Gamma \rhd n[\![P]\!] \xrightarrow{m.c?v} n[\![Q]\!]}\ \Gamma \vdash m \to n$$

$$\text{(B-DEAF)}\quad \frac{\neg \mathrm{rcv}(P,c)}{\Gamma \rhd n[\![P]\!] \xrightarrow{m.c?v} n[\![P]\!]}\ \Gamma \vdash m \to n$$

$$\text{(B-DISC)}\quad \frac{}{\Gamma \rhd n[\![P]\!] \xrightarrow{m.c?v} n[\![P]\!]}\ \begin{array}{l}\Gamma \vdash m \nrightarrow n \\ m \neq n\end{array}$$

$$\text{(B-}\mathbf{0}\text{)}\quad \frac{}{\mathbf{0} \xrightarrow{m.c?v} \mathbf{0}}$$

$$\text{(B-}\tau\text{)}\quad \frac{P \xrightarrow{\tau} Q}{n[\![P]\!] \xrightarrow{n.\tau} n[\![Q]\!]}$$

$$\text{(B-}\tau\text{.PROP}-\text{L)}\quad \frac{\Gamma \rhd M \xrightarrow{n.\tau} L}{\Gamma \rhd M \mid N \xrightarrow{n.\tau} L \mid M}$$

$$\text{(B-}\tau\text{.PROP}-\text{R)}\quad \frac{\Gamma \rhd N \xrightarrow{n.\tau} L}{\Gamma \rhd M \mid N \xrightarrow{n.\tau} M \mid L}$$

$$\text{(B-PROP)}\quad \frac{\Gamma \rhd M \xrightarrow{m.c?v} M',\ \Gamma \rhd N \xrightarrow{m.c?v} N'}{\Gamma \rhd M \mid N \xrightarrow{m.c?v} M' \mid N'}$$

$$\text{(B-SYNC}-\text{L)}\quad \frac{\Gamma \rhd M \xrightarrow{m.c!v} M',\ \Gamma \rhd N \xrightarrow{m.c?v} N'}{\Gamma \rhd M \mid N \xrightarrow{m.c!v} M' \mid N'}$$

$$\text{(B-SYNC}-\text{R)}\quad \frac{\Gamma \rhd M \xrightarrow{m.c?v} M',\ \Gamma \rhd N \xrightarrow{m.c!v} N'}{\Gamma \rhd M \mid N \xrightarrow{m.c!v} M' \mid N'}$$

Figure 2.4: Labelled Transition Semantics for (high level) networks

for networks is the smallest relation generated by the inference rules of Figure 2.4; here we use the shortcut notation $\Gamma \rhd M \xrightarrow{\lambda} N$ instead of $\Gamma \rhd M \xrightarrow{\lambda} \Gamma \rhd N$. These rules rely on a pre-semantics for processes, which is presented later in this Section.

Let us comment the rules of Figure 2.4. Rule (B-BROAD) models the behaviour of a node which is willing to broadcast some value v.

Rule (B-REC) says that, if a node n which is waiting to receive a value along channel c detects a transmission (along the same channel) from another node m, and n is in the transmission range of m, then it receives the value which has been broadcast by the latter correctly. On the other hand, if n is not waiting to receive a value along channel c, or if n is not in the sender's range of transmission, then it is not affected by the broadcast. These situations are modelled by rules (B-DEAF) and (B-DISC), respectively.

Note that the rules for receiving a value being broadcast along a channel c model the fact that a value can only be received by a node which is in the sender's range of transmission, and which is waiting to detect a value along channel c. Further, such nodes will always receive the value being broadcast, thus implementing our concept of reliable communication. While assuming that communication may not be reliable (that is, a node can always avoid to detect a message broadcast along a channel) would have led to a simpler operational semantics for networks, many protocols defined at the network layer of the *ISO/OSI reference model* [65] and in general protocols for distributed computing [2] rely on the assumption that communication primitives are reliable. Further, in our framework it is always possible to model an unreliable receiver by introducing a τ-prefix before a receiving primitive; that is, an unreliable receiver along channel c can be modelled with the code $\tau.c?(x).P$. A node running this code is not waiting to receive any value along channel c; however, it can always perform an internal activity (see Rule (B-τ)) to evolve to a perfect receiver.

Rule (B-τ) defines internal transitions for nodes, while rules (B-τ.PROP $-$ L) and (B-τ.PROP $-$ R) propagate internal actions through parallel components.

Rule (B-PROP) propagates input actions through parallel components; here the transition is inferred by the

(S-SND)
$$\frac{}{c!\langle e\rangle.P \xrightarrow{c!v} P}\ v = [\![e]\!]$$

(S-RCV)
$$\frac{}{c?(x).P \xrightarrow{c?v} \{v/x\}P}$$

(S-τ)
$$\frac{}{\tau.P \xrightarrow{\tau} P}$$

(S-SUM−L)
$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

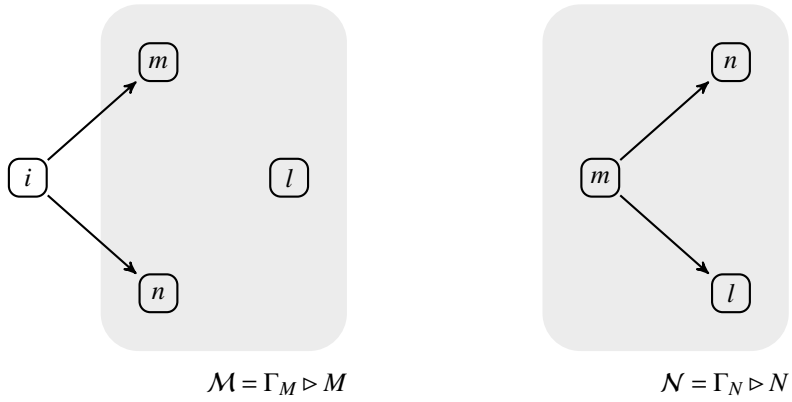
(S-SUM−R)
$$\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

(S-THEN)
$$\frac{P \xrightarrow{\alpha} P'}{\text{if } b \text{ then } P \text{ else } Q \xrightarrow{\alpha} P'}\ [\![b]\!] = \text{true}$$

(S-ELSE)
$$\frac{Q \xrightarrow{\alpha} Q'}{\text{if } b \text{ then } P \text{ else } Q \xrightarrow{\alpha} Q'}\ [\![b]\!] = \text{false}$$

(S-PDEF)
$$\frac{A(\tilde{x}) \Leftarrow P \qquad \{\tilde{v}/\tilde{x}\}P \xrightarrow{\alpha} Q}{A\langle\widetilde{e}\rangle \xrightarrow{\alpha} Q}\ \widetilde{[\![e]\!]} = \tilde{v}$$

Figure 2.5: Pre-semantics of states



$\mathcal{M} = \Gamma_M \rhd M$ $\qquad$ $\mathcal{N} = \Gamma_N \rhd N$

Figure 2.6: Two networks $\mathcal{M}$ and $\mathcal{N}$

input transitions performed by the two system terms which are composed.

Rules (B-SYNC − L) and (B-SYNC − R) model broadcast communication. Note that if a network $\Gamma \rhd M$ performs a broadcast action of the form $m.c!v$, while a second network $\Gamma \rhd N$ receives such a value by performing a $m.c?v$ action, the action performed by the overall network $\Gamma \rhd M \,|\, N$ still has the form $m.c!v$. Informally speaking, this means that the output action performed by node m is still available to other nodes, thus implementing the behaviour of broadcast communication. See [55] for a detailed discussion.

Let us now turn our attention to the pre-semantics for the code. Judgements take the form

$$P \xrightarrow{\mu} Q$$

where P is a closed state, that is containing no free occurrences of any variable and α ranges over $c!v$, $c?v$ or τ. The deductive rules for inferring these judgements are given in Figure 2.5 and should be self-explanatory.

Later we will need the following useful definitions, which are standard in concurrency theory [50, 58, 34]: we use the notation $\Gamma \rhd M \xrightarrow{\lambda}$ if there exists a system term M' such that $\Gamma \rhd M \xrightarrow{\lambda} M'$ and $\Gamma \rhd M \not\xrightarrow{\lambda}$ as the negation of $\Gamma \rhd M \xrightarrow{\lambda}$. Similar definitions apply to processes.

Let us look at some examples that show how transitions can be derived in the labelled transition semantics.

Example 2.3.2 (Input Actions)**.** Let $\mathcal{M}$ be the network depicted in Figure 2.6, where

$$M = m[\![c?(x).P]\!] \mid n[\![d?(x).Q]\!] \mid l[\![c?(x).P_1 + d?(x).P_2]\!]$$

This network has three internal nodes, m, n, and l. The first two of them can receive messages broadcast by an external node i, while the last cannot.

If we use the reduction semantics defined in Section 2.2 we find that no reduction is possible for this network; this is because none of the internal nodes can broadcast a value along a channel, or perform an internal action. On the other hand, in the labelled transition semantics this network can receive inputs from the external node i.

In fact, it is possible to show that, for any value v, $\Gamma_M \rhd M \xrightarrow{c?v} M_c^v$, where

$$M_c^v = m[\![\{v/x\}P]\!] \mid n[\![d?(x).Q]\!] \mid l[\![(c?(x).P_1) + d?(x).Q_1]\!]$$

In this example we show how it is possible to derive this transition, by using the inference rules defined in figures 2.4 and 2.5. To maintain the notation easier, let

$$\begin{aligned} P_n &= d?(x).Q \\ P_l &= c?(x).P_1 + c?(x).Q_1 \end{aligned}$$

Let us first focus on the code in node m. By an application of Rule (S-RCV) we have that $c?(x).P \xrightarrow{i.c?v} \{v/x\}P$, where v is an arbitrary value. As $\Gamma_M \vdash i \to m$, we can apply Rule (B-REC) to the last transition to infer $\Gamma_M \rhd m[\![c?(x).P]\!] \xrightarrow{i.c?v} m[\![\{v/x\}P]\!]$ for any value $v \in \mathrm{Val}$.

Now let us focus on the other two internal nodes, n and l. For node n, note that the predicate $\mathrm{rcv}(P_n, c)$ is false; thus, we can apply Rule (B-DEAF) to infer $\Gamma_M \rhd n[\![P_n]\!] \xrightarrow{i.c?v} n[\![P_n]\!]$ for any value v.

For node l the predicate $\mathrm{rcv}(P_l, c)$ turns out to be true. However, in contrast with the case of node m, Rule (B-REC) cannot be applied to the network $\Gamma_M \rhd l[\![P_l]\!]$, for the side condition $\Gamma_M \vdash i \to l$ does not hold. Instead, as $\Gamma_M \vdash i \nrightarrow l$, rule (B-DISC) can be applied; hence for any value v we can infer the transition $\Gamma_M \rhd l[\![P_l]\!] \xrightarrow{i.c?v} l[\![P_l]\!]$.

Finally, we put together the transitions derived for the three networks $\Gamma_M \rhd m[\![c?\langle x\rangle.P]\!]$, $\Gamma_M \rhd n[\![P_n]\!]$ and $\Gamma_M \rhd l[\![P_l]\!]$ to infer the transitions $\Gamma_M \rhd M \xrightarrow{i.c?v} \Gamma_M \rhd M_c^v$, where v is an arbitrary value. This can be done by a double application of Rule (B-PROP).

The proof of the Derivation $\Gamma_M \rhd M \xrightarrow{i.c?v} M_c^v$ is provided in Equation 2.3

$$\dfrac{\dfrac{\dfrac{}{c?\langle x\rangle.P \xrightarrow{c?v} \{v/x\}P}}{\Gamma_M \rhd m[\![c?\langle x\rangle.P]\!] \xrightarrow{i.c?v} m[\![\{v/x\}P]\!]}\,\Gamma_M \vdash i \to m \qquad \dfrac{\dfrac{\dfrac{\neg\mathrm{rcv}(P_n, c)}{}}{\Gamma_M \rhd n[\![P_n]\!] \xrightarrow{i.c?v} n[\![P_n]\!]} \qquad \dfrac{}{\Gamma_M \rhd l[\![P_l]\!] \xrightarrow{i.c?v} l[\![P_l]\!]}\,\Gamma_M \vdash i \nrightarrow l}{\Gamma_M \rhd n[\![P_n]\!] \mid l[\![P_l]\!] \xrightarrow{i.c?v} n[\![P_n]\!] \mid l[\![P_l]\!]}}{\Gamma_M \rhd M \xrightarrow{i.c?v} M_c^v} \tag{2.3}$$

□

Example 2.3.3 (Broadcast)**.** Consider the network $\Gamma_N \rhd N$ depicted in Figure 2.6 Here, let

$$N = m[\![c!\langle v\rangle]\!] \mid n[\![c?(x).P]\!] \mid l[\![c?(x).P]\!]$$

We show that this network can perform a broadcast action $\Gamma_N \rhd N \xrightarrow{m.c!v} N'$, where $N' = m[\![\mathbf{0}]\!] \mid n[\![\{v/x\}P]\!] \mid l[\![\{v/x\}Q]\!]$. That is, the broadcast of value v performed by node m affects both nodes n, l.

First, let us focus on the code run by node m in $\Gamma_N \rhd N$. By applying rule (S-SND), we infer the transition

$$c!\langle v\rangle \xrightarrow{c!v} \mathbf{0}$$

Now, it is sufficient to apply Rule (B-BROAD) to the transition above to obtain

$$\Gamma_N \rhd m[\![c!\langle v\rangle]\!] \xrightarrow{m.c!v} m[\![\mathbf{0}]\!] \tag{2.4}$$

Let us now turn our attention to node n. This node is running process $c?(x).P$; by an application of Rule (S-RCV) we have that

$$c?(x).P \xrightarrow{c?v} \{v/x\}P$$

Since $\Gamma_N \vdash m \to n$ we can apply this derivation to Rule (B-REC) to obtain infer

$$\Gamma_N \rhd n[\![c?(x).P]\!] \xrightarrow{m.c?v} n[\![\{v/x\}P]\!] \tag{2.5}$$

In the same way we derived the Transition (2.5) we can prove that

$$\Gamma_N \rhd l[\![c?(x).Q]\!] \xrightarrow{m.c?v} l[\![\{v/x\}Q]\!] \tag{2.6}$$

Now, we can apply Rule (B-SYNC – L) to transitions (2.4) and (2.5), thus obtaining

$$\Gamma_N \rhd m[\![c!\langle v\rangle]\!] \,|\, n[\![c?(x).P]\!] \xrightarrow{m.c!v} m[\![\mathbf{0}]\!] \,|\, n[\![\{v/x\}P]\!] \tag{2.7}$$

A final application of Rule (B-SYNC – L) to transitions (2.7) and (2.6) produces the desired transition $\Gamma_N \rhd N \xrightarrow{m.c!v} N'$.

The full proof of the transition $\Gamma_N \rhd N \xrightarrow{m.c!v} N'$ is given in Equation (2.8). To maintain the notation simpler, we define $N_n = n[\![c?(x).P]\!]$, $N'_n = n[\![\{v/x\}P]\!]$, $N_l = l[\![c?(x).Q]\!]$ and $N'_l = l[\![\{v/x\}Q]\!]$.

$$\dfrac{\dfrac{\dfrac{}{c!\langle v\rangle \xrightarrow{c!v} \mathbf{0}}\,[\![v]\!] = v}{\Gamma_N \rhd m[\![c!\langle v\rangle]\!] \xrightarrow{m.c!v} m[\![\mathbf{0}]\!]} \qquad \dfrac{\dfrac{}{c?(x).P \xrightarrow{c?v} \{v/x\}P}}{\Gamma_N \rhd N_n \xrightarrow{m.c?v} N'_n}\,\Gamma_n \vdash m \to n \qquad \dfrac{\dfrac{}{c?(x).Q \xrightarrow{c?v} \{v/x\}Q}}{\Gamma_N \rhd N_l \xrightarrow{m.c?v} N'_l}\,\Gamma_n \vdash m \to l}{\dfrac{\Gamma_N \rhd m[\![c!\langle v\rangle]\!] \,|\, N_n \xrightarrow{m.c!v} m[\![\mathbf{0}]\!] \,|\, N'_n}{\Gamma_N \rhd N \xrightarrow{m.c!v} N'}} \tag{2.8}$$

□

2.4 Properties of the Calculus

In this Section we prove several properties concerning the labelled transition semantics, defined in Section 2.3. We focus mainly on the structure which is required by a network $\Gamma \rhd M$ to perform a transition $\Gamma \rhd M \xrightarrow{\lambda} M'$, and how this transition affects the system term M (or, more specifically, how the syntactic structure of M' can be inferred from that of the system term M).

The properties we provide are very useful to prove that the labelled transition semantics coincides, up-to input actions and modulo structural congruence, with the reduction semantics defined in Section 2.2. This result is stated in Theorem 2.4.10 at the end of this Section.

The proofs of the results stated contain sometimes technical details, and are quite long to read. In order to make the contents of this Section more readable, we decided to include the details of the proofs in Appendix A, Section A.1.

Before proving any property about networks, we need some auxiliary lemmas concerning the pre-semantics for processes. The first property we prove for processes says that transitions are preserved by structurally congruent processes.

Lemma 2.4.1. Let P, Q be two processes such that $P \equiv Q$. Then, for any process P and action α, $P \xrightarrow{\alpha} R$ if and only if $Q \xrightarrow{\alpha} R$

Proof. See the Appendix. □

Let us now turn our attention to the structure of processes which can perform an internal action.

Lemma 2.4.2 (Internal actions for processes)**.** For any processes P, P', $P \xrightarrow{\tau} P'$ if and only if $P \equiv \tau.P' + Q$ for some process Q.

Proof. See the Appendix. □

In a similar way, we can derive the structure of processes which can perform a broadcast or an input action.

Lemma 2.4.3 (Broadcast actions for processes)**.** For all processes P, P' we have that $P \xrightarrow{c!v} P'$ iff there exist a process Q and an expression e such that $[\![e]\!] = v$ and $P \equiv c!\langle e\rangle.P' + Q$.

Proof. The proof is analogous to that of Lemma 2.4.2. The if implication is proved by inferring the transition $c!\langle e\rangle.P' + Q \xrightarrow{c!v} P'$, then by applying Lemma 2.4.1, while the only if implication is proved by Rule Induction on the last rule applied in the proof of the transition $P \xrightarrow{c!v} P'$. □

Lemma 2.4.4 (Input actions for processes)**.** For all processes P, P' we have that $P \xrightarrow{c?v} P'$ iff there exist P_1, Q such that $P \equiv c?(x).P_1 + Q$ and $P' \equiv \{v/x\}P_1$.

Proof. Analogous to those of Lemma 2.4.2 and Lemma 2.4.3 □

We now turn our attention to system terms. First, we want to show that the actions performed by a network are preserved by structurally equivalent system terms. In order to prove this result, we need the following Lemma, which deals with parallel components in system terms.

Lemma 2.4.5 (Parallel Components)**.** Whenever $\Gamma \rhd M_1 \mid M_2 \xrightarrow{\lambda} N$

(i) if $\lambda = m.c?v$, then there exist N_1, N_2 such that $\Gamma \rhd M_1 \xrightarrow{m.c?v} N_1$, $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$ and $N = N_1 \mid N_2$

(ii) if $\lambda = m.c!v$, then either

  (a) $\Gamma \rhd M_1 \xrightarrow{m.c!v} N_1$, $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$ for some N_1, N_2 such that $N = N_1 \mid N_2$, or

  (b) $\Gamma \rhd M_1 \xrightarrow{m.c?v} N_1$, $\Gamma \rhd M_2 \xrightarrow{m.c!v} N_2$ for some N_1, N_2 such that $N = N_1 \mid N_2$

(iii) if $\lambda = m.\tau$, then either

  (a) $\Gamma \rhd M_1 \xrightarrow{m.\tau} N_1$ for some N_1 such that $N = N_1 \mid M_2$, or

  (b) $\Gamma \rhd M_2 \xrightarrow{m.\tau} N_2$ for some N_2 such that $N = M_1 \mid N_2$

Proof. See the Appendix. □

We are now ready to show that transitions for networks are preserved by structural congruence.

Proposition 2.4.6. Suppose $\Gamma \rhd M \equiv \Gamma \rhd N$, and $\Gamma \rhd M \xrightarrow{\lambda} M'$. Then there exists $N' \equiv M'$ such that $\Gamma \rhd N \xrightarrow{\lambda} N'$.

Proof. See the Appendix. □

The last proposition is very useful if we want to analyse the structure that is required by a network to perform a given action.

Proposition 2.4.7 (Broadcast)**.** Let m be a node, c a channel and v a value. For any system networks $\Gamma \rhd M, \Gamma \rhd N$ we have that $\Gamma \rhd M \xrightarrow{m.c!v} \Gamma \rhd N$ iff $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \mid M'$ for some expression e, processes P, Q, system terms M', N' such that $[\![e]\!] = v$, $N \equiv m[\![P]\!] \mid N'$ and $\Gamma \rhd M' \xrightarrow{m.c?v} N'$.

Proof. See the Appendix. □

Proposition 2.4.8 (Internal actions)**.** Let $\Gamma \rhd M, \Gamma \rhd N$ be networks. Then $\Gamma \rhd M \xrightarrow{m.\tau} N$ iff $M \equiv m[\![\tau.P + Q]\!] \mid M'$ for some processes P, Q and M' such that $N \equiv m[\![P]\!] \mid M'$.

Proof. For the if implication it is easy to provide a derivation of $\Gamma \rhd m[\![\tau.P + Q]\!] \mid M' \xrightarrow{m.\tau} m[\![P]\!] \mid M'$. Then we can use Lemma 2.4.6 to show that whenever $M \equiv m[\![\tau.P + Q]\!] \mid M'$, then $\Gamma \rhd M \xrightarrow{m.\tau} N$, where $N \equiv m[\![P]\!] \mid M'$.

The only if implication is proved by Rule Induction on the proof of the transition $\Gamma \rhd M \xrightarrow{m.\tau} N$, using Lemma 2.4.2. □

The last kind of actions which can be performed by a network are input actions. These turn out to be the most difficult to analyse.

Proposition 2.4.9 (Input actions)**.** For any networks $\Gamma \rhd M, \Gamma \rhd N$, node m, channel c and value v, $\Gamma \rhd M \xrightarrow{m.c?v} N$ iff there exist a finite index set I, a collection of nodes $\{n_i\}_{i \in I}$, two collections of processes $\{P_i\}_{i \in I}$ and $\{Q_i\}_{i \in I}$, and two networks M_1 and M_2 such that:

(i) $M \equiv \prod_{i \in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2$

(ii) $m \notin \text{nodes}(M)$

(iii) for every $i \in I$, $\Gamma \vdash m \to n_i$

(iv) $\neg \text{rcv}(M, c)$

(v) for any $n \in \text{nodes}(M_2)$, $\Gamma \vdash m \nrightarrow n$

(vi) $N \equiv \prod_{i \in I} n_i[\![\{v/x\}P_i]\!] \mid M_1 \mid M_2$

Proof. See the Appendix. □

The results we proved until now allow us to reason about parallel composition of system terms; also, given a transition of the form $\Gamma \rhd M \xrightarrow{\lambda} N$, we are able to perform a case analysis on λ to infer the structure of both the system terms M and N. These properties are very useful to prove that the labelled transition semantics for our calculus is consistent with the reduction semantics. That is, we can show that the reduction relation $\twoheadrightarrow$ coincides, up-to structural congruence, with the union of the transition relations $\xrightarrow{m.\tau}$ and $\xrightarrow{m.c!v}$, quantified over all nodes m, channels c and values v. This is the main result of this Section; it is stated precisely and proved below.

Theorem 2.4.10 (Harmony Theorem)**.**

(i) Whenever $\Gamma \rhd M \twoheadrightarrow N$, then either

  (a) $\Gamma \rhd M \xrightarrow{m.\tau} N'$ for some m and N' such that $N' \equiv N$, or

  (b) $\Gamma \rhd M \xrightarrow{m.c!v} N'$ for some m, c, v and N' such that $N' \equiv N$

(ii) If $\Gamma \rhd M \xrightarrow{m.\tau} N$ for some node m, then $\Gamma \rhd M \twoheadrightarrow N$

(iii) If $\Gamma \rhd M \xrightarrow{m.c!v} N$ for some node m, channel c and value v, then $\Gamma \rhd M \twoheadrightarrow N$.

Proof. The three statements are proved separately.

(i) We perform a rule induction on the proof of $\Gamma \rhd M \twoheadrightarrow N$. If the last rule applied in such a proof is (R-BCAST), then

$$M = m[\![c!\langle e \rangle.P + Q]\!] \mid \prod_{i \in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2$$

for some expression e, node m, finite index set I, collection of processes $\{P_i\}_{i \in I}, \{Q_i\}_{i \in I}$, collection of nodes $\{n_i\}_{i \in I}$, system terms M_1, M_2 and value v such that

- $[\![e]\!] = v$,
- For any $i \in I$, $\Gamma \vdash m \to n_i$,
- $\neg \mathrm{rcv}(M_1, c)$,
- For any $n \in \mathrm{nodes}(M_2)$, $\Gamma \vdash m \twoheadrightarrow n_i$,
- $N = m[\![P]\!] \mid \prod_{i\in I} n_i[\![\{v/x\}P_i]\!] \mid M_1 \mid M_2$

By an application of Proposition 2.4.9 we obtain that

$$\Gamma \rhd \prod_{i\in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2 \xrightarrow{m.c?v} \prod_{i\in I} n_i[\![\{v/x\}P_i]\!] \mid M_1 \mid M_2$$

Finally, by applying Proposition 2.4.7 we can infer

$$\Gamma \rhd m[\![c!\langle e\rangle.P + Q]\!] \mid \prod_{i\in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2 \xrightarrow{m.c!v} m[\![P]\!] \mid \prod_{i\in I} n_i[\![\{v/x\}P_i]\!] \mid M_1 \mid M_2$$

If the last rule applied to infer $\Gamma \rhd M \twoheadrightarrow N$ is Rule (R-τ) then $M = m[\![\tau.P + Q]\!] \mid M_1$ for some node m, processes P, Q and system term M_1 such that $N = m[\![P]\!] \mid M_1$. Here we obtain

$$\Gamma \rhd m[\![\tau.P + Q]\!] \mid M_1 \xrightarrow{m.\tau} m[\![P]\!] \mid M_1$$

as a direct consequence of Lemma 2.4.8.

The last case to check is that in which the last rule applied in the proof of the reduction $\Gamma \rhd M \twoheadrightarrow N$ is Rule (R-STRUCT). In this case we have that there exist M', N' such that $M \equiv M', N \equiv N'$ and $\Gamma \rhd M' \twoheadrightarrow N'$. By inductive hypothesis, we have that $\Gamma \rhd M' \xrightarrow{\lambda} N''$ where λ has either the form $m.\tau$ or $m.c!v$, and N'' is a system term such that $N'' \equiv N'$. By transitivity of $\equiv$, we have that $N'' \equiv N$, so that we can apply Proposition 2.4.6 to show that $\Gamma \rhd M' \xrightarrow{\lambda} N'$.

(ii) Suppose $\Gamma \rhd M \xrightarrow{m.\tau} N$ for an arbitrary node m. It follows from Proposition 2.4.8 that $M \equiv m[\![\tau.P + Q]\!] \mid M'$ and $N \equiv m[\![P]\!] \mid M'$ for some processes P, Q and system term M'. Rule (R-τ) of the reduction semantics ensures that

$$\Gamma \rhd m[\![\tau.P + Q]\!] \mid M' \twoheadrightarrow m[\![P]\!] \mid M'$$

Then we can apply Rule (R-STRUCT) to the reduction above to infer $\Gamma \rhd M \twoheadrightarrow N$.

(iii) Suppose $\Gamma \rhd M \xrightarrow{m.c!v} N$, where m is a node, c a channel and v a value. By Proposition 2.4.7 we have that there exist an expression e, processes P, Q and a system term M' such that $[\![e]\!] = v$ and $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \mid M'$, with $\Gamma \rhd M' \xrightarrow{m.c?v} N'$ for some N' such that $N \equiv m[\![P]\!] \mid N'$.

We can now apply Proposition 2.4.9 to the transition $\Gamma \rhd M' \xrightarrow{m.c?v} N'$ to show that

$$M' \equiv \prod_{i\in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2$$

for some finite index set I, collections $\{P_i\}_{i\in I}$, $\{Q_i\}_{i\in I}$, $\{n_i\}_{i\in I}$ and system terms M_1, M_2 such that

(a) For all $i \in I$, $\Gamma \vdash m \to n_i$

(b) $\neg \mathrm{rcv}(M_1, c)$

(c) For all $n \in \mathrm{nodes}(M_2)$, $\Gamma \vdash m \twoheadrightarrow n$

(d) $N' \equiv \prod_{i\in I} n_i[\![\{v/x\}.P + i]\!] \mid M_1 \mid M_2$

Now it is not difficult to note that

$$\begin{aligned} M &\equiv m[\![c!\langle e\rangle.P + Q]\!] \mid \prod_{i\in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2 \\ N &\equiv m[\![P]\!] \mid \prod_{i\in I} n_i[\![\{v/x\}.P_i + Q_i]\!] \mid M_1 \mid M_2 \end{aligned}$$

Also, note that we can apply Rule (R-BCAST) to infer

$$\begin{gathered} \Gamma \rhd m[\![c!\langle e\rangle.P + Q]\!] \mid \textstyle\prod_{i\in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2 \\ \rightarrowtriangle \\ m[\![P]\!] \mid \textstyle\prod_{i\in I} n_i[\![\{v/x\}.P + i]\!] \mid M_1 \mid M_2 \end{gathered}$$

In fact, all the requirements which are needed to derive the reduction above have already been proved to be satisfied. Now a simple application of Rule (R-STRUCT) to the reduction above leads us to $\Gamma \rhd M \rightarrowtriangle N$.

□

2.5 Related Work

We end this chapter by comparing our calculus with others which can be found in the literature; we hope that this comparison can help the reader to better understand the philosophy which inspired our framework.

The use of connectivity graphs in our calculus has been inspired by [53]; in contrast with this work, in our calculus we assume that the network topology of a network is static, that is mobile networks are not considered. On the other hand, in the paper the authors define the transition relation of the code run by a network to be parametric in a connectivity graph. One of the main advantages of this approach is that of using different connectivity graphs for different reductions in a computation, thus implementing the concept of mobile networks. As we have already pointed out, we are not interested in modelling mobile networks, hence we considered transitions parametrised in a connectivity graph as superfluous in our framework.

Below we discuss other alternatives that have been proposed in the literature to model wireless networks formally. In [48, 43] metric spaces are used to define the topology of a network; locations running code are equipped with a radius of transmission, and a distance function between locations is assumed to determine whether a location is in another's range of transmission. We believe that, despite the use of metric spaces allows a very intuitive definition of wireless networks, the analysis of wireless networks is rather complicated in this framework; further, developing compositional theories in such a framework seems to be a hard task to accomplish. We will discuss this topic more in detail in Chapter 3, for compositionality in wireless networks is discussed in such a chapter.

In [27, 28] the authors view a network as a collection of processes, each of which is associated with an address; in the operational semantics transitions (and, more specifically, broadcasts) are parametrised in a set of addresses which denote the processes which are influenced by the activity being performed. One of the main advantages of this calculus is that of encoding the topology of a network in the syntax of the calculus. This leads to the possibility of defining a wide range of operators for wireless networks, which have not been introduced in our calculus; examples include non-deterministic choice and prefixing of networks (note that these operations, in our framework, have been defined only at the process level).

In [47] a network is described as a collection of nodes running processes; each node has a semantic tag associated, which denoted the neighbourhood of the former. In this calculus the topology of a network is again embedded inside the syntax of the calculus, rather than relying on an additional mathematical structure. However, we point out that a connectivity graph can be built by a set of nodes equipped with their semantic tags, and vice versa.

Finally, in [62] the authors define a network as a collection of processes, each of which belongs to one or more groups. Processes having at least one group in common are considered to be in each other's range of transmission. This approach has the main advantage of abstracting from the concept of node location. However, in our own point of view, this calculus can be used to model networks at the *Transport Layer* of the *ISO/OSI Reference Model*; groups can be related to sub-networks [65], while the broadcast primitive corresponds to broadcast routing along one or more sub-nets. In our framework, we decided to model networks at the *Network Layer*, thus allowing the design of networks at a lower level.

Chapter 3

Behavioural Theories for Networks

In this Chapter we develop *behavioural theories* for wireless networks. Our aim is to determine whether two arbitrary networks, possibly with different connectivity graphs, exhibit the same behaviour.

The notion of behaviour of a network has to be stated precisely. Many of the works in Concurrency Theory focus on *observational theories*; that is, the notion of equivalence between two system depends on the *observations* that an external agent can perform on a system [17, 49]. Here much depends on the power that the external agent has over the system being observed. If this notion changes, the corresponding theory changes as well. Many different notions of behavioural equivalences and behavioural preorders have been investigated; see [67, 68] for a detailed discussion.

The most desirable property that we require from a behavioural theory is that of being *compositional*. In few words, this means that two equivalent networks can be interchanged each other in a larger network, without affecting the (observational) behaviour of the latter. Defining compositional theories in our framework requires that we state precisely how networks are composed together. As we will see, there are many possibilities to address this topic, each of which induce a different behavioural theory. As we have specific requirements for our behavioural theories to be met, we choose our notion of network composition according to them. We remark that compositional theories can be developed only if we restrict to a specific class of networks, which take the name of *composable networks*.

The behavioural theories we decided to develop for networks are those based on *may* and *must* testing, introduced by Hennessy and de Nicola [17]. This choice has been made for two reasons; first, to the best of our knowledge there is no work in the literature which addresses the problem of developing testing preorders for wireless networks. The second reason consists of the possibility of using the testing preorders to develop refinement techniques for networks; informally speaking, we can see a (possibly complicated) network as the refinement of another if we can show that the latter is testing related to the former.

Intuitively, in the testing framework a network $\mathcal{M}$ interacts with another one $\mathcal{T}$; the network $\mathcal{T}$ is called a *test* (or *testing network*). Through the interaction between $\mathcal{M}$ and $\mathcal{T}$, the testing network checks whether $\mathcal{M}$ satisfies a desired property, for which $\mathcal{T}$ was designed to check. In order to report success, the test $\mathcal{T}$ reaches a state in which the process ω is enabled.

Due to the non-deterministic nature of the calculus, this approach leads to two possible notions of whether a network passes a test; one possibility is to say that $\mathcal{M}$ **may-pass** $\mathcal{T}$, if the network resulting from the interaction between $\mathcal{M}$ and $\mathcal{T}$ has at least a sequence of reductions in which a configuration where the process ω is enabled can be reached. The other possibility is to say that the network $\mathcal{M}$ **must-pass** $\mathcal{T}$, if all the maximal sequences of reductions rooted in the network obtained by composing $\mathcal{M}$ with $\mathcal{T}$ have a configuration in which the process ω is enabled. Based on these notions of passing a test, two different preorders are induced. Both of them compare networks in terms of the tests that they pass, but they differ in the notion of passing a test which is used.

This Chapter is organised as follows: In Section 3.1 we first give a general definition for network composition. We show that, if we want to enable compositional reasoning, we need to restrict our attention to a specific

class of networks, which we call composable. We give two different examples of composition operators; a symmetric one $\|$, and an asymmetric one $\between$.

In Section 3.2 we study the algebraic structure of the set of composable networks, when equipped with the operator $\between$ defined in Section 3.1. We show that this algebraic structure is a partial monoid up-to structural equivalence. Further, we exhibit a set of generators for it; this result allows us to define a principle of induction for composable networks.

In Section 3.3 we first give a generic definition of the testing preorders; this definition is parametric in a composition operator $|||$. Then we establish the requirements that we want our testing preorders to meet. Finally, we show that the largest composition operator for which the induced introduced in Section 3.1.

We end the Chapter by providing a comparison between our approach to compositionality for networks and other works that consider compositional theories for wireless systems; this is done in Section 3.4.

3.1 Composing Networks

In Section 2.1 we have developed a simple calculus for wireless networks. In this calculus, the syntax for system terms is equipped with a composition operator. Given two system terms M and N, their composition $M \,|\, N$ is defined, provided that $\mathrm{nodes}(M) \cap \mathrm{nodes}(N) = \emptyset$.

Note that this composition operator is defined at the level of system terms, and it has several limitations. Given a network $\Gamma \rhd M$ and a system term N, we can define the network $\Gamma \rhd M \,|\, N$, provided that the latter is well-formed. By using this approach, it is evident that the only use of the operator $(\cdot \,|\, \cdot)$ is that of filling code in nodes which were external in the original network $\Gamma \rhd M$. However, it is not possible to merge two different networks $\Gamma_M \rhd M$ and $\Gamma_N \rhd N$ together.

In this Section, we study the problem of composing networks. In general, the way two networks should be composed together is rather intuitive; given two networks $\Gamma_M \rhd M$ and $\Gamma_N \rhd N$, we want to define a new network $\Gamma \rhd M \,|\, N$, where the connectivity graph Γ is obtained by merging Γ_M and Γ_N together.

This leads to a generic definition of network composition.

Definition 3.1.1 (Generic network composition)**.** Let $\Gamma_M \rhd M$, $\Gamma_N \rhd N$ be two networks; let also $\mathscr{P} : \mathsf{Nets} \times \mathsf{Nets} \to \{\mathsf{true}, \mathsf{false}\}$ be a predicate over pairs of networks. The composition of $\Gamma_M \rhd M$ and $\Gamma_N \rhd N$ with respect to the predicate $\mathscr{P}$ is defined as

$$(\Gamma_M \rhd M \,|||_{\mathscr{P}}\, \Gamma_N \rhd N) = \begin{cases} (\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) & \text{if } \mathscr{P}(\Gamma_M \rhd M, \Gamma_N \rhd N) = \mathsf{true} \\ \text{undefined} & \text{otherwise} \end{cases}$$

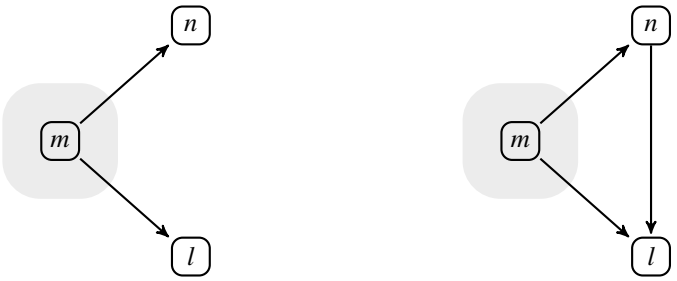
Here $\Gamma_M \cup \Gamma_N$ is defined by letting

$$\begin{aligned} (\Gamma_M \cup \Gamma_N)_V &= (\Gamma_M)_V \cup (\Gamma_N)_V \\ (\Gamma_M \cup \Gamma_N)_E &= (\Gamma_M)_E \cup (\Gamma_N)_E \end{aligned}$$

Definition 3.1.1 states that network composition is a partial operator. A *consistency predicate* $\mathscr{P}$ is used to establish whether the composition between two networks is defined. This is needed, for example, to ensure that well-formedness of networks is preserved by a composition operator $|||_{\mathscr{P}}$. This can be done by requiring that, for an arbitrary predicate $\mathscr{P}$ and networks $\mathcal{M}$, $\mathcal{N}$, if $\mathrm{nodes}(\mathcal{M}) \cap \mathrm{nodes}(\mathcal{N}) \neq \emptyset$ then $\mathscr{P}(\mathcal{M}, \mathcal{N}) = \mathsf{false}$.

In practice we will focus on weaker predicates; that is, whenever $\mathcal{M}$ and $\mathcal{N}$ are two networks such that $\mathrm{nodes}(\mathcal{M}) \cap \mathrm{nodes}(\mathcal{N}) = \emptyset$, it is not ensured that $\mathscr{P}(\mathcal{M}, \mathcal{N})$ is defined to be true.

Before presenting some particular instances of composition operator, let us recall that we want to enable compositional reasoning over networks. Let $|||_{\mathscr{P}}$ be a composition operator and $\leq$ be a preorder between networks; we say that $\leq$ is compositional if, for any $\mathcal{M}, \mathcal{N}$ and $\mathcal{T}$ such that $\mathcal{M} \leq \mathcal{N}$ and $\mathcal{M} |||_{\mathscr{P}} \mathcal{T}$, $\mathcal{N} ||| \mathcal{T}$ are both defined, then $\mathcal{M} |||_{\mathscr{P}} \mathcal{T} \leq \mathcal{N} |||_{\mathscr{P}} \mathcal{T}$. If $\leq$ is a preorder that compares networks in terms of their behaviour,



$\mathcal{M} = \Gamma_1 \rhd M$ $\mathcal{N} = \Gamma_2 \rhd M$

Figure 3.1: Two behaviourally equivalent networks

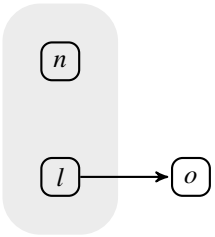
this means that $\mathcal{N}$ can be used to replace $\mathcal{M}$ in a larger network, without affecting the overall behaviour of the composed network $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$.

The notion of compositional reasoning is parametric the definition of the composition operator $|||_{\mathscr{P}}$ and the preorder $\leq$. The first has been defined only in a general way, while the latter has not been defined at all. However, we can already provide an illuminating example showing that, at least intuitively, we need to place some limitations to the topological structure of networks in order to enable compositional reasoning.

Example 3.1.2. Let $\mathcal{M}, \mathcal{N}$ be the two networks depicted in Figure 3.1; here $M = m[\![c!\langle v\rangle]\!]$. At least intuitively, these two networks exhibit the same behaviour. In fact in $\mathcal{M}$ node m broadcasts a message v along channel c to nodes n, l. The same happens in network $\mathcal{N}$. However, consider now the network $\mathcal{T} = \Gamma_T \rhd T$, where

$$T = n[\![c?(x).c!\langle v\rangle.\mathbf{0}]\!] \,|\, l[\![c?(x).c?(y).c!\langle v\rangle]\!]$$

and Γ_T is the connectivity graph depicted below



Finally, suppose that $|||_{\mathscr{P}}$ is a composition operator such that both $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ and $\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{T}$ are defined. In a compositional setting, these two networks would exhibit the same behaviour. However, this is not the case; in fact, in $\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{T}$, it is possible to reach a state in which node l can broadcast value v along channel c to node o, while this is not possible in network $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$.

We will revisit this example again in Example 3.3.17 and Example 4.3.27. □

The main problem in Example 3.1.2 lies in the presence of the connection from node n to node l, which is present in Γ_2, but not in Γ_1. This difference does not affect the behaviour of the two networks $\mathcal{M}$ and $\mathcal{N}$ of the Example above, but it plays a significant role in $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ and $\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{T}$.

A possible solution to the problem presented in Example 3.1.2 is that of focusing on a specific class of networks; specifically, we restrict our setting to networks in which external nodes cannot be connected to each other. This approach does not influence the expressive power of networks, as connections between external nodes in a network do not affect the rules of the labelled transition semantics defined in Section 2.3.

Definition 3.1.3 (Composable Networks)**.** A network $\mathcal{M} = \Gamma_M \rhd M$ is *composable* if and only if

1. $\mathcal{M}$ is well-formed
2. Whenever $\Gamma_M \vdash m \to n$, either $m \in \mathrm{nodes}(M)$ or $n \in \mathrm{nodes}(M)$

3. Whenever $m \in \mathsf{Int}(\mathcal{M})$ there exists a node $n \in \mathrm{nodes}(M)$ such that $\Gamma_M \vdash m \rightleftarrows n$.

The set of all composable networks is denoted by CNets.

A network is composable if and only if there are no external nodes which are directly connected one to another; further, any external node should be connected to at least an internal node. While the last constraint is not strictly necessary for our purposes, it can be used to help the proofs of some statements. However, all such statements remain valid in the case that constraint (3) is dropped. Henceforth we assume that a network $\mathcal{M}$ is composable, unless otherwise stated.

Remark 3.1.4. The reader could argue why we did not include the constraints required by composable networks in the definition of well-formed networks (Definition 2.1.1). To this end, note that the labelled transition semantics allows us to infer a transition for a composable network of the form $\Gamma \rhd M \,|\, N$ from those performed by its components $\Gamma \rhd M$ and $\Gamma \rhd N$. These two networks, however, are not composable.

Remark 3.1.5. The reader could also argue that our decision of focusing on composable networks is needed for purely technical reasons. However, this is not the case.

One could think of a network $\Gamma \rhd M$ as a mathematical representation of a (broadcast) distributed system. When designing such a system, one has to make some assumptions about the nodes which can communicate with the external environment, or equivalently to equip the distributed system with terminals which can be used by end users of the system to interact with it; these are exactly the interface nodes.

On the other hand, the designer of the distributed system should not assume any further knowledge of the external environment but the interface nodes (and their connections to internal nodes). In fact, it is in the power of the end user to take control of different terminals (interface nodes), possibly connecting them via another network whose nodes are hidden to the distributed system. This philosophy is reflected in our definition of composable network.

Later in this chapter we will present a composition operator for composable networks which implements the idea that users of the distributed system have control over the topology of the external environment.

Let $\mathcal{M} = \Gamma_M \rhd M$ be a network. We define $\mathsf{Input}(\mathcal{M}) = \{m \in \mathsf{Int}(\mathcal{M}) \mid \Gamma_M \vdash m \rightarrow n\}$ and $\mathsf{Output}(\mathcal{M}) = \{m \in \mathsf{Int}(\mathcal{M}) \mid \Gamma_M \vdash m \leftarrow n\}$. If $m \in \mathsf{Input}(\mathcal{M})$ we say that it is an *input node*, while if it is included in $\mathsf{Output}(\mathcal{M})$ we say that it is an *output node*. Finally, if $m \in \mathsf{Input}(\mathcal{M}) \cap \mathsf{Output}(\mathcal{M})$, we say that it is *fully connected*.

Note that the sets $\mathsf{Input}(\mathcal{M})$ and $\mathsf{Output}(\mathcal{N})$ are not necessarily disjoint. For example, if $\mathcal{M} = \Gamma \rhd M$ and $\Gamma \vdash m \leftrightarrow n$ for some node $m \in \mathsf{Int}(\mathcal{M})$, then $n \in \mathsf{Input}(\mathcal{M}) \cap \mathsf{Output}(\mathcal{N})$. Also, note that the definition of Input and Output interface relies only on those connectivities which connect an external node with an internal one. This is because the definition of both $\mathsf{Input}(\mathcal{M}), \mathsf{Output}(\mathcal{N})$ is given so that only external nodes can be included in such sets, and such nodes can be connected only to internal nodes by the definition of composable networks.

It is easy to check that, for any network $\mathcal{M}$,
$\mathsf{Int}(\mathcal{M}) = \mathsf{Input}(\mathcal{M}) \cup \mathsf{Output}(\mathcal{M})$. Note that, for any (well-formed) network $\Gamma_M \rhd M$,
we have $(\Gamma_M)_V = \mathrm{nodes}(M) \cup \mathsf{Int}(\Gamma_M \rhd M)$ and $\mathrm{nodes}(M) \cap \mathsf{Int}(\Gamma_M \rhd M) = \emptyset$.

Henceforth we will use the more graphic notation $\Gamma \vdash m \rightarrow M$ for $m \in \mathsf{Input}(\Gamma \rhd M)$, $\Gamma \vdash M \rightarrow m$ for $m \in \mathsf{Output}(\Gamma \rhd M)$ and $\Gamma \vdash m \rightleftarrows M$ for $m \in \mathsf{Input}(\Gamma \rhd M) \cap \mathsf{Output}(\Gamma \rhd M)$.

We conclude this Section by providing two instances of composition operators.

Definition 3.1.6 (Symmetric Composition)**.** Let $\mathscr{P}_s : \mathsf{CNets} \times \mathsf{CNets} \rightarrow \{\mathsf{true}, \mathsf{false}\}$ be the consistency predicate such that $\mathscr{P}(\Gamma_M \rhd M, \Gamma_N \rhd N) = \mathsf{true}$ if and only if

1. $\mathrm{nodes}(M) \cap \mathrm{nodes}(N) = \emptyset$
2. for any $m \in \mathrm{nodes}(M)$ $\Gamma_N \vdash m \rightarrow n$ ($\Gamma_N \vdash m \leftarrow n$) implies $\Gamma_M \vdash m \rightarrow n$ ($\Gamma_M \vdash m \leftarrow n$)
3. for any $n \in \mathrm{nodes}(N)$ $\Gamma_M \vdash m \rightarrow n$ ($\Gamma_M \vdash m \leftarrow n$) implies $\Gamma_N \vdash m \rightarrow n$ ($\Gamma_N \vdash m \leftarrow n$)

We us the symbol $\|$ to be the composition operator $|||_{\mathscr{P}_s}$.

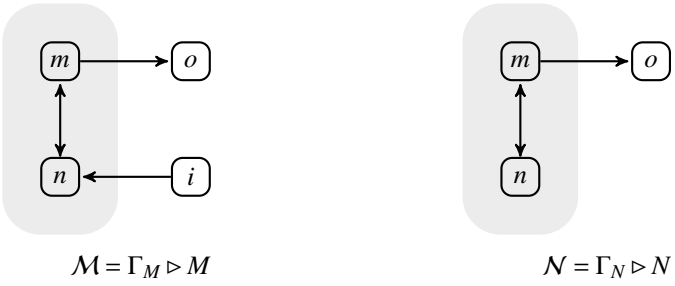


Figure 3.2: Two networks differing in their inferface

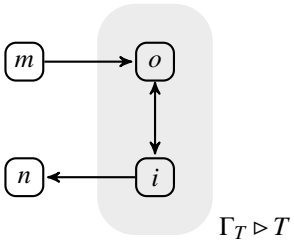


Figure 3.3: A network $\mathcal{T}$ which can be composed with $\mathcal{M}$ in Figure 3.2

The main idea in the definition of the composition operator $\|$ is that of allowing two the networks $\mathcal{M}$ and $\mathcal{N}$ to be composed together only whenever the information of their connectivity graphs is preserved. That is, the connections of internal nodes in $\mathcal{M}$ which appear in the composed network $\mathcal{M} \| \mathcal{N}$ have to be present in the network $\mathcal{M}$ itself; the same applies to network $\mathcal{N}$. Example 3.1.7 shows a simple application of this operator.

Example 3.1.7 (Symmetric Composition)**.** Let $\mathcal{M}, \mathcal{N}$ be the networks depicted in Figure 3.2. Here $M = m[\![P_m]\!] \mid n[\![P_n]\!]$ for some P_m, P_n. Also, $N = m[\![Q_m]\!] \mid n[\![Q_n]\!]$ for some Q_m, Q_n. Consider the network $\mathcal{T} = \Gamma_T \rhd T$ depicted in Figure 3.3. It is trivial to note that $\mathcal{M} \| \mathcal{T}$ is defined. In fact, we just need to note that the connection $\Gamma_M \vdash m \to o$ is defined in Γ_T and vice versa, The same applies to the connection $n \leftarrow i$, which is defined both in Γ_M and Γ_T. Here we have $\mathcal{M} \| \mathcal{T} = (\Gamma_M \cup \Gamma_T) \rhd (M \mid T)$, where the connectivity graph $(\Gamma_M \cup \Gamma_T)$ is depicted in Figure 3.4.

Now, consider the network $\mathcal{N}$. It is immediate to note that $\mathcal{N} \| \mathcal{T}$ is not defined. In fact, we have that $\Gamma_T \vdash n \leftarrow i$, where $n \in \text{nodes}(N)$. However, we do not have $\Gamma_N \vdash n \leftarrow i$, so that $\mathscr{P}_s(\mathcal{N}, \mathcal{T}) = \mathsf{false}$. In other words, if the composition $\mathcal{N} \| \mathcal{T}$ were defined, then it would have a connection from node n to node i, which is not present in network $\mathcal{N}$. □

The name *symmetric composition* acquires meaning only in contrast with another composition operator, which is defined below.

Definition 3.1.8 (Network Extension)**.** Let $\mathscr{P}_e : \mathsf{CNets} \times \mathsf{CNets} \to \{\mathsf{true}, \mathsf{false}\}$ be the consistency predicate such that $\mathscr{P}_e(\Gamma_M \rhd M, \Gamma_N \rhd N) = \mathsf{true}$ if and only if $(\Gamma_N)_V \cap \text{nodes}(M) = \emptyset$. The *Network Extension Operator* ,

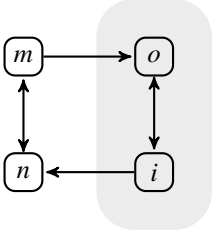


Figure 3.4: The result of the composition $\mathcal{M} \| \mathcal{T}$.

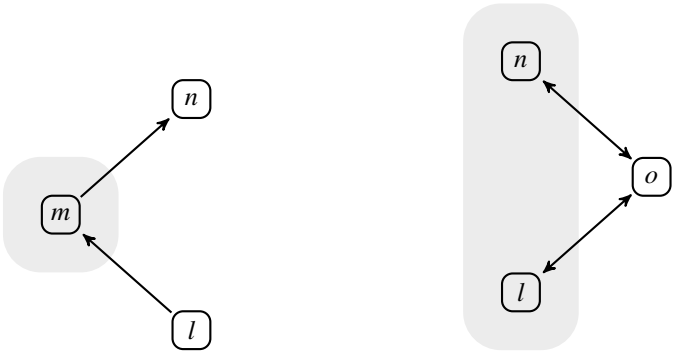
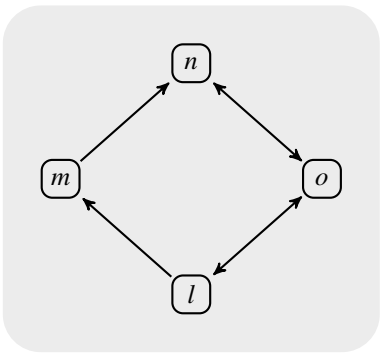


Figure 3.5: Network extension

denoted as $\between$, corresponds to the predicate $|||_{\mathscr{P}_e}$.

The operator $\between$, in contrast with $\|$, is not symmetric; in few words $\mathcal{M} \between \mathcal{N}$ is defined whenever the latter network can be used to extend the former one. This amounts to require that no node in $\mathcal{N}$, external or internal, is also an internal node in Γ_M. From the point of view of connectivities, this coincides with stating that the connections of $\mathcal{N}$ do not affect the behaviour of the internal nodes in $\mathcal{M}$ in the extended network $\mathcal{M} \between \mathcal{N}$. Example 3.1.9 provides a simple use of this operator.

Example 3.1.9 (Network Extension)**.** Let $\mathcal{M}, \mathcal{N}$ be the two networks defined in Figure 3.5. Here $M = m[\![P_m]\!]$ for some P_m, while $N = n[\![P_n]\!] \,|\, l[\![P_l]\!]$ for some P_n, P_l. It is easy to show that $\mathcal{M} \between \mathcal{N}$ is defined. In fact, it is sufficient to note that $\Gamma_N \nvdash m$. The network $\mathcal{M} \between \mathcal{N}$ is defined as $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N)$, where the connectivity graph $(\Gamma_M \cup \Gamma_N)$ is depicted below.



However, the network $\mathcal{N} \between \mathcal{M}$ is not defined. This is because $\Gamma_M \vdash n$ and $n \in \text{nodes}(N)$.

In other words, the network $\mathcal{N}$ can be used to extend network $\mathcal{M}$, as the connectivity of the internal nodes in $\mathcal{M}$ is not changed by $\mathcal{N}$ when these two networks are composed together; in contrast, the connectivity of the internal nodes in $\mathcal{N}$ is affected by the connectivity graph Γ_M when the composition of these two networks is considered; therefore, the extension $\mathcal{N} \between \mathcal{M}$ is not defined. □

Remark 3.1.10. Let us think again of a network $\mathcal{M}$ as the mathematical representation of a (broadcast) distributed system. We have already pointed out in Remark 3.1.5 that interface nodes can be viewed as the terminals provided to end users to access such a system. However, we mentioned that the distributed system has no further knowledge of the topology of the external environment, for its control has been granted to the end user.

The extension operator $\between$ reflects this philosophy. The way in which the end user interacts with a network $\mathcal{M}$ consists of another network $\mathcal{N}$, whose network topology is unspecified but for the fact that it contains no internal nodes of $\mathcal{M}$. Then, the interaction between the network $\mathcal{M}$ and the end user can be formalised as the network extension $\mathcal{M} \between \mathcal{N}$.

Despite being asymmetric, the extension operator $\sharp$ can be used to enable compositional reasoning over (composable) networks. This is the topic of Section 3.3.

3.2 Algebraic Properties of Composable Networks

In this Section we analyse the algebraic structure of CNets, when equipped with the extension operator $\sharp$.

The first property we prove concerns input nodes and output nodes of networks.

Lemma 3.2.1. Let $\mathcal{M}$, $\mathcal{N}$ be two composable networks such that $\mathcal{M} \sharp \mathcal{N}$ is defined. Then

(i) $\mathsf{Input}(\mathcal{M} \sharp \mathcal{N}) = (\mathsf{Input}(\mathcal{M}) \cup \mathsf{Input}(\mathcal{N})) \setminus \mathrm{nodes}(\mathcal{N})$

(ii) $\mathsf{Output}(\mathcal{M} \sharp \mathcal{N}) = (\mathsf{Output}(\mathcal{N}) \cup \mathsf{Output}(\mathcal{N})) \setminus \mathrm{nodes}(\mathcal{N})$

(iii) $\mathsf{Int}(\mathcal{M} \sharp \mathcal{N}) = (\mathsf{Int}(\mathcal{M}) \cup \mathsf{Int}(\mathcal{N})) \setminus \mathrm{nodes}(\mathcal{N})$

Proof. See Appendix A, Section A.2. □

Next we show that CNets is a closed set with respect to the operator $\sharp$.

Proposition 3.2.2 (Preservation of composable networks)**.** Let $\mathcal{M}, \mathcal{N} \in \mathsf{CNets}$; if $\mathcal{M} \sharp \mathcal{N}$ is defined, then $\mathcal{M} \sharp \mathcal{N} \in \mathsf{CNets}$.

Proof. Let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$, and suppose $P_e(\mathcal{M}, \mathcal{N}) = \text{true}$. We need to show that $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N)$ satisfies the requirements of Definition 3.1.3.

1. $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N)$ is well formed.

   We first check that $\mathrm{nodes}(M \,|\, N) \subseteq (\Gamma_M \cup \Gamma_N)_V$, then we prove that $M \,|\, N \in \mathsf{sSys}$; finally, we show that $(\Gamma_M \cup \Gamma_N)$ does not contain any self-loop.

   (a) Since $\mathcal{M}, \mathcal{N}$ are composable, hence well-formed, we have that $\mathrm{nodes}(M) \subseteq (\Gamma_M)_V$ and $\mathrm{nodes}(N) \subseteq (\Gamma_N)_V$. Thus,

   $$\mathrm{nodes}(M \,|\, N) = \mathrm{nodes}(M) \cup \mathrm{nodes}(N) \subseteq (\Gamma_M)_V \cup (\Gamma_N)_V = (\Gamma_M \cup \Gamma_N)_V$$

   (b) By definition of $\mathscr{P}_e$ we have that $\mathrm{nodes}(M) \cap (\Gamma_N)_V = \emptyset$. Since $\mathrm{nodes}(N) \subseteq (\Gamma_N)_V$, we also have $\mathrm{nodes}(M) \cap \mathrm{nodes}(N) = \emptyset$. Since $M \in \mathsf{sSys}, N \in \mathsf{sSys}$, it follows that $M \,|\, N \in \mathsf{sSys}$.

   (c) Suppose $(\Gamma_M \cup \Gamma_N) \vdash m \to n$. We need to shown that $m \neq n$. It is easy to show that either $\Gamma_M \vdash m \to n$ or $\Gamma_N \vdash m \to n$; we only consider the first case. Since $\mathcal{M}$ is a well-formed network, it follows that $m \neq n$.

2. for any m, n such that $\Gamma_M \cup \Gamma_N \vdash m \to n$, either $m \in \mathrm{nodes}(M \,|\, N)$ or $m \in \mathrm{nodes}(|N)$.

   Let m, n be two nodes for which $(\Gamma_M \cup \Gamma_N) \vdash m \to n$, Then either $\Gamma_M \vdash m \to n$ or $\Gamma_N \vdash m \to n$.

   Without loss of generality, let $\Gamma_M \vdash m \to n$. Since $\mathcal{M} \in \mathsf{CNets}$, we have that either $m \in \mathrm{nodes}(M)$ or $n \in \mathrm{nodes}(N)$ If $m \in \mathrm{nodes}(M)$ then $m \in \mathrm{nodes}(M \,|\, N)$, while if $n \in \mathrm{nodes}(M)$, then $n \in \mathrm{nodes}(M \,|\, N)$. Thus, either $m \in \mathrm{nodes}(M \,|\, N)$ or $n \in \mathrm{nodes}(M \,|\, N)$.

3. for any $m \in \mathsf{Int}(\mathcal{M} \sharp \mathcal{N})$ there exists $n \in \mathrm{nodes}(M \,|\, N)$ such that $(\Gamma_M \cup \Gamma_N) \vdash m \to n$ or $(\Gamma_M \cup \Gamma_N) \vdash m \leftarrow n$.

   Let $m \in \mathsf{Int}(\mathcal{M} \,\|\, \mathcal{N})$. Then $m \in (\Gamma_M \cup \Gamma_N)_V$ and $m \notin \mathrm{nodes}(M \,|\, N)$. In particular, either $m \in (\Gamma_M)_V$ and $m \notin \mathrm{nodes}(M)$ or $m \in (\Gamma_N)_V$ and $m \notin \mathrm{nodes}(N)$, which is equivalent to state that either $m \in \mathsf{Int}(\mathcal{M})$ or $m \in \mathsf{Int}(\mathcal{N})$. We only consider the former case; the proof for the latter one in analogous. Let $m \in \mathsf{Int}(\mathcal{M})$; since $\mathcal{M}$ is composable, there exists $n \in \mathrm{nodes}(M)$ such that either $\Gamma_M \vdash m \to n$ or $\Gamma_M \vdash m \leftarrow n$. That is, either $(\Gamma_M \cup \Gamma_N) \vdash m \to n$ or $(\Gamma_M \cup \Gamma_N) \vdash m \leftarrow n$, with $n \in \mathrm{nodes}(M \,|\, N)$.

□

Proposition 3.2.2 establishes that the mathematical structure $\langle \mathsf{CNets}, \sharp \rangle$ is a closed algebra. Note that the only properties of $\sharp$ we needed to perform this proof are the requirement that the sets nodes(M) and nodes(N) are disjoint whenever $(\Gamma_M \rhd M) \sharp (\Gamma_N \rhd N)$ is defined, and that $(\Gamma_M \rhd M) \sharp (\Gamma_N \rhd N) = (\Gamma_M \cup \Gamma_N) \rhd (M \mid N)$. In fact, we could have proved a more general result which states that any operator $|||_{\mathscr{P}}$ which preserves well-formedness of networks induces a closed algebra over the set CNets.

We have already shown that the operator $\sharp$ is non-commutative. Luckily, it enjoys other standard properties, one of which is associativity.

Proposition 3.2.3. [Associativity of $\sharp$] Let $\mathcal{M}, \mathcal{N}, \mathcal{L} \in \mathsf{CNets}$; then $\mathcal{M} \sharp (\mathcal{N} \sharp \mathcal{L}) \approx (\mathcal{M} \sharp \mathcal{N}) \sharp \mathcal{L}$. Here $\approx$ represents *Kleene's equality* [42]; that is, if $\mathcal{M} \sharp (\mathcal{N} \sharp \mathcal{L})$ is defined then so is $(\mathcal{M} \sharp \mathcal{N}) \sharp \mathcal{L}$, and $\mathcal{M} \sharp (\mathcal{N} \sharp \mathcal{L}) = (\mathcal{M} \sharp \mathcal{N}) \sharp \mathcal{L}$.

Proof. See Appendix A, Section A.2 □

Corollary 3.2.4. Let $\Gamma_{\mathbf{0}}$ be the empty graph, that is $(\Gamma_{\mathbf{0}})_V = \emptyset$, $(\Gamma_{\mathbf{0}})_E = \emptyset$. Let $O = \Gamma_{\mathbf{0}} \rhd \mathbf{0}$.

Then $\langle \mathsf{CNets}, \sharp, O \rangle$ is a partial monoid up-to structural congruence.

Proof. It is not difficult to show that the network $O \in \mathsf{CNets}$; further, for any $\mathcal{M} \in \mathsf{CNets}$ we obtain that both $O \sharp \mathcal{M}$ and $\mathcal{M} \sharp O$ are defined and structurally congruent to $\mathcal{M}$. Thus, O is an identity element for composable networks with respect to the composition operator $\sharp$ and up-to structural congruence.

Associativity follows directly from Proposition 3.2.3. □

In the future we will call the network O the *identity network*.

A pleasing property for the partial monoid $\langle \mathsf{CNets}, \|, O \rangle$ is that it can be generated starting from a given set of networks. That is, we can select a subset of networks such that its *closure* with respect to the operator $\sharp$ coincides, up-to structural congruence, with $\mathsf{CNets} \setminus \{O\}$.

Definition 3.2.5 (Generating networks)**.** Let $\mathcal{M} = \Gamma_M \rhd M \in \mathsf{CNets}$, with $M = m[\![P]\!]$ for some node m and process P. Then we say that $\mathcal{M}$ is a *generating network*.

We use the symbol $\mathbb{G}$ to denote the set of all generating networks.

Theorem 3.2.6. Let $\mathcal{M} = \Gamma_M \rhd M \in \mathsf{CNets}$ such that $|\text{nodes}(M)| \geq 1$. Then there exist a network $\mathcal{N} \in \mathsf{CNets}$ and a generating network $\mathcal{G} \in \mathbb{G}$ such that $\mathcal{M} \equiv \mathcal{N} \sharp \mathcal{G}$.

Proof. Let $\mathcal{M} = \Gamma_M \rhd M \in \mathsf{CNets}$, $|\text{nodes}(M)| \geq 1$. Then there exist a node m, a process P and a system term N such that $M \equiv m[\![P]\!] \mid N$.

We define $\mathcal{G}$ to be the network $\Gamma_G \rhd m[\![P]\!]$, where Γ_G is defined by

$$\begin{aligned} (\Gamma_G)_V &= \{m\} \cup \{n \in \mathsf{Int}(\Gamma_M \rhd M) \mid \Gamma_M \vdash m \leftrightarrows n\} \\ (\Gamma_G)_E &= \{(m', n) \in (\Gamma_G)_V \mid \Gamma_M \vdash m' \to n\} \end{aligned}$$

Now let $\mathcal{N}$ be the network $\Gamma_N \rhd N$, where Γ_N is defined below.

$$\begin{aligned} (\Gamma_N)_V &= \text{nodes}(N) \cup \{n \;\; \Gamma_M \vdash m' \leftrightarrows n \text{ for some } m' \in \text{nodes}(N)\} \\ (\Gamma_N)_E &= (\Gamma_M)_E \setminus (\Gamma_G)_E \end{aligned}$$

We need to show the following

1. $\mathcal{G} \in \mathbb{G}$
2. $\mathcal{N} \in \mathsf{CNets}$
3. $P_e(\mathcal{N}, \mathcal{G}) = \text{true}$

4. $(\Gamma_N \cup \Gamma_G) = \Gamma_M$

5. $N \,|\, m[\![P]\!] \equiv M$.

Statement (3) ensures that $\mathcal{N} \between \mathcal{G}$ is defined, while statements (4) and (5) establish that their composition is a network which is structurally congruent to $\mathcal{M}$.

Each of the statements above is proved separately. Statement (3) is an immediate consequence of how $(\Gamma_G)_V$ has been defined; in fact, it contains no nodes in nodes(N). Statement (5) follows by hypothesis; for all the three remaining statements the proofs are quite technical, and are therefore relegated to the Appendix. □

Theorem 3.2.6 has deep consequences. In fact, it provides an inductive definition of networks modulo structural congruence. This allows us to define an inductive proof principle for networks. However, since we are working modulo structural congruence, we can only use this proof method for proving properties which are preserved by structurally congruent networks.

Theorem 3.2.7 (Network Induction)**.** Let $\mathscr{P}$ be a property over CNets. Suppose also that $\mathscr{P}$ is preserved by structurally congruent networks; that is, for any composable networks $\mathcal{M}, \mathcal{N}$ such that $\mathcal{M} \equiv \mathcal{N}$, we have that $\mathscr{P}(\mathcal{M}) \Rightarrow \mathscr{P}(\mathcal{N})$.

If

- $\mathscr{P}(\mathcal{O})$
- $\forall \mathcal{N} \in \mathsf{CNets}, \mathcal{G} \in \mathbb{G}. \mathscr{P}(\mathcal{N}) \wedge \mathscr{P}_e(\mathcal{N}, \mathcal{G}) \Rightarrow \mathscr{P}(\mathcal{N} \between \mathcal{G})$

then $\forall \mathcal{M} \in \mathsf{CNets}. \mathscr{P}(\mathcal{M})$.

Proof. Let $\mathscr{P}$ be a property over CNets which is preserved by structurally congruent networks. Suppose that $P(\mathcal{O})$, and for all networks $\mathcal{N}$ and generating networks $\mathcal{G}$, if $\mathscr{P}(\mathcal{N})$ and $\mathscr{P}_e(\mathcal{N}, \mathcal{G})$, then $\mathscr{P}(\mathcal{N} \between \mathcal{G})$.

Let $\mathcal{M} = \Gamma_M \rhd M \in \mathsf{CNets}$. We show, by induction on $|\text{nodes}(M)|$, that $\mathscr{P}(\mathcal{M})$.

- $|\text{nodes}(M)| = 0$. Since $\mathcal{M} \in \mathsf{CNets}$, network, it is straightforward to show that $\mathcal{M} \equiv \mathcal{O}$. Since $\mathscr{P}(\mathcal{O})$ and $\mathscr{P}$ is preserved by structural congruent networks, we also have that $\mathscr{P}(\mathcal{M})$.

- $|\text{nodes}(M)| > 0$. By the inductive hypothesis, we assume that the property $\mathscr{P}$ holds for any network $\Gamma'_M \rhd M'$ such that $|\text{nodes}(M')| = |\text{nodes}(M)| - 1$. By Theorem 3.2.6 there exist a network $\mathcal{N} = \Gamma_N \rhd N$ and a generating network $\mathcal{G} = \Gamma_G \rhd G$, such that $\mathcal{M} \equiv \mathcal{N} \between \mathcal{G}$. Since the composition of $\mathcal{N}$ with $\mathcal{G}$ is defined, we have that $\mathscr{P}_e(\mathcal{N}, \mathcal{G}) = \mathsf{true}$. Further, it is straightforward to note that $|\text{nodes}(N)| = |\text{nodes}(M)| - 1$; this is because $|\text{nodes}(G)| = 1$, for $\mathcal{G}$ is a generating network and $\text{nodes}(G) \cap \text{nodes}(M) = \emptyset$. By inductive hypothesis, we have that $\mathscr{P}(\mathcal{N})$ holds. Since $P(\mathcal{N})$ and $\mathscr{P}_e(\mathcal{N}, \mathcal{G})$ are both true, we can apply the hypothesis to infer that $\mathscr{P}(\mathcal{N} \between \mathcal{G})$ is also true. It remains to note that $\mathscr{P}$ is preserved by structurally congruent networks, and $\mathcal{M} \equiv \mathcal{N} \,||\, \mathcal{G}$. Thus, $\mathscr{P}(\mathcal{M})$ holds as well.

□

Henceforth we will always deal with properties which are preserved by structurally congruent networks.

3.3 Behavioural Preorders for Networks

In this Section we develop two testing preorders for composable networks, based on Hennessy's and de Nicola's *may-testing* and *must-testing* preorders [17, 33].

First we show how these preorders can be defined, starting for a generic composition operator $|||_P$. In this case, the definition of the behavioural preorder is parametric in the composition operator that it is used to compose networks. Then we place some properties that we want our preorders to enjoy; we show that the largest composition operator whose induced testing preorders satisfy these properties is exactly $\between$.

Roughly speaking, the testing framework is based on the idea that we can interact with a network $\mathcal{M}$, by observing the values that its internal nodes send to output ones and by broadcasting values to the network from input nodes. Formally, we can design a network $\mathcal{T}$ and compose it with the network $\mathcal{M}$ via a composition operator $|||_{\mathscr{P}}$; then we can define how these two networks interact with each other by looking at the behaviour of the resulting network $\mathcal{M} |||_{\mathscr{P}} \mathcal{T}$, if it is defined. [1]

Usually the network $\mathcal{T}$ is designed to determine if the component $\mathcal{M}$ being tested satisfies a given property. For example, we would like to check if network $\mathcal{M}$ can eventually broadcast a value along channel c, which can be detected by an external node o. To this end, we can define the network $\mathcal{T}$ to have a receiver along channel c placed at node o, which guards the special process ω; the latter is used by $\mathcal{T}$ to report that the network $\mathcal{M}$ has performed the activity for which $\mathcal{T}$ was designed. This scenario is discussed formally in Example 3.3.4.

Definition 3.3.1 (Tests and Experiments)**.** Sometimes we will use the term *test*, or *testing network* to denote networks included in the set CNets.

Let $|||_{\mathscr{P}}$ be a composition operator. Given two tests $\mathcal{T}_1$ and $\mathcal{T}_2$, the network $\mathcal{T}_1 |||_P \mathcal{T}_2$, if defined, is called an *experiment*.

In the future, we will refer to networks in which the process ω does not appear in the code of any of its nodes by using the term *proper networks*.

When testing the behaviour of an experiment $\mathcal{M} |||_{\mathscr{P}} \mathcal{T}$, we assume that no other component can interact with it. That is, the network $\mathcal{M} |||_{\mathscr{P}} \mathcal{T}$ is isolated from the external environment. We have already developed a semantics for networks that only consider the interactions between internal nodes of a network. This is the Reduction Semantics, defined in Table 2.3. Computations for the experiment $\mathcal{M} |||_{\mathscr{P}} \mathcal{T}$ are maximal sequences of reduction steps rooted in the network itself.

Definition 3.3.2 (Computations)**.** Let $\mathcal{T}$ be a test. A *computation fragment* for it is a sequence

$$\mathcal{T} \twoheadrightarrow \mathcal{T}_1 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{T}_n \twoheadrightarrow \cdots$$

It can be either finite and infinite. A *computation* is a maximal computation fragment. Note that every infinite computation fragment is also a computation.

When referring to a network which appears in a computation, we will sometimes use the term *configuration*.

It is possible that an experiment $\mathcal{M} |||_{\mathscr{P}} \mathcal{T}$ contains a configuration in which in the code of at least one node the special process ω is enabled. In this case we say that the computation is successful.

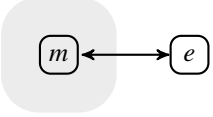
Definition 3.3.3 (Successful Computations)**.** A configuration $\Gamma_T \rhd T$ is successful if $T \equiv \omega + P[\![m]\!] \,|\, T'$ for some node m, process P and system term T'.

A computation fragment

$$\mathcal{T}_0 \twoheadrightarrow \mathcal{T}_1 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{T}_n \twoheadrightarrow \mathcal{T}_{n+1} \twoheadrightarrow \cdots$$

is *successful* if there exists an index $n \geq 0$ such that $\mathcal{T}_n$ is successful. . A computation is *successful* if it contains a finite successful computation fragment.

Example 3.3.4 (Testing a Network)**.** Consider the two networks $\mathcal{M} = \Gamma_M \rhd m[\![c!\langle v\rangle]\!]$ and $\mathcal{N} = \Gamma_M \rhd n[\![\mathbf{0}]\!]$, where Γ_M is depicted below.



Let also $\mathcal{T} = \Gamma_T \rhd e[\![c?\langle x\rangle.\omega]\!]$, where Γ_T is defined by $\Gamma_T \vdash e$ and $(\Gamma_T)_E = \emptyset$. For this Example we consider the extension operator $\sharp$.

[1] Note that the operator $|||_P$ is not necessarily commutative, so that there are actually two different ways in which these networks can interact; $\mathcal{M} |||_P \mathcal{T}$ and $\mathcal{T} |||_T \mathcal{M}$.

It is trivial to show that $\mathcal{M} \between \mathcal{T}$ is defined. In this network m can broadcast value v along channel c, and node e (which is connected to m) is waiting to receive a value along the same channel, we have the reduction

$$(\Gamma_M \rhd m[\![c!\langle v\rangle]\!]) \between \Gamma_T \rhd e[\![c?(x).\omega]\!] \twoheadrightarrow (\Gamma_M \rhd m[\![\mathbf{0}]\!]) \between (\Gamma_T \rhd e[\![\omega]\!]) \tag{3.1}$$

No reductions are possible for the network in the right-hand side of this reduction; therefore, Equation (3.1) is a maximal computation fragment, and therefore a computation. Further, the last configuration in this computation is successful, and therefore Equation 3.1 is a successful computation.

Now consider the network $\mathcal{N} \between \mathcal{T}$, which is defined. It is straightforward to note that this network is deadlocked. Thus, the only possible computation for this network consists of the network itself, with no reduction steps. As the configuration $\mathcal{N} \between \mathcal{T}$ is not successful, it follows that this network has no successful computations. □

Due to the non-deterministic nature of the calculus, it is possible that an experiment $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ contains both computations which are successful and computations which are not successful. This leads to two different scenarios.

Definition 3.3.5 (**may-pass**, **must-pass**)**.** Let $\mathcal{M}, \mathcal{T}$ be a two testing networks. Let also $|||_{\mathscr{P}}$ be an arbitrary composition operator, and suppose $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ is defined.

We say that $\mathcal{M}$ $\mathscr{P}$ – **may-pass** $\mathcal{T}$ if the experiment $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ has at least a successful computation, and that $\mathcal{M}$ $\mathscr{P}$ – **must-pass** $\mathcal{T}$ if all the computations of the experiment $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ are successful.

Usually we will choose $\mathcal{M}$ to be a proper network.

Note that the relations of Definition 3.3.5 are parametric in a consistency predicate $\mathscr{P}$.

We are now ready to define how networks can be compared. This amounts to relate them according to the tests they may (must) pass. Since a test interacts with a network through the external nodes of the latter, we only compare networks which have the same input and output interfaces.

Definition 3.3.6 (Testing Preorders)**.** Let $\mathcal{M}, \mathcal{N}$ be two testing networks. Suppose also that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}), \mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$. We say that $\mathcal{M} \sqsubseteq^{\mathscr{P}}_{\text{may}} \mathcal{N}$ if, for any testing network $\mathcal{T}$ such that $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ and $\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{T}$ are both defined, then $\mathcal{M}$ $\mathscr{P}$ – **may-pass** $\mathcal{T}$ implies $\mathcal{N}$ $\mathscr{P}$ – **may-pass** $\mathcal{T}$. The preorder $\sqsubseteq^{\mathscr{P}}_{\text{must}}$ is defined similarly, using the $\mathscr{P}$ – **must-pass** testing relation.

Remark 3.3.7. Note that Definition 3.3.6 states that two networks $\mathcal{M}, \mathcal{N}$ may be related via one of the testing preorder $\sqsubseteq_{\text{may}}, \sqsubseteq_{\text{must}}$ only in the case that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$. To explain why we introduced this constraint, let us think again of a network $\mathcal{M}$ as a (broadcast) distributed system. The nodes in $\mathsf{Input}(\mathcal{M})$ correspond to terminals which can be used to provide requests to the distributed system, while nodes in $\mathsf{Output}(\mathcal{M})$ can be seen as terminals which can be used to fetch messages forwarded from the distributed system to the external environment. In our own point of view, an end user is always able to distinguish between two distributed systems, mathematically represented as $\mathcal{M}, \mathcal{N}$ if he notes a difference in the set of terminals that she can use to interact with them, without even trying to interact with such systems.

Definition 3.3.6 is again parametric in a consistency predicate $\mathscr{P}$. However, most of them are not of interest for our goal. As we already mentioned, we want to enable compositional reasoning over networks. Suppose that $\mathcal{M}, \mathcal{N}$ are two networks such that $\mathcal{M} \sqsubseteq^{\mathscr{P}}_{\text{may}} \mathcal{N}$ ($\mathcal{M} \sqsubseteq^{\mathscr{P}}_{\text{must}} \mathcal{N}$); also, let $\mathcal{L}$ be a testing network such that both $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{L}$ and $\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{L}$ are defined. In this case, we would like to obtain the equivalence $(\mathcal{M} |||_{\mathscr{P}}) \mathcal{L} \sqsubseteq^{\mathscr{P}}_{\text{may}} (\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{L})$ $((\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{L}) \sqsubseteq^{\mathscr{P}}_{\text{must}} (\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{L}))$. As Example 3.3.8 shows, this requirement is not satisfied by any arbitrary consistency predicate $\mathscr{P}$. In particular, it is not satisfied by the consistency predicate $\mathscr{P}_s$, whose induced operator is the symmetric composition operator $\|$.

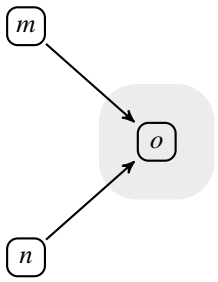
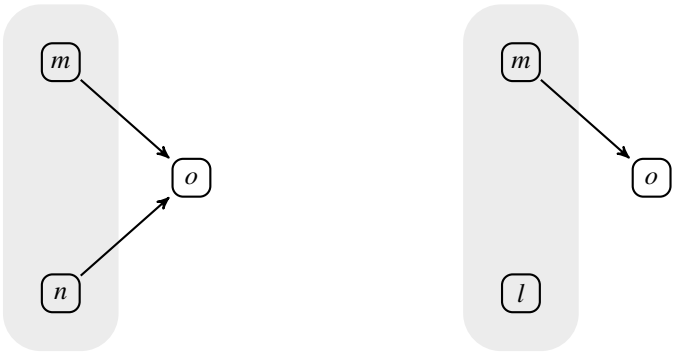


Figure 3.6: A simple connectivity graph Γ

Example 3.3.8 (Non Compositional Testing Preorder)**.** Let Γ_M, Γ_N be the connectivity graph depicted below



Let also $M = m[\![P]\!] \mid n[\![\mathbf{0}]\!]$, $N = m[\![P]\!] \mid l[\![\mathbf{0}]\!]$, and let us consider the testing preorders generated by the consistency predicate $\mathscr{P}_s$, introduced in Example 3.1.7.

The networks $\mathcal{M} = \Gamma_M \triangleright M$ and $\mathcal{N} = \Gamma_N \triangleright N$ can interact with the (only) external node o only via the node m. This is because node n is deadlocked in $\mathcal{M}$, while node l is not connected to any other node in $\mathcal{N}$. Further, they run the same code at node m. Therefore, it is immediate to note that $\mathcal{M} \sqsubseteq^{\mathscr{P}_s}_{\text{may}} \mathcal{N}$ and $\mathcal{M} \sqsubseteq^{\mathscr{P}_s}_{\text{must}} \mathcal{N}$.

Let now $\mathcal{L} = \Gamma \triangleright o[\![Q]\!]$ be another network, where Q is an arbitrary process and Γ is the connectivity graph depicted in Figure 3.6. It is easy to show that $\mathcal{M} \parallel \mathcal{L}$ and $\mathcal{N} \parallel \mathcal{L}$ are defined. In the first case, it is sufficient to check that the connections $\Gamma_M \vdash m \to o$ and $\Gamma_M \vdash n \to o$ are preserved by Γ_L, while in the second one we only need to check that the connectivity $\Gamma_N \vdash n \to o$ is preserved by Γ_L.

We would expect that $\mathcal{M} \parallel \mathcal{L} \sqsubseteq^{\mathscr{P}_s}_{\text{may}} \mathcal{N} \parallel \mathcal{L}$, and $\mathcal{M} \parallel \mathcal{L} \sqsubseteq^{\mathscr{P}_s}_{\text{must}} \mathcal{N} \parallel \mathcal{L}$. However, this is not true. In fact, note that $n \in \text{nodes}(M)$, while $\Gamma_N \nvdash n$. Also, $n \notin \text{nodes}(L)$. Thus, node n (which is included in the input interface of $\mathcal{L}$) is an internal node in the network $\mathcal{M} \parallel \mathcal{L}$, but it is an external node in $\mathcal{N} \parallel \mathcal{L}$. As a result, we have that $\mathsf{Output}(\mathcal{M} \parallel \mathcal{L}) \neq \mathsf{Output}(\mathcal{N} \parallel \mathcal{L})$, and by Definition 3.3.6 it holds $\mathcal{M} \parallel \mathcal{L} \not\sqsubseteq^{\mathscr{P}_s}_{\text{may}} \mathcal{N} \parallel \mathcal{L}$, $\mathcal{M} \parallel \mathcal{L} \not\sqsubseteq^{\mathscr{P}_s}_{\text{must}} \mathcal{N} \parallel \mathcal{L}$. □

Example 3.3.8 shows that a necessary condition for a preorder $\sqsubseteq^{\mathscr{P}}_{\text{may}}$ to be compositional, then it is necessary for the composition operator $|||_{\mathscr{P}}$ to preserve input and output interface equivalences.

Definition 3.3.9. Interface Preservation A composition operator $|||_{\mathscr{P}}$ is said to be *interface preserving* if, whenever $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$, and $\mathcal{L}$ is a network such that both $\mathcal{M} |||_{\mathscr{P}} \mathcal{L}$ and $\mathcal{N} |||_{\mathscr{P}} \mathcal{L}$ are defined, then

- $\mathsf{Input}(\mathcal{M} |||_{\mathscr{P}} \mathcal{L}) = \mathsf{Input}(\mathcal{N} |||_{\mathscr{P}} \mathcal{L})$,
- $\mathsf{Output}(\mathcal{M} |||_{\mathscr{P}} \mathcal{L}) = \mathsf{Output}(\mathcal{N} |||_{\mathscr{P}} \mathcal{L})$.

Clearly we want our testing preorders to be generated by consistency predicates whose induced composition operators are interface preserving. But this is not the only requirement we need.

Another issue that arises when testing networks concerns the internal nodes of a network. Intuitively, external nodes of a network can only detect values being broadcast by a node, but they cannot detect the name of the node that performed the broadcast. This is shown in Example 3.3.10

Example 3.3.10. Consider again the connectivity graph Γ of Figure 3.6. Let $\mathcal{M} = \Gamma \rhd m[\![P]\!] \,|\, n[\![\mathbf{0}]\!]$ and $\mathcal{N} = \Gamma \rhd m[\![\mathbf{0}]\!] \,|\, n[\![P]\!]$, where P is an arbitrary process.

The networks $\mathcal{M}$ and $\mathcal{N}$ are the same, except that the names of the nodes m and n have been switched. Note that the external node o in Γ can receive values which have been broadcast either by node m or n.

Now suppose that we place some code at node o. We expect this node to interact with the networks $\mathcal{M}$ and $\mathcal{N}$ in the same way. This is because node o can only detect a value being broadcast, but it has no information about the node which actually performed the broadcast. Therefore, we would expect networks $\mathcal{M}$ and $\mathcal{N}$ to be equivalent. □

From the point of view of composition operators, we require that the truth of a consistency predicate, used to define a composition operator, is preserved if we rename the internal nodes of a network.

Definition 3.3.11 (Renaming resistance)**.** Let $\mathcal{M} = \Gamma_M \rhd M$ be a network; a composition operator $|||_{\mathscr{P}}$ is said to be *renaming resistant* if, for any testing network $\mathcal{N} = \Gamma_N \rhd N$ such that $\mathcal{M} |||_{\mathscr{P}} \mathcal{N}$ is defined, and any permutation $\sigma : \mathbf{Nodes} \to \mathbf{Nodes}$ such that

- if $e \in \mathsf{Int}(\mathcal{M})$ then $\sigma(e) = e$
- for any $m \in \mathrm{nodes}(M)$, $\Gamma_N \not\vdash \sigma(m)$

then $\mathcal{M}\sigma |||_{\mathscr{P}} \mathcal{N}$ is defined as well.

Here $\mathcal{M}\sigma$ is the network obtained by substituting all the nodes m occurring in $\mathcal{M}$ with $\sigma(m)$. That is, $\mathcal{M}\sigma = (\Gamma_M)\sigma \rhd M\sigma$, where $((\Gamma_M)\sigma)_V = (\Gamma_M)_V$, $(\Gamma_M)\sigma \vdash m \to n$ if and only if $m = \sigma(m'), n = \sigma(n')$ for some m', n' such that $\Gamma_M \vdash m' \to n'$ and $M\sigma$ is defined by letting $\mathbf{0}\sigma = \mathbf{0}$, $m[\![P]\!]\sigma = \sigma(m)[\![P]\!]$ and $(M \,|\, N)\sigma = M\sigma \,|\, N\sigma$.

In Definition 3.3.11 we placed some constraints on the permutation σ that we want to apply to a testing network $\mathcal{N}$. Intuitively, the first one says that we are not allowed to change the name of the nodes at its interface; this is because those are the nodes to be used to connect $\mathcal{M}$ to another network. The second constraints says that we are not allowed to change the name of an internal node of $\mathcal{M}$ with one which is already being used by the network with which $\mathcal{M}$ is being composed. This is needed to ensure that the well-formedness of the composed network is preserved.

Remark 3.3.12. Let us discuss the intuition behind Definition 3.3.11; one could think of node names as *IP addresses* in a network. Renaming resistance allows us to abstract from such addresses; at a very informal level, we do not want to distinguish between two networks which differ only in the IP addresses associated with wireless stations[2].

The last property we require from a composition operator is that the preorders it induces allow us to distinguish networks with the same interface.

Definition 3.3.13 (Consistent Composition)**.** A composition operator $|||_{\mathscr{P}}$ is *consistent* if there exist two **proper** networks $\mathcal{M}, \mathcal{N}$[3] and a testing network $\mathcal{L}$ such that $\mathsf{Int}(\mathcal{M}) = \mathsf{Int}(\mathcal{N})$, $\mathcal{M} |||_{\mathscr{P}} \mathcal{L}$ and $\mathcal{N} |||_{\mathscr{P}} \mathcal{L}$ are defined and

- $\mathcal{M}$ $\mathscr{P}$ – **may-pass** $\mathcal{L}$ holds
- $\mathcal{N}$ $\mathscr{P}$ – **may-pass** $\mathcal{L}$ does not hold

Now that we have placed all the properties that we require from a composition operator, we can show that the largest operator that satisfies them is the extension operator $\between$.

Theorem 3.3.14. The operator $\between$ is interface preserving, renaming resistant, consistent and which preserves well-formedness of networks. Further, it is the largest operator that enjoys such properties.

[2]Note that in practice the IP address of a node is contained in all the packets it broadcasts. However, in our calculus we assume that a node never encapsulates its name in the contents of the packet it broadcasts, unless it is explicitly programmed in the code it runs. This enforces the concept that our testing preorders abstract from IP addresses of wireless stations.

[3]note that here we require such networks to be proper; they do not contain the success process ω in the code placed at their nodes

Proof. We first show that $\natural$ satisfies the requirements of definitions 3.3.9, 3.3.11 and 3.3.13; we have already shown that it preserves well-formedness of networks.

For interface preservation, suppose $\mathcal{M},\mathcal{N}$ are two testing networks such that $\mathsf{Int}(\mathcal{M}) = \mathsf{Int}(\mathcal{N})$ and. Let also $\mathcal{L} = \Gamma_L \rhd L$ be a testing network such that $\mathcal{M} \natural \mathcal{L}$ and $\mathcal{N} \natural \mathcal{L}$ are defined. By Lemma 3.2.1 it follows that

$$\mathsf{Int}(\mathcal{M} \natural \mathcal{L}) = (\mathsf{Int}(\mathcal{M}) \cup \mathsf{Int}(\mathcal{L})) \setminus \mathsf{nodes}(L) = (\mathsf{Int}(\mathcal{N}) \cup \mathsf{Int}(\mathcal{L})) \setminus \mathsf{nodes}(L) = \mathsf{Int}(\mathcal{N} \natural \mathcal{L})$$

as we wanted to prove.

For renaming resistance, let $\mathcal{M} = \Gamma_M \rhd M, \mathcal{N} = \Gamma_N \rhd N$ be testing networks such that $\mathcal{M} \natural \mathcal{N}$ is defined. Let also σ be a permutation such that, for any $e \in \mathsf{Int}(\mathcal{M}), \sigma(e) = e$ and for any $m \in \mathsf{nodes}(M), \Gamma_N \nvdash \sigma(m)$. Then it is immediate to show that $\mathsf{nodes}(M\sigma) \cap (\Gamma_N)_V = \emptyset$, and therefore $\mathcal{M}\sigma \natural \mathcal{N}$ is defined.

It remains to check that the operator $\natural$ is consistent. To this end, it suffices to show two testing networks which can be distinguished by a test, according to the $\natural$ *−may-pass* testing relation. However, we already exhibited two testing networks which can be distinguished by a test in Example 3.3.4, so that in this case there is nothing to prove.

Now we prove that $\natural$ is indeed the largest operator which is interface preserving, renaming resistant, consistent and which preserves networks well-formedness. In this case, we just need to show that whenever a composition operator $|||_{\mathscr{P}}$ satisfies such properties then, if $\mathcal{M} |||_{\mathscr{P}} \mathcal{N}$, where $\mathcal{M} = \Gamma_M \rhd M, \mathcal{N} = \Gamma_N \rhd N$ is defined, it holds $(\Gamma_N)_V \cap \mathsf{nodes}(M) = \emptyset$.

The proof is by contradiction. Suppose that $|||_{\mathscr{P}}$ is a composition operator which is interface preserving and renaming resistant[4]. Let also $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$ two networks such that $\mathcal{M} |||_{\mathscr{P}} \mathcal{N}$ is defined, and $(\Gamma_N)_V \cap \mathsf{nodes}(M) \neq \emptyset$. Thus, there exists a node n such that $n \in \mathsf{nodes}(M)$ and $\Gamma_N \vdash n$.

Consider now the permutation σ defined by $\sigma(n) = l$, where $\Gamma_M \nvdash l, \Gamma_N \nvdash l$, $\sigma(l) = n$ and $\sigma(m) = m$ for all m such that $m \neq l, n \neq l$. That is, the permutation σ replaces the node name n with a fresh node name l. By renaming resistance, we have that $\mathcal{M}\sigma \natural \mathcal{N}$ is defined.

Further, we have that $(\Gamma_M)\sigma \nvdash n$, $\Gamma_N \vdash n$, so that it Since $n \in \mathsf{nodes}(M)$ and $\mathcal{M} |||_P \mathcal{N}$ is well defined, it also follows that $n \notin \mathsf{nodes}(N)$, hence $n \in \mathsf{Int}(\mathcal{N})$. This is because we are assuming that the operator $|||_{\mathscr{P}}$ preserves well-formed networks, that is whenever $(\Gamma_M \rhd M) |||_{\mathscr{P}} (\Gamma_N \rhd N)$ is defined we have $\mathsf{nodes}(M) \cap \mathsf{nodes}(N) = \emptyset$.

Since $(\Gamma_M)\sigma \nvdash n$, $n \in \mathsf{Int}(\Gamma_M)$, it is straightforward to show that $n \in \mathsf{Int}(\mathcal{M}\sigma |||_P \mathcal{N})$. However, since $n \in \mathsf{nodes}(M)$, we have that $n \notin \mathsf{Int}(\mathcal{M} |||_P \mathcal{N})$. Thus, $\mathsf{Int}(\mathcal{M} |||_P \mathcal{N}) \neq \mathsf{Int}(\mathcal{M}\sigma |||_P \mathcal{N})$. By interface preservation, it follows that $\mathsf{Int}(\mathcal{M}) \neq \mathsf{Int}(\mathcal{M}\sigma)$. This is not possible; in fact, if $e \in \mathsf{Int}(\mathcal{M})$, we have that $\sigma(e) = e \in \mathsf{Int}(\mathcal{M}\sigma)$. Conversely, if $e \in \mathsf{Int}(\mathcal{M}\sigma)$, then there exists $e' \in \mathsf{Int}(\mathcal{M})$ such that $e = \sigma(e')$. By definition of σ we have that $e' = e$, hence $e \in \mathsf{Int}(\mathcal{M})$. Thus $e \in \mathsf{Int}(\mathcal{M})$ iff $e \in \mathsf{Int}(\mathcal{M}\sigma)$, or equivalently $\mathsf{Int}(\mathcal{M}) = \mathsf{Int}(\mathcal{M}\sigma)$. Contradiction. □

Recall that the extension operator $\natural$ is the composition operator induced by the consistency predicate $\mathscr{P}_e$. In the following we use the notation $\mathcal{M}$ **may-pass** $\mathcal{T}$ and $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ in lieu of $\mathcal{M}\mathscr{P}_e - \text{may-pass}\, \mathcal{T}$ and $\mathcal{M} \sqsubseteq_{\text{may}}^{\mathscr{P}_e} \mathcal{N}$, respectively. Also, we say that $\mathcal{M} =_{\text{may}} \mathcal{N}$ when both $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ and $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$ hold. In a similar way, we can define the predicates $\mathcal{M}$ **must-pass** $\mathcal{T}$, $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$ and $\mathcal{M} =_{\text{must}} \mathcal{N}$.

Finally, we define the testing preorder $\subseteq$ to coincide with the intersection of the preorder $\sqsubseteq_{\text{may}}, \sqsubseteq_{\text{must}}$; that is $\mathcal{M} \subseteq \mathcal{N}$ whenever $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ and $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$. The equivalence relation induced by the preorder $\subseteq$ is denoted as $\simeq$.

We now show that the preorders $\sqsubseteq_{\text{may}}, \sqsubseteq_{\text{must}}$ are compositional. This is needed, as interface preservation is a necessary but not sufficient condition to ensure compositionality of a testing preorder.

Proposition 3.3.15. Suppose $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ ($\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$) and let $\mathcal{L}$ be a testing network such that $\mathcal{M} \natural \mathcal{L}$, $\mathcal{N} \natural \mathcal{L}$ are defined. Then $\mathcal{M} \natural \mathcal{L} \sqsubseteq_{\text{may}} \mathcal{N} \natural \mathcal{L}$ ($\mathcal{M} \natural \mathcal{L} \sqsubseteq_{\text{must}} \mathcal{N} \natural \mathcal{L}$).

Proof. We prove the statement only for the preorder $\sqsubseteq_{\text{may}}$; the proof for $\sqsubseteq_{\text{must}}$ is analogous.

First note that if $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ then $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$. By interface preservation we have that $\mathsf{Input}(\mathcal{M} \natural \mathcal{L}) = \mathsf{Input}(\mathcal{N} \natural \mathcal{L})$. Similarly, we can prove that $\mathsf{Output}(\mathcal{M} \natural \mathcal{L}) = \mathsf{Output}(\mathcal{N} \natural \mathcal{L})$.

[4] for this proof, we do not require $|||_{\mathscr{P}}$ to be consistent

Now let $\mathcal{T}$ be a test such that $(\mathcal{M} \between \mathcal{L})$ **may-pass** $\mathcal{T}$ and $(\mathcal{N} \between \mathcal{L}) \between \mathcal{T}$ is defined. We have to show that $(\mathcal{N} \between \mathcal{L})$ **may-pass** $\mathcal{T}$.

In this case we have that $((\mathcal{M} \between \mathcal{L}) \between \mathcal{T})$ has a successful computation. By Associativity of $\between$, Proposition 3.2.3, it follows that $\mathcal{M} \between (\mathcal{L} \between \mathcal{T})$ is defined, and equal to $(\mathcal{M} \between \mathcal{N}) \between \mathcal{T}$. Thus $\mathcal{M} \between (\mathcal{L} \between \mathcal{T})$ has a successful computation, or equivalently $\mathcal{M}$ **may-pass** $(\mathcal{L} \between \mathcal{T})$. Note that, since $(\mathcal{N} \between \mathcal{L}) \between \mathcal{T}$ is defined, by associativity we have that $\mathcal{N} \between (\mathcal{L} \between \mathcal{T})$ is defined and equal to $(\mathcal{N} \between \mathcal{L}) \between \mathcal{T}$. For $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, and $\mathcal{N} \between (\mathcal{L} \between \mathcal{T})$ is defined, it follows that $\mathcal{N}$ **may-pass** $(\mathcal{L} \between \mathcal{T})$. In other words, the experiment $\mathcal{N} \between (\mathcal{L} \between \mathcal{T})$, which is equivalent to $(\mathcal{N} \between \mathcal{L}) \between \mathcal{T}$ by associativity, has a successful computation. Hence $(\mathcal{N} \between \mathcal{L})$ **may-pass** $\mathcal{T}$, as we wanted to prove. □

By Proposition 3.3.14 and Proposition 3.3.15 we obtain that the extension operator $\between$ is the largest composition operator whose induced testing preorders are compositional. However, we have already shown that it is not commutative. The reader could argue that it would be better to choose another composition operator, which despite being smaller than $\between$ it enjoys the commutative property, to define the testing preorder. As we show in Proposition 3.3.16 this is not possible, as in this case the induced preorder could not be used to distinguish networks with the same interface.

Proposition 3.3.16. Let $|||_{\mathscr{P}}$ be a composition operator such that, whenever $(\Gamma_M \rhd M) \,|||_{\mathscr{P}}\, (\Gamma_N \rhd N)$ is defined, then $(\Gamma_N)_V \cap \text{nodes}(M) = \emptyset$. If $|||_{\mathscr{P}}$ is commutative, then it is not consistent.

Outline. We only give an outline of the proof, as a complete argument would require a rather technical and long proof.

Suppose that $|||_{\mathscr{P}}$ is a composition operator as in the hypothesis of the proposition; also, assume it is commutative.

Then, whenever $(\Gamma_M \rhd M) \,|||_{\mathscr{P}}\, (\Gamma_N \rhd N)$ is defined, we have that $(\Gamma_N \rhd N) \,|||_{\mathscr{P}}\, (\Gamma_M \rhd M)$ is defined too. By hypothesis, it holds that $(\Gamma_N)_V \cap \text{nodes}(M) = \emptyset$, and $(\Gamma_M)_V \cap \text{nodes}(M) = \emptyset$.

That is, whenever $m \in \text{nodes}(M)$, we have that $\Gamma_N \vdash N \not\leftrightarrows m$; similarly, for any $n \in \text{nodes}(M)$ it holds that $\Gamma_M \vdash M \not\leftrightarrows n$. An immediate consequence of these two statements is that for any $m \in \text{nodes}(M)$, $n \in \text{nodes}(N)$, we have that $\Gamma_M \vdash m \not\leftrightarrows n$ and $\Gamma_N \vdash m \not\leftrightarrows n$, hence $(\Gamma_M \cup \Gamma_N) m \not\leftrightarrows n$.

Informally, the last statement establishes that whenever $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{N}$ is defined, then $\mathcal{M}$ and $\mathcal{N}$ cannot interact with each other in the composed network. This is because it is not possible to connect the internal nodes of $\mathcal{M}$ with those of $\mathcal{N}$ in the composed network.

Let now $\mathcal{M}, \mathcal{N}$ be two networks such that $\mathsf{Int}(\mathcal{M}) = \mathsf{Int}(\mathcal{N})$, and let $\mathcal{T}$ be a test such that $\mathcal{M} \,|||_P\, \mathcal{T}$, $\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{T}$ are defined. Suppose also that $\mathcal{M}$ $\mathscr{P}$ – **may-pass** $\mathcal{N}$. Since the special process ω cannot appear in $\mathcal{M}$ (recall that we are assuming that it is a proper network), and the internal nodes of $\mathcal{M}$ and $\mathcal{T}$ cannot interact in the composed network $\mathcal{M} \between \mathcal{T}$, we have that the experiment $\mathcal{M} \,|||_{\mathscr{P}}\, \mathcal{T}$ has a successful computation if and only if $\mathcal{T}$ has one as well. Thus, we have that $\mathcal{T}$ has a successful computation, hence (again since the internal nodes of $\mathcal{N}$ and those of $\mathcal{T}$ are not connected to each other) $\mathcal{N} \,|||_{\mathscr{P}}\, \mathcal{T}$ has a successful computation as well. Thus $\mathcal{N}$ $\mathscr{P}$ – **may-pass** $\mathcal{T}$.

Note that the proper networks $\mathcal{M}, \mathcal{N}$ and the test $\mathcal{T}$ are arbitrary. By Definition 3.3.13 the composition operator $|||_{\mathscr{P}}$ is not consistent. □

We conclude this Chapter by providing another example that explains in further detail our choice of focusing on composable networks for defining our testing preorders.

Example 3.3.17. Remarks on composable networks Let $\mathcal{M}, \mathcal{N}$ be the networks of Example 3.1.2, and suppose that the testing preorders $\sqsubseteq_{\text{may}}$ and $\sqsubseteq_{\text{must}}$ have been defined to compare well-formed networks.

We have already noticed that the only action that networks $\mathcal{M}$ and $\mathcal{N}$ can perform corresponds to letting node m broadcast value v along channel c, and that this value can be detected by the output nodes n and l. From the point of view of their behaviour, they do not present any difference; therefore, we would expect $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$.

However, the networks $\mathcal{M}$ and $\mathcal{N}$ can be distinguished by the test $\Gamma_T \triangleright T'$, where Γ_T has been defined again in Example 3.1.2, and T' is the process $n[\![c?(x).c!\langle x\rangle.\mathbf{0}]\!] \mid l[\![c?(x).c?(y).\omega]\!]$. It is straightforward now to note that $\mathcal{M}$ **may-pass** $\mathcal{T}$ and $\mathcal{M}$ **must-pass** $\mathcal{T}$, while $\mathcal{N}$ **may-pass** $\mathcal{T}$ and $\mathcal{N}$ **must-pass** $\mathcal{T}$ are not true. □

3.4 Related Work

In this chapter we analysed the problem of developing a compositional theory for the calculus defined in Chapter 2. We successfully managed to adapt the well-known may-testing and must-testing preorders to our framework. Below we compare our approach with other compositional theories for wireless networks that have been investigated by the research community.

Our approach to compositionality has been mainly inspired by [47]; here each node in a wireless networks is associated with a semantic tag, consisting of the set of its neighbours. A network M is composed with another N, whose semantics tags are only partially defined. The composition $M|N$ consists of the network M running in parallel with a network N', which is obtained by adding to the nodes contained in N the semantic tags needed to make the connection of the latter network consistent with those contained in M. The authors adapt the notion of reduction barbed congruence [49] to their framework. Given two networks M, N which are deemed equivalent in their framework, and another network L, their behavioural equivalence allows to infer that $M|L$ and $N|L$ are deemed to be equivalent. However, since the changes of the semantic tags in the sub-network L depends on the networks M, N respectively, the network $M|L$ can be seen as M running in parallel with a network L', while $N|L$ can be seen as N running in parallel with another network L''; since L' and L'' can be different networks, for they can differ in the semantic tags associated with nodes, we do not regard this approach as compositional.

In [27] the authors define a notion of *rooted branching bisimulation* in their framework, and they prove such a relation to be a congruence in their framework. Despite the main advantage that, in their framework the network topology is embedded in the definition of processes, which avoids problems when defining network composition, their behavioural equivalence is defined over the intensional activities of networks, rather than taking into account their possible interactions with the external environment. In particular, it is the case that their notion of bisimulation distinguishes between two different locations running the same code; as we have already discussed in this chapter, our behavioural preorders abstract from names of locations.

The last paper which presents compositional theory for wireless networks, to the best of our knowledge, is [62]. As we have already argued in Chapter 2 in our own point of view their ω-calculus models networks at the *Transport Layer*, rather than at the *Network Layer*, of the *ISO/OSI reference model*. On the other hand, the approach followed by the authors of abstracting from node names and grouping processes into subnets (or groups) allows a nice approach of compositionality. As the authors point out in the paper, each network defined via connectivity graphs can be encoded in their calculus; this can be done by partitioning the connectivity graph of a network in maximal cliques, and defining a group for each of them. While it would be possible to adapt the definition of testing preorders to the ω-calculus and give a translation $(\!|\cdot|\!)$ which encodes networks in our calculus into networks in the ω-calculus, it turns out that the testing preorders would not be preserved.

That is, it is possible to exhibit two networks $\mathcal{M}, \mathcal{N}$ such that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ but $(\!|\mathcal{M}|\!) \not\sqsubseteq_{\text{may}} (\!|\mathcal{N}|\!)$. In other words, tests in the ω-calculus are more powerful than in our framework. Intuitively speaking, given a network $\mathcal{M} = \Gamma \triangleright M$ with an input node i such that $\Gamma \vdash i \to m$, $\Gamma \vdash i \to n$ (here we assume $m, n \in \text{nodes}(M)$) it is possible to define a test in the ω-calculus which interacts with $(\!|\mathcal{M}|\!)$ by broadcasting a message only to the internal node m. In our framework this is not possible, for whenever the input node i broadcasts a message, it can be detected by both the internal nodes m, n. Since we assume that we are working with static networks, we prefer the latter approach, for granting an input node to broadcast a message only to a part of its neighbour can be considered as a form of mobility.

Chapter 4

Characterisation of the Testing Preorders

In Chapter 3 we managed to define testing preorders for composable networks. However, due to the quantification over all possible tests, it is difficult to prove that two networks $\mathcal{M}$ and $\mathcal{N}$ with the same input and output interface are may-testing (must-testing) related.

In this Chapter we develop proof methods that can be used for proving that two networks are may-testing (must-testing) related. These are based on the idea that, to check whether two networks are related via some testing preorder, we just need to know their observational behaviour; this corresponds to the activities that a network can perform, and which can be verified by a test (according to the may-pass or must-pass testing relation).

Also suitable relations, whose definitions are based solely on the observable behaviour of networks, have to be defined in a way such that they characterise the testing preorders we have defined. For standard process calculi, such as *CCS* and *CSP*, it is well known that the may-testing preorder is characterised by *trace inclusion*, while the must-testing preorder is characterised by the *Smith's preorder* induced by *acceptance trees*. See [17, 33] for details.

We remark that these results hold in a setting where some limitations over terms of CCS (or CSP) are placed; specifically, we require the LTS induced by a term to be finite state and finite branching, the latter meaning that each state in the LTS such a term generates has a finite number of transitions.

Here we revisit the question for (composable) networks. We provide a characterisation for the $\sqsubseteq_{\text{may}}$ and $\sqsupseteq_{\text{must}}$ inclusion, respectively in terms of traces and deadlock traces inclusion. These results however hold only if we impose some limitations to the structure of networks, which are discussed later.

Intuitively, a trace of a network corresponds to a sequence of activities which can be detected by the external environment, which can possibly end with a special mark ω denoting that a successful configuration has been reached; deadlock traces are defined similarly, but the possibility of reaching a configuration which cannot evolve, with respect to the reduction semantics (Figure 2.3), is also taken into account.

As we will point out, activities which can be observed by external nodes of a network differ from the actions of the labelled transition semantics of Figure 2.4. Formally, we define the extensional behaviour of composable networks by introducing an *extensional LTS* for them, whose transitions correspond to activities which can be checked by the external environment.

Since internal transitions cannot be observed by a test, we expect to use their weak version [51, 33, 68] to prove our results. However, the definition of weak transitions in our framework is non-standard; this is needed to ensure that our proof methods for the testing preorders are complete.

Our first full abstraction result states that, in a finitary setting, trace inclusion coincides with the may-testing preorder. For the must-testing preorder, however, we have to impose a further restriction to the structure of networks. Specifically, we only focus on strongly convergent networks; these are networks whose generated LTS does not include any state where the computation can proceed indefinitely. Under this assumption, we prove that deadlock trace inclusion coincides with the inverse of the must-testing preorder, $\sqsupseteq_{\text{must}}$.

The rest of the Chapter is organised as follows: in Section 4.1 we define formally the extensional behaviour

$\mathcal{M} = \Gamma_M \rhd m[\![c!\langle v\rangle]\!]$ $\quad$ $\mathcal{N} = \Gamma_N \rhd n[\![c!\langle v\rangle]\!]$

Figure 4.1: Two networks $\mathcal{M}$, $\mathcal{N}$ such that $\mathcal{M} =_{\text{may}} \mathcal{N}$ and $\mathcal{M} =_{\text{must}} \mathcal{N}$.

of networks. We provide illuminating example that show the need for a non-standard definition of weak extensional actions.

In Section 4.2 we prove decomposition and composition results for weak extensional actions, which will be needed to prove soundness of our proof principles. In Section 4.2.1 we show how we can infer the (weak extensional) behaviour of two networks $\mathcal{M}, \mathcal{N}$ if that of $\mathcal{M} \parallel \mathcal{N}$ is known. Note that here we use the symmetric operator $\parallel$; this cannot be helped, for we provide examples that show it is not possible in general to prove decomposition results for the extension operator $\between$, due to its asymmetric nature. However, this does not cause any major problems in the proofs of our characterisation results, as there is a close relationship between the operators $\parallel$ and $\between$. In Section 4.2.2 we prove composition properties for weak extensional actions. If the (weak) behaviour of two networks $\mathcal{M}$, $\mathcal{G}$ is known, then we are able to infer that of the composed network $\mathcal{M} \parallel \mathcal{G}$; however, such results hold only if we assume that $\mathcal{G}$ is a generating network.

In Section 4.3 we provide a full abstraction result for the $\sqsubseteq_{\text{may}}$ testing preorder. The proof is split in two parts; in Section 4.3.1 we show that trace inclusion for testing networks is sound with respect to the $\sqsubseteq_{\text{may}}$ testing preorder, while in Section 4.3.2 we show that completeness also holds. The proofs of all the propositions are discussed in deep detail. Finally, in Section 4.3.3 we provide some simple applications of our proof principle, by focusing on networks which have already been discussed in the previous chapters.

In Section 4.4 we repeat the work for must-testing, this time using the acceptance-trees preorder. Again, the proof is split in two parts; soundness is proved in Section 4.4.1, while completeness is proved in Section 4.4.2. In many cases we will omit the proofs of the technical propositions that we need to prove soundness and completeness. This is because they are analogous in style to their respective statements presented when dealing with the may-testing case. Section 4.4.3 contains a short discussion presenting a conjecture of a possible characterisation of the $\sqsubseteq_{\text{must}}$ preorder in a non strongly-convergent setting. Finally, we provide some simple applications of our proof principle in Section 4.4.4.

4.1 Extensional Behaviour of Networks

In this Section we define the observable activities of networks. This amount to defining an alternative LTS for networks, whose actions can be detected by the external environment.

We first show that the actions of the labelled transition semantics, defined in Section 2.3 cannot be observed by external agents, or tests; then we show how a different Labelled Transition System, called the *extensional LTS*, can be defined starting from the LTS we have defined for networks. We conclude this Section by defining the weak version of extensional actions. This definition is non-standard with respect to the majority of the works which can be found in the literature [1, 34, 58]; however, our definition is justified by the fact that we aim to develop complete proof methods for the testing preorders. In sections 4.3.3 and 4.4.4 we exhibit some illuminating examples that show that completeness of our proof methods would not holds if we used a standard definition of weak extensional transitions.

Before defining formally which extensional actions can be observed by the external environment, let us provide a simple example that shows why the actions of the labelled transition semantics we have defined for networks are not suitable to this purpose.

$\mathcal{M} = \Gamma_M \rhd m[\![c!\langle v\rangle]\!] \,|\, n[\![\mathbf{0}]\!]$ $\quad$ $\mathcal{N} = \Gamma_N \rhd m[\![c!\langle v\rangle]\!] \,|\, n[\![\mathbf{0}]\!]$

Figure 4.2: Two networks $\mathcal{M}$, $\mathcal{N}$ with the same intensional behaviour but which are not testing equivalent.

Example 4.1.1. Let $\mathcal{M}$, $\mathcal{N}$ be the networks depicted in Figure 4.1. Such networks can both perform a broadcast action, which can be detected by the only output node o. However, the action that models this broadcast activity is different in $\mathcal{M}$ and $\mathcal{N}$, if we consider the Labelled Transition Semantics (or *intensional semantics*) defined in Section 2.3. In fact, $\mathcal{M} \xrightarrow{m.c!v} \Gamma_M \rhd n[\![\mathbf{0}]\!]$, while $\mathcal{M} \xrightarrow{n.c!v} \Gamma_N \mathbf{0}[\![\rhd]\!] n$. These two actions differ in the name of the node that performed the broadcast.

On the other hand, we cannot test for the name of the node which performed the broadcast action. In fact, if we place some code at o, it is straightforward to note that we can only detect the transmission of value v along channel c, but no information about the name of the node which transmitted the value is known. Therefore, we would expect $\mathcal{M} =_{\text{may}} \mathcal{N}$, and $\mathcal{M} =_{\text{must}} \mathcal{N}$. These two statements are proved formally in section 4.3.3 and 4.4.4, respectively. □

Example 4.1.1 shows that names of nodes performing a broadcast cannot be observed. On the other hand, in Example 4.1.2 we show that the names of the output nodes which are affected by a broadcast transmission play a significant role in modelling the extensional behaviour of networks.

Example 4.1.2. Consider the networks $\mathcal{M}$, $\mathcal{N}$ depicted in Figure 4.2. These networks have the same intensional behaviour; in both of them node m broadcasts value v along channel c in both of them, after which no further actions are possible. Formally we have $\mathcal{M} \xrightarrow{m.c!v} \Gamma_M \rhd m[\![\mathbf{0}]\!] \,|\, n[\![\mathbf{0}]\!]$ and $\mathcal{N} \xrightarrow{m.c!v} \Gamma_M \rhd m[\![\mathbf{0}]\!] \,|\, n[\![\mathbf{0}]\!]$.

However, the names of the output nodes which can detect the broadcast performed by node m are different in $\mathcal{M}$ and $\mathcal{N}$. In fact, in $\mathcal{M}$ the broadcast performed by node m can be detected only by the output node o_1, while in $\mathcal{N}$ it can be detected by the external node o_2. Now it is not difficult to show that $\mathcal{M} \not\sqsubseteq_{\text{may}} \mathcal{N}$ ($\mathcal{M} \not\sqsubseteq_{\text{must}} \mathcal{N}$), and $\mathcal{N} \not\sqsubseteq_{\text{may}} \mathcal{N}$ ($\mathcal{N} \not\sqsubseteq_{\text{must}} \mathcal{M}$).

Consider in fact the tests $\mathcal{T}_1 = \Gamma_T \rhd o_1[\![c?(x).\omega]\!] \,|\, o_2[\![\mathbf{0}]\!]$, and $\mathcal{T}_2 = \Gamma_T \rhd o_1[\![\mathbf{0}]\!] \,|\, o_2[\![c?(x).\omega]\!]$. It is easy to note that, for $i = 1, 2$, both $\mathcal{M} \between \mathcal{T}_i$ and $\mathcal{N} \between \mathcal{T}_i$ are defined. Further, we have that $\mathcal{M}$ **may-pass** $\mathcal{T}_1$, for we have the reduction

$$(\mathcal{M} \between \mathcal{T}_1) \twoheadrightarrow ((\Gamma_M \rhd m[\![\mathbf{0}]\!] \,|\, n[\![\mathbf{0}]\!]) \between (\Gamma_T \rhd o_1[\![\omega]\!] \,|\, o_2[\![\mathbf{0}]\!]))$$

where the right hand side of the reduction is a successful configuration.

On the other hand, the only possible reduction for $\mathcal{N} \between \mathcal{T}_1$ is given by

$$(\mathcal{N} \between \mathcal{T}_1) \twoheadrightarrow ((\Gamma_M \rhd m[\![\mathbf{0}]\!] \,|\, n[\![\mathbf{0}]\!]) \between (\Gamma_T \rhd o_1[\![c?(x).\omega]\!] \,|\, o_2[\![\mathbf{0}]\!]))$$

Since the configuration in the right hand side of the reduction above is deadlocked, this is also the only possible computation for $\mathcal{N} \between \mathcal{T}_1$, which is not successful.

For $\mathcal{M}$ **may-pass** $\mathcal{T}_1$, but $\neg(\mathcal{N}$ **may-pass** $\mathcal{T}_1)$, we have that $\mathcal{M} \not\sqsubseteq_{\text{may}} \mathcal{T}_1$. A similar argument shows that $\mathcal{N}$ **may-pass** $\mathcal{T}_2$ and $\neg(\mathcal{M}$ **may-pass** $\mathcal{T}_2)$; hence $\mathcal{N} \not\sqsubseteq_{\text{may}} \mathcal{M}$.

The tests $\mathcal{T}_1, \mathcal{T}_2$ can be used to show that $\mathcal{M} \not\sqsubseteq_{\text{must}} \mathcal{N}$ and $\mathcal{N} \not\sqsubseteq_{\text{must}} \mathcal{M}$, respectively. □

We are now ready to define the extensional behaviour of networks. This amounts to defining a set of extensional actions $\mathsf{EAct}_\tau = \mathsf{EAct} \cup \{\tau\}, \tau \notin \mathsf{EAct}$ and extensional transitions of the form $\mathcal{M} \xrightarrow{\mu} \mathcal{N}$, where $\mu \in$

EAct_τ.

The set EAct_τ includes the following actions:

Internal actions An internal action is denoted by τ, and it corresponds to some activity that cannot be detected by the external environment.

Broadcast actions A broadcast action is denoted by $c!v \triangleright \eta$, where η is a set of external node names.

Input actions An input action is denoted by $i.c?v$.

We now present the extensional semantics for networks; here the visible actions consist of activities which can be detected (hence tested) by placing code at the interface of a network. In this semantics we have internal, input and output actions.

Definition 4.1.3 (Extensional Transitions)**.** The actions of the extensional semantics are defined as follows:

(1) **internal**, $\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{N}$; some internal activity reduces the network $\mathcal{M}$ to some network $\mathcal{N}$. Here the internal activity of a network coincides either with some node performing a silent move $m.\tau$ or broadcasting a value which cannot be detected by any node in the interface of the network itself.

Formally, $(\Gamma \triangleright M) \stackrel{\tau}{\longmapsto} (\Gamma \triangleright N)$ whenever

(a) $\Gamma M \stackrel{m.\tau}{\longrightarrow} N$

(b) or $\Gamma \triangleright M \stackrel{n.c!v}{\longrightarrow} N$ for some value v, channel c and node name n satisfying $\Gamma \vdash n \to m$ implies $m \in \mathrm{nodes}(M)$

(2) **input**, $(\Gamma \triangleright M) \stackrel{i.c?v}{\longmapsto} (\Gamma \triangleright N)$; an observer placed at the input node i can send the value v along the channel c to the network $(\Gamma \triangleright M)$. For node i to be able to affect the network $\Gamma \triangleright M$ by broadcasting a value, we must have $i \in \mathsf{Input}(\Gamma \triangleright M)$.

Formally $(\Gamma \triangleright M) \stackrel{i.c?v}{\longmapsto} (\Gamma \triangleright N)$ whenever

(a) $\Gamma \triangleright M \stackrel{i.c?v}{\longrightarrow} N$

(b) $i \in \mathsf{Input}(\Gamma \triangleright M)$

(3) **output**, $(\Gamma \triangleright M) \stackrel{c!v \triangleright \eta}{\longmapsto} (\Gamma \triangleright N)$, where η is a non-empty set of nodes; an observer placed at any node $n \in \eta$ can receive the value v along the channel c. For this to happen each node $n \in \eta$ must be in $\mathsf{Output}(\Gamma \triangleright M)$, and there must be some code running at some node in M which can broadcast along channel c to each such n; that is, for any $n \in \eta, \Gamma \vdash m \to n$.

Formally, $(\Gamma \triangleright M) \stackrel{c!v \triangleright \eta}{\longmapsto} (\Gamma \triangleright N)$ whenever

(i) $(\Gamma \triangleright M) \stackrel{m.c!v}{\longrightarrow} N$ for some node m

(ii) $\eta = \{ n \in \mathsf{Output}(\Gamma \triangleright M) \mid \Gamma \vdash m \to n \} \neq \emptyset$.

These extensional transitions endow the set of testing networks with the structure of a LTS. In the future we will use the term *strong extensional transition* for them, in contrast with their weak variant, which is defined later in this Section.

Later in this Chapter we will need the following definitions: if the set of states of the LTS generated by a testing network $\mathcal{M}$ (by using extensional actions for defining the transition relation) is finite, we say that $\mathcal{M}$ is *finite state*. If every state of the LTS generated by a testing network $\mathcal{M}$ has a finite number of transitions we say that the network $\mathcal{M}$ is *finite branching*. Finally, a testing network $\mathcal{M}$ is *finitary* if it is both finite state and finite branching.

Next we show that there is a close relationship between the extensional actions of Definition 4.1.3 and the reduction relation defined in Section 2.2.

Proposition 4.1.4. Let $\mathcal{M}, \mathcal{N}$ be networks;

1. If $\mathcal{M} \twoheadrightarrow \mathcal{N}$ then either
   - $\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{N}'$, for some $\mathcal{N}'$ such that $\mathcal{N}' \equiv \mathcal{N}$, or
   - $\mathcal{M} \stackrel{c!v \triangleright \eta}{\longmapsto} \mathcal{N}'$, for some channel c, value v, non-empty set of nodes η and networks $\mathcal{N}'$ such that $\mathcal{N}' \equiv \mathcal{N}$
2. If $\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{N}$ then $\mathcal{M} \twoheadrightarrow \mathcal{N}$
3. If $\mathcal{M} \stackrel{c!v \triangleright \eta}{\longmapsto} \mathcal{N}$ for some channel c, value v and non-empty set of nodes η, then $\mathcal{M} \twoheadrightarrow \mathcal{N}$.

Proof. All the statements are a direct consequence of the Harmony Theorem, Theorem 2.4.10 and the definition of the extensional transitions. Detailed proofs are provided in Appendix A, Section A.3 □

Now we turn our attention at the definition of weak extensional actions; as we have already mentioned, these are needed to abstract from internal activities of networks, which cannot be detected by the external environment. Weak extensional transitions have been employed in many process calculi, such as *CCS* [1], the *π-calculus* [58] and many others; however, the definition of weak extensional transitions we provide for testing networks is non-standard, in the sense that it differs from those provided in the works cited above.

Definition 4.1.5 (Weak Extensional Transitions)**.** (1) $\mathcal{M} \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{N}$ whenever $\mathcal{M} \stackrel{\tau}{\longmapsto}{}^{*} \mathcal{N}$

(2) $\mathcal{M} \stackrel{n.c?v}{\models\!\Longrightarrow} \mathcal{N}$ whenever $\mathcal{M} \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{M}' \stackrel{n.c?v}{\longmapsto} \mathcal{N}' \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{N}$ for some networks $\mathcal{M}', \mathcal{N}'$

(3) Let $\mathcal{M} \stackrel{c!v \triangleright \eta}{\models\!\Longrightarrow} \mathcal{N}$ be the least relation satisfying

(a) $\mathcal{M} \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{M}' \stackrel{c!v \triangleright \eta}{\longmapsto} \mathcal{N}' \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{N}$ for some testing networks $\mathcal{M}', \mathcal{N}'$ implies $\mathcal{M} \stackrel{c!v \triangleright \eta}{\models\!\Longrightarrow} \mathcal{N}$

(b) $\mathcal{M} \stackrel{c!v \triangleright \eta_1}{\models\!\Longrightarrow} \mathcal{M}'$, $\mathcal{M}' \stackrel{c!v \triangleright \eta_2}{\models\!\Longrightarrow} \mathcal{N}$, where $\eta_1 \cap \eta_2 = \emptyset$, implies $\mathcal{M} \stackrel{c!v \triangleright (\eta_1 \cup \eta_2)}{\models\!\Longrightarrow} \mathcal{N}$

These weak transitions endow the set of networks Nets with the structure of another LTS, called the *extensional LTS* and denoted by $\mathsf{LTS}_{\mathsf{Nets}}$.

Some explanations are necessary for the non-standard definition of output actions in Definition 4.1.5(3b) Informally speaking, the definition of weak extensional output transitions expresses the capability of simulating broadcast through multicast; that is, a single (weak) broadcast transition detected by a set of nodes η can be matched by a sequence of broadcast transitions (possibly interrupted by internal actions), detected respectively by $\eta_1, \cdots, \eta_i \subseteq \eta$, provided that the collection $\{\eta_1, \cdots, \eta_i\}$ is a partition of η. This constraint is needed to ensure that

(i) every node in η will detect the transmitted value and

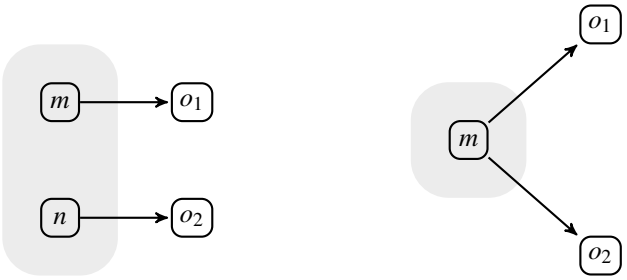
(ii) no node in η will detect the value more than once. This is ensured since, in our definition of weak extensional outputs, a transition $\mathcal{M} \stackrel{c!v \triangleright \eta_1 \cup \eta_2}{\models\!\Longrightarrow} \mathcal{N}$ can be derived from two (weak) transitions $\mathcal{M} \stackrel{c!v \triangleright \eta_1}{\models\!\Longrightarrow} \mathcal{M}'$ and $\mathcal{M}' \stackrel{c!v \triangleright \eta_2}{\models\!\Longrightarrow} \mathcal{N}$ only in the case that $\eta_1 \cap \eta_2 = \emptyset$.

Example 4.1.6 (Broadcast versus Multicast)**.** Consider the network $\mathcal{M}$ in Figure 4.3. This network can perform two different broadcasts of value v along channel c; the one performed by node m, can be detected by the output node o_1; the other, performed by node n, can be detected by the external node o_2. From the point of view of extensional transitions, we have the sequence

$$\mathcal{M} \stackrel{c!v \triangleright \{o_1\}}{\longmapsto} \mathcal{M}'' \stackrel{c!v \triangleright \{o_2\}}{\longmapsto} \mathcal{M}'$$

where $\mathcal{M}'' = \Gamma_M \triangleright m[\![\mathbf{0}]\!] \,|\, n[\![c!\langle v \rangle]\!]$, $\mathcal{M}' = \Gamma_M \triangleright m[\![\mathbf{0}]\!] \,|\, n[\![\mathbf{0}]\!]$.

This sequence of transitions ensures that, in the extensional semantics, we have $\mathcal{M} \stackrel{c!v \triangleright \{o_1, o_2\}}{\models\!\Longrightarrow} \mathcal{M}'$. Note that, if weak extensional actions were defined in a standard way, the transition above for network $\mathcal{M}$ could not have been derived.



$\mathcal{M} = \Gamma_M \rhd m[\![c!\langle v\rangle.\mathbf{0}]\!] \mid n[\![c!\langle v\rangle.\mathbf{0}]\!]$ $\quad\quad$ $\mathcal{N} = \Gamma_N \rhd m[\![c!\langle v\rangle]\!]$

Figure 4.3: Broadcast versus Multicast

Consider now the network $\mathcal{N}$, again in Figure 4.3. For this network, we have a single broadcast of value v along channel c, which can be detected by both the output nodes o_1 and o_2. Thus, in the extensional semantics, we have the transition $\mathcal{N} \xmapsto{c!v\rhd\{o_1,o_2\}} \mathcal{N}'$, where $\mathcal{N}' = \Gamma_N \rhd m[\![\mathbf{0}]\!]$.

Intuitively speaking, the non-standard definition of extensional broadcast action ensures that the behaviour of $\mathcal{N}$ can be replicated by network $\mathcal{M}$. In fact, for any testing network $\mathcal{T}$ such that $\mathcal{M} \between \mathcal{T}$ and $\mathcal{N} \between \mathcal{T}$ are defined, it is not difficult to show that if $\mathcal{N} \between \mathcal{T} \rightarrow^* \mathcal{N}' \between \mathcal{T}'$, then there exists a test $\mathcal{T}''$ such that $\mathcal{M} \between \mathcal{T} \rightarrow^* \mathcal{M}'' \between \mathcal{T}'' \rightarrow^* \mathcal{M}' \between \mathcal{T}'$.

This argument leads to the intuition that $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$, which is proved formally in Section 4.3.3 where we will revisit this Example. Further, we have that $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$, which is proved formally in Section 4.4.4. □

Remark 4.1.7 (Inductive Principle For Weak Extensional Output Actions)**.** Note that we are able to properties for weak transitions of the form $\mathcal{M} \overset{c!v\rhd\eta}{\models\!\Longrightarrow} \mathcal{N}$ by proceeding by induction on the structure of the proof of its derivation; specifically, if we want to prove that a property $\mathscr{P}$ holds for any transition of the form $\mathcal{M} \overset{c!v\rhd\eta}{\models\!\Longrightarrow} \mathcal{N}$, where c, v are fixed, then it suffices to show that

- For any set of nodes $\eta \neq \emptyset$ and networks $\mathcal{M}, \mathcal{N}$, if $\mathcal{M} \overset{c!v\rhd\eta}{\models\!\Longrightarrow} \mathcal{N}$ because $\mathcal{M} \overset{\tau}{\models\!\Longrightarrow} \xmapsto{c!v\rhd\eta} \overset{\tau}{\models\!\Longrightarrow} \mathcal{N}$, then $\mathscr{P}(\mathcal{M} \overset{c!v\rhd\eta}{\models\!\Longrightarrow} \mathcal{N})$
- if $\mathscr{P}(\mathcal{M} \overset{c!v\rhd\eta_1}{\models\!\Longrightarrow} \mathcal{L})$, $\mathscr{P}(\mathcal{L} \overset{c!v\rhd\eta_2}{\models\!\Longrightarrow} \mathcal{N})$, and $\eta_1 \cap \eta_2 = \emptyset$, then $\mathscr{P}(\mathcal{M} \overset{c!v\rhd(\eta_1\cup\eta_2)}{\models\!\Longrightarrow} \mathcal{N}|)$

□

The induction principle of Remark 4.1.7 allows us to prove that a weak extensional output transition is a sequence of (strong) extensional outputs, possibly interleaved by internal transitions. This property will be useful later.

Proposition 4.1.8. Let $\mathcal{M}, \mathcal{N}$ be testing networks. Then $\mathcal{M} \overset{c!v\rhd\eta}{\models\!\Longrightarrow} \mathcal{N}$ for some channel c, value v, and non-empty set of nodes η if and only if there exist an index $k \geq 1$ and a collection of pairwise disjoint, non-empty sets of nodes $\{\eta_j\}_{j=1}^k$, such that

- $\bigcup_{j=1}^k \eta_i = \eta$
- $\mathcal{M} \overset{\tau}{\models\!\Longrightarrow} \xmapsto{c!v\rhd\eta_1} \overset{\tau}{\models\!\Longrightarrow} \cdots \overset{\tau}{\models\!\Longrightarrow} \xmapsto{c!v\rhd\eta_k} \overset{\tau}{\models\!\Longrightarrow} \mathcal{N}$

Proof. See the Appendix. □

Another property that we will need later, and which can be proved by exploiting the induction principle of Remark 4.1.7, is the following.

Proposition 4.1.9. *Let $\mathcal{M}, \mathcal{N}$ be two testing networks such that $\mathcal{M} \overset{c!v\rhd\eta}{\models\!\Longrightarrow} \mathcal{N}$. Then $\mathcal{M} \rightarrow^\star \mathcal{N}$.*

$\mathcal{M} = \Gamma_M \rhd m[\![c!\langle v\rangle]\!]$ $\quad\quad$ $\mathcal{N} = \Gamma_N \rhd n[\![c?(x).P]\!]$

Figure 4.4: Two networks $\mathcal{M}$ and $\mathcal{N}$; here the extensional behaviour of the network $\mathcal{M} \natural \mathcal{N}$ cannot be inferred by those of its individual components.

Proof. We first remark that if $\mathcal{M} \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{M}'$ for some network $\mathcal{M}'$, then $\mathcal{M} \rightarrowtriangle^* \mathcal{M}'$. This can be proved by performing a natural induction on the number of (strong) τ-transitions required to infer the transition $\mathcal{M} \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{M}'$, and by using Proposition 4.1.4.

Now we prove that if $\mathcal{M} \stackrel{c!v\rhd\eta}{\models\!\Longrightarrow} \mathcal{N}$ then $\mathcal{M} \rightarrowtriangle^\star \eta$ by induction on the proof of this transition.

- $\mathcal{M} \stackrel{\tau}{\models\!\Longrightarrow} \stackrel{c!v\rhd\eta}{\longmapsto} \stackrel{\tau}{\models\!\Longrightarrow} \mathcal{N}$. By Proposition 4.1.4, and by the remark at the beginning of the proof, it follows that $\mathcal{M} \rightarrowtriangle^* \rightarrowtriangle \rightarrowtriangle^* \mathcal{N}$, hence $\mathcal{M} \rightarrowtriangle^* \mathcal{N}$
- $\mathcal{M} \stackrel{c!v\rhd\eta_1}{\models\!\Longrightarrow} \stackrel{c!v\rhd\eta_2}{\models\!\Longrightarrow} \mathcal{N}$, for some sets of nodes η_1, η_2 such that $\eta_1 \cap \eta_2 = \emptyset$, $\eta_1 \cup \eta_2 = \eta$.

  By the inductive hypothesis it follows that $\mathcal{M} \rightarrowtriangle^* \rightarrowtriangle^* \mathcal{N}$, hence $\mathcal{M} \rightarrowtriangle^\star \mathcal{N}$.

□

4.2 Composition and Decomposition Results for Extensional Actions

We prove compositional and decompositional properties for extensional actions of testing networks.

Specifically, given two networks $\mathcal{M}, \mathcal{N}$ and a composition operator $|||_{\mathscr{P}}$ such that $\mathcal{M} |||_{\mathscr{P}} \mathcal{N}$ is defined, we want to be able to infer a transition of the form $(\mathcal{M} |||_{\mathscr{P}} \mathcal{N}) \stackrel{\mu}{\longmapsto} (\mathcal{M}' |||_{\mathscr{P}} \mathcal{N}')$, starting from transitions for the individual networks $\mathcal{M} \stackrel{\mu_1}{\longmapsto} \mathcal{M}'$ and $\mathcal{N} \stackrel{\mu_2}{\longmapsto} \mathcal{N}'$. Conversely, given a transition of the form $(\mathcal{M} |||_{\mathscr{P}} \mathcal{N}) \stackrel{mu}{\longmapsto} \mathcal{L}$, we want to determine two extensional transitions μ_1 and μ_2 such that $\mathcal{M} \stackrel{\mu_1}{\longmapsto} \mathcal{M}'$ and $\mathcal{N} \stackrel{\mu_2}{\longmapsto} \mathcal{N}'$, and $\mathcal{L} = \mathcal{M}' |||_{\mathscr{P}} \mathcal{N}'$.

Unfortunately, we are not able to solve this problem for the extension operator $\natural$, as the following Example shows.

Example 4.2.1. Consider the networks $\mathcal{M}, \mathcal{N}$ depicted in Figure 4.4. Here it is easy to show that $\mathcal{M} \natural \mathcal{N}$ is defined, and it is equipped with an extensional transition of the form $(\mathcal{M} \natural \mathcal{N}) \stackrel{\tau}{\longmapsto} (\mathcal{M}' \natural \mathcal{N}')$, where $\mathcal{M}' = \Gamma_M \rhd m[\![\mathbf{0}]\!]$ and $\mathcal{N}' = \Gamma_N \rhd n[\![\{v/x\}P]\!]$.

However, this transition cannot be inferred by looking at the extensional behaviour of the individual networks $\mathcal{M}, \mathcal{N}$. In particular, there is no possible (extensional) transition for the network $\mathcal{N}$, for there the only node n it is waiting to receive some value along channel c, which is not connected to any other node. □

The problem discussed in Example 4.2.1 has its roots in the asymmetric definition of the extension operator $\natural$. This operator, in fact, only preserves the connections of the left hand side network in a composition. For the network on the right hand side, however, new connections of its internal nodes can appear in the composed network. For example, in Example 4.2.1 node n in network $\mathcal{N}$ is not connected to any node, but we have the connection $m \rightarrow n$ in the composed network $\mathcal{M} \natural \mathcal{N}$.

Luckily, we are able to prove compositional and decompositional results for the symmetric composition operator $\|$, Definition 3.1.6. While we have already shown that this operator cannot be used to define compositional testing preorders, Example 3.3.8, it is strictly related to the extension operator $\natural$; in particular, we will see that it is always possible to rewrite a well-defined network of the form $\mathcal{M} \natural \mathcal{N}$ using the symmetric composition operator $\|$, that is $(\mathcal{M} \natural \mathcal{N}) = (\mathcal{M} \| \mathcal{N}')$ for some network $\mathcal{N}'$. Conversely, every network of the

form $(\mathcal{M} \parallel \mathcal{N})$ can be rewritten using the extension operator, that is $(\mathcal{M} \parallel \mathcal{N}) = (\mathcal{M} \between \mathcal{N}'')$ for some network $\mathcal{N}''$. Since the operator $\parallel$ is commutative, it can be used to prove decomposition and composition results for extensional transitions, which are stated formally in sections 4.2.1 and 4.2.2, respectively.

Now we present formally how it is possible to switch from using from the extension operator $\between$ to the composition operator $\parallel$, and vice versa.

Definition 4.2.2. Let $\mathcal{M} = \Gamma_M \rhd M, \mathcal{N} = \Gamma_N \rhd N$ two testing networks such that $\mathrm{nodes}(M) \cap \mathrm{nodes}(N) = \emptyset$.

The *symmetric counterpart* of $\mathcal{N}$ with respect to $\mathcal{M}$ is the network $\mathrm{sym}_{\mathcal{M}}(\mathcal{N})$, which is defined as $\Gamma'_N \rhd N$, where

$$(\Gamma'_N)_V = (\Gamma_N)_V \cup \{m : \Gamma_M \vdash m \rightleftarrows n \text{ for some } n \in \mathrm{nodes}(N)\} \tag{4.1}$$

$$(\Gamma'_N)_E = (\Gamma_N)_E \cup \{(m,n) \mid n \in \mathrm{nodes}(N), \Gamma_M \vdash m \to n\} \cup \{(m,n) \mid m \in \mathrm{nodes}(N), \Gamma_M \vdash m \to n\} \tag{4.2}$$

The *extension counterpart* of $\mathcal{N}$ with respect to $\mathcal{M}$ is the network $\mathrm{ext}_{\mathcal{M}}(\mathcal{N})$, which is defined as $\Gamma''_N \rhd N$, where

$$(\Gamma''_N)_V = (\Gamma_N)_V \setminus \mathrm{nodes}(M) \tag{4.3}$$

$$(\Gamma''_N)_E = \{(m,n) \mid (\Gamma''_N)_V \vdash m, n, \Gamma_N \vdash m \to n\} \tag{4.4}$$

□

Intuitively, for any network $\mathcal{M}, \mathcal{N}$, the network $(\mathcal{M} \between \mathcal{N})$ (if defined) is equal to $(\mathcal{M} \parallel \mathrm{sym}_{\mathcal{M}}(\mathcal{N}))$. Conversely, if $(\mathcal{M} \parallel \mathcal{N})$ is defined, then it is equal to $(\mathcal{M} \parallel \mathrm{ext}_{\mathcal{M}}(\mathcal{N}))$. Before proving this statement, we perform some sanity checks for Definition 4.2.2. First we prove that, the operators $\mathrm{sym}_{\mathcal{M}}(\mathcal{N})$ and $\mathrm{ext}_{\mathcal{M}}(\mathcal{N})$ is closed with respect to the set CNets, under the assumption that $\mathrm{nodes}(\mathcal{M}) \cap \mathrm{nodes}(\mathcal{N}) = \emptyset$.

Proposition 4.2.3. Let $\mathcal{M}, \mathcal{N} \in \mathsf{CNets}$ be two networks such that that $\mathrm{nodes}(\mathcal{M}) \cap \mathrm{nodes}(\mathcal{N}) = \emptyset$. Then

(i) $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \in \mathsf{CNets}$

(ii) $\mathrm{ext}_{\mathcal{M}}(\mathcal{N}) \in \mathsf{CNets}$

Proof. See Appendix A, Section A.3. □

Next, we perform another sanity check for Definition 4.2.2. Informally speaking, the operations $\mathrm{sym}_{\mathcal{M}}(\mathcal{N})$ and $\mathrm{ext}_{\mathcal{M}}(\mathcal{N})$ can be inverted.

Proposition 4.2.4.

(i) If $\mathcal{M} = \Gamma_M \rhd M$ and $\mathcal{N} = \Gamma_N \rhd N$ are two networks such that $(\Gamma_N)_V \cap \mathrm{nodes}(M) = \emptyset$, then $\mathrm{ext}_{\mathcal{M}}(\mathrm{sym}_{\mathcal{M}}(\mathcal{N})) = \mathcal{N}$.

(ii) If $\mathcal{M} = \Gamma_M \rhd M$ and $\mathcal{N} = \Gamma_N \rhd N$ are two networks such that $\mathscr{P}_s(\mathcal{M}, \mathcal{N}) = \mathrm{true}$ then $\mathrm{sym}_{\mathcal{M}}(\mathrm{ext}_{\mathcal{M}}(\mathcal{N})) = \mathcal{N}$. Recall that $\mathscr{P}_s$ is the consistency predicate for the composition operator $\parallel$, and it is defined in Definition 3.1.6.

Proof. See the Appendix. □

Now that we have performed some sanity checks on the operators of Definition 4.2.2, we can show that they can be used to switch the composition operators used for composing networks.

Proposition 4.2.5. Let $\mathcal{M}, \mathcal{N}$ be two testing networks. Then

(i) $\mathcal{M} \parallel \mathcal{N} \approx \mathcal{M} \between \mathrm{ext}_{\mathcal{M}}(\mathcal{N})$,

(ii) $\mathcal{M} \between \mathcal{N} \approx \mathcal{M} \parallel \mathrm{sym}_{\mathcal{M}}(\mathcal{N})$.

We recall that $\approx$ is Kleene's equality.

Proof. The two statements are proved separately. In both the proofs, we let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$.

1. By definition 4.2.2 we know that $\mathrm{ext}_{\mathcal{M}}(\mathcal{N}) = \Gamma'_N \rhd N$, where Γ'_N is defined by equations (4.3) and (4.4).

   Suppose that $\mathcal{M} \parallel \mathcal{N}$ is defined; then we have that $\mathscr{P}_s(\mathcal{M}, \mathcal{N}) = \mathsf{true}$. Recall that $\mathscr{P}_s$ is the consistency predicate for the composition operator $\parallel$, Definition 3.1.6.

   By Equation (4.3) we have that $(\Gamma'_N)_V = (\Gamma_N)_V \setminus \mathrm{nodes}(M)$; hence it follows that $(\Gamma'_N)_V \cap \mathrm{nodes}(M) = \emptyset$. This ensures that $(\mathcal{M} \between \mathrm{ext}_{\mathcal{M}}(\mathcal{N}))$ is defined.

   For networks $\Gamma_M \rhd M$, $\Gamma_N \rhd N$ and $\Gamma'_N \rhd N$ we have that $(\Gamma_M \rhd M) \parallel (\Gamma_N \rhd N) = (\Gamma_M \cup \Gamma_N) \rhd (M \mid N)$, $(\Gamma_M \rhd M) \between (\Gamma'_N \rhd N) = (\Gamma_M \cup \Gamma'_N) \rhd (M \mid N)$. Thus, in order to prove that $\mathcal{M} \parallel \mathcal{N} = \mathcal{M} \between \mathrm{ext}_{\mathcal{M}}(\mathcal{N})$, it is sufficient to prove that $(\Gamma_M \cup \Gamma_N) = (\Gamma_M \cup \Gamma'_N)$.

   Note that the inclusions $(\Gamma'_N)_V \subseteq (\Gamma_N)_V$ and $(\Gamma'_N)_E \subseteq (\Gamma_N)_E$ are an immediate consequence of equations (4.3) and (4.4), respectively. Then it is straightforward to show that $(\Gamma_M \cup \Gamma'_N)_V \subseteq (\Gamma_M \cup \Gamma_N)_V$ and $(\Gamma_M \cup \Gamma'_N)_E \subseteq (\Gamma_M \cup \Gamma_N)_E$. It remains to prove that $(\Gamma_M \cup \Gamma_N)_V \subseteq (\Gamma_M \cup \Gamma'_N)_V$, and $(\Gamma_M \cup \Gamma_N)_E \subseteq (\Gamma_M \cup \Gamma'_N)_E$. These two statements are proved separately.

   - First, suppose that $(\Gamma_M \cup \Gamma_N) \vdash m$ for some node m; we need to show that $(\Gamma_M \cup \Gamma'_N) \vdash m$. In this case, either $\Gamma_M \vdash m$, from which it is trivial to note that $(\Gamma_M \cup \Gamma'_N) \vdash m$, or $\Gamma_N \vdash m$. In the last case either $m \in \mathrm{nodes}(N)$ or $m \in \mathsf{Int}(\Gamma_N \rhd N)$.

     If $m \in \mathrm{nodes}(N)$, we have that $m \notin \mathrm{nodes}(M)$. This is because we are assuming that $\mathscr{P}_s(\mathcal{M}, \mathcal{N}) = \mathsf{true}$. By Equation (4.3) it follows that $\Gamma'_N \vdash m$, since $\Gamma_N \vdash m$ and $m \notin \mathrm{nodes}(M)$. Now it is immediate to show that $(\Gamma_M \cup \Gamma'_N) \vdash m$.

     If $m \in \mathsf{Int}(\Gamma_N \rhd N)$, then there exists a node $n \in \mathrm{nodes}(N)$ such that $\Gamma_N \vdash m \rightleftarrows n$. This is because we are assuming that $\Gamma_N \rhd N \in \mathsf{CNets}$. We perform a case analysis on node m; for this node, in fact, we have that either $m \in \mathrm{nodes}(M)$ or $m \notin \mathrm{nodes}(M)$.

     If $m \in \mathrm{nodes}(M)$ then $\Gamma_M \vdash m \to n$. This is because $\mathscr{P}_s(\mathcal{M}, \mathcal{N}) = \mathsf{true}$ by hypothesis, and by noticing that $\Gamma_N \vdash m \to n$ with $m \in \mathrm{nodes}(M)$. Since $\Gamma_M \vdash m \to n$, we have $\Gamma_M \vdash m$, hence $(\Gamma_M \cup \Gamma'_N) \vdash m$.

     If $m \notin \mathrm{nodes}(M)$, then by Equation (4.3) it follows that $\Gamma'_N \vdash m$, hence $(\Gamma_M \cup \Gamma'_N) \vdash m$.

   - Now suppose that $(\Gamma_M \cup \Gamma_N) \vdash m \to n$. We need to show that $(\Gamma_M \cup \Gamma'_N) \vdash m \to n$.

     In this case either $\Gamma_M \vdash m \to n$, in which case the proof of $(\Gamma_M \cup \Gamma'_N) \vdash m \to n$ is trivial, or $\Gamma_N \vdash m \to n$.

     We have two possibilities; the first one is that $m, n \notin \mathrm{nodes}(M)$, in which case Equation (4.3) ensures that $\Gamma'_N \vdash m, n$, then we obtain $\Gamma'_N \vdash m \to n$ from Equation (4.4). Now it is trivial to show that $(\Gamma_M \cup \Gamma'_N) \vdash m \to n$.

     The other possibility is that either $m \in \mathrm{nodes}(M)$ or $n \in \mathrm{nodes}(N)$. In both cases, the hypothesis $\mathscr{P}_s(\mathcal{M}, \mathcal{N}) = \mathsf{true}$ ensures that $\Gamma_M \vdash m \to n$, hence $(\Gamma_M \cup \Gamma'_N) \vdash m \to n$.

2. By definition (4.2.2) we know that $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) = \Gamma'_N \rhd N$, where Γ'_N is defined from Γ_N using equations (4.1) and (4.2).

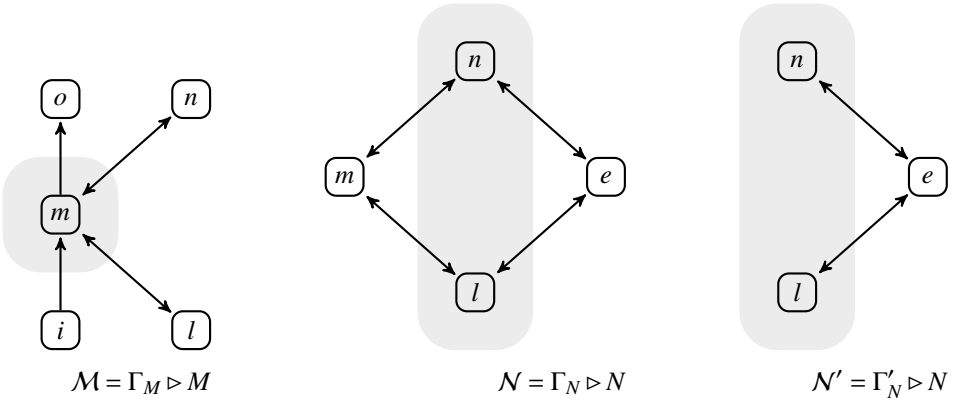
   Suppose $\mathcal{M} \between \mathcal{N}$ is defined. Then it holds that $(\Gamma_N)_V \cap \mathrm{nodes}(M) = \emptyset$. We first show that $\mathscr{P}_s(\mathcal{M}, \mathrm{sym}_{\mathcal{M}}(\mathcal{N})) = \mathsf{true}$, thus showing that $\mathcal{M} \parallel \mathrm{sym}_{\mathcal{M}}(\mathcal{N})$ is defined as well. To this end, we need to show the following:

   - $\mathrm{nodes}(M) \cap \mathrm{nodes}(N) = \emptyset$.

     This is trivial; for $\mathcal{N} \in \mathsf{CNets}$, we have that $\mathrm{nodes}(N) \subseteq (\Gamma_N)_V$, hence $\mathrm{nodes}(M) \cap \mathrm{nodes}(N) = \emptyset$ since we are assuming that $\mathrm{nodes}(M) \cap (\Gamma_N)_V = \emptyset$.

   - If $\Gamma_M \vdash m \to n$ ($\Gamma_M \vdash m \leftarrow n$) for some node m, n with $m \in \mathrm{nodes}(N)$, then $\Gamma'_N \vdash m \to n$ ($\Gamma'_N \vdash m \leftarrow n$).

     This statement follows immediately from Equation (4.2).



$\mathcal{M} = \Gamma_M \rhd M$ $\quad \mathcal{N} = \Gamma_N \rhd N$ $\quad \mathcal{N}' = \Gamma'_N \rhd N$

Figure 4.5: Three networks $\mathcal{M}, \mathcal{N}$ and $\mathcal{N}'$, such that $\mathcal{N} = \mathrm{sym}_{\mathcal{M}}(\mathcal{N}')$, $\mathcal{N}' = \mathrm{ext}_{\mathcal{M}}(\mathcal{N})$.

- Suppose that $\Gamma'_N \vdash m \to n$ ($\Gamma'_N \vdash m \leftarrow n$) for some $m \in \mathrm{nodes}(M)$. Then $\Gamma_M \vdash m \to n$ ($\Gamma_M \vdash m \leftarrow n$). If $m \in \mathrm{nodes}(M)$, then we have that $\Gamma_N \nvdash m$. In fact, by hypothesis it holds that $\mathrm{nodes}(M) \cap (\Gamma_N)_V = \emptyset$. Thus, it is not possible to have $\Gamma_N \vdash m \to n$ ($\Gamma_N \vdash m \leftarrow n$). Since $\Gamma'_N \vdash m \to n$ ($\Gamma'_N \vdash m \leftarrow n$), $\Gamma_N \vdash m \nrightarrow n$ ($\Gamma_N \vdash m \nleftarrow n$), Equation (4.2) ensures that $\Gamma_M \vdash m \to n$ ($\Gamma_M \vdash m \leftarrow n$), as we wanted to prove.

It remains to show that $(\mathcal{M} \nparallel \mathcal{N}) = (\mathcal{M} \parallel \mathrm{sym}_{\mathcal{M}}(\mathcal{N}))$. For this purpose, it is sufficient to prove that $(\Gamma_M \cup \Gamma_N) = (\Gamma_M \cup \Gamma'_N)$. Note that the inclusions $(\Gamma_M \cup \Gamma_N)_V \subseteq (\Gamma_M \cup \Gamma'_N)_V$ and $(\Gamma_M \cup \Gamma_N)_E \subseteq (\Gamma_M \cup \Gamma'_N)_E$ are an immediate consequence of Equations (4.1) and (4.2), respectively. Thus we only need to prove that $(\Gamma_M \cup \Gamma'_N)_V \subseteq (\Gamma_M \cup \Gamma_N)_V$, $(\Gamma_M \cup \Gamma'_N)_E \subseteq (\Gamma_M \cup \Gamma_N)_E$.

- Suppose $(\Gamma_M \cup \Gamma'_N) \vdash m$ for some node m. We want to show that $(\Gamma_M \cup \Gamma_N) \vdash m$.
  Since $(\Gamma_M \cup \Gamma'_N) \vdash m$, then either $\Gamma_M \vdash m$, from which it follows immediately that $(\Gamma_M \cup \Gamma_N) \vdash m$, or $\Gamma'_N \vdash m$.
  In this last case, Equation (4.1) ensures that either $\Gamma_N \vdash m$, from which it follows $(\Gamma_M \cup \Gamma_N) \vdash m$, or $\Gamma_M \vdash m \to n$ for some $n \in \mathrm{nodes}(N)$; here it follows that $\Gamma_M \vdash m$, hence $(\Gamma_M \cup \Gamma_N) \vdash m$.
- Suppose $(\Gamma_M \cup \Gamma'_N) \vdash m \to n$ for some nodes m, n. We show that $(\Gamma_M \cup \Gamma_N) \vdash m \to n$.
  If $(\Gamma_M \cup \Gamma'_N) \vdash m \to n$ because $\Gamma_M \vdash m \to n$ then the statement is trivial to prove. Otherwise, $(\Gamma_M \cup \Gamma'_N) \vdash m \to n$ because $\Gamma'_N \vdash m \to n$.
  In this last case, by Equation (4.2) either $\Gamma_N \vdash m \to n$, from which $(\Gamma_M \cup \Gamma_N) \vdash m \to n$ follows immediately, or $\Gamma_M \vdash m \to n$ where either $m \in \mathrm{nodes}(N)$ or $n \in \mathrm{nodes}(N)$; here again it is trivial to prove that $(\Gamma_M \cup \Gamma_N) \vdash m \to n$.

□

Example 4.2.6 (Change of connectivity graphs)**.** Consider the networks $\mathcal{M}, \mathcal{N}$ and $\mathcal{N}'$ depicted in Figure 4.5.

Here it is easy to show that $\mathcal{N}' = \mathrm{ext}_{\mathcal{M}}(\mathcal{N})$; conversely, $\mathcal{N} = \mathrm{sym}_{\mathcal{M}}(\mathcal{N}')$. In practice, $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}')$ is defined by adding to $\mathcal{N}$ the connections between internal nodes of $\mathcal{M}$ and internal nodes of $\mathcal{N}$ which are defined in the former network. On the other hand, $\mathrm{ext}_{\mathcal{M}}(\mathcal{N})$ is defined by removing in $\mathcal{N}$ all the nodes that are internal in $\mathcal{M}$, together with the associated connections.

This ensures that $(\mathcal{M} \parallel \mathcal{N})$ and $(\mathcal{M} \nparallel \mathcal{N}')$ are defined; further, by Proposition 4.2.5 they are also equivalent.
□

Note that the definition of the operators $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}), \mathrm{ext}_{\mathcal{M}}(\mathcal{N})$ depends on the connectivity graph and the internal nodes of $\mathcal{M}$, but not on the code that its nodes are running. That is, we expect that if $\mathcal{M} = \Gamma_M \rhd M, \mathcal{M}' = \Gamma_M \rhd M'$ and $\mathrm{nodes}(M) = \mathrm{nodes}(M')$, then $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) = \mathrm{sym}_{\mathcal{M}'}(\mathcal{N})$, $\mathrm{ext}_{\mathcal{M}}(\mathcal{N}) = \mathrm{ext}_{\mathcal{M}'}(\mathcal{N})$.

In practice the only useful application of this fact is given by the following.

Fact 4.2.7. If $\mathcal{M} \stackrel{\mu}{\longmapsto} \mathcal{M}'$, then for any network $\mathcal{N}$ it holds $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) = \mathrm{sym}_{\mathcal{M}'}(\mathcal{N})$, $\mathrm{ext}_{\mathcal{M}}(\mathcal{N}) = \mathrm{ext}_{\mathcal{M}'}(\mathcal{N})$.

If $(\mathcal{M} \between \mathcal{N}) \stackrel{\mu}{\longmapsto} (\mathcal{M}' \between \mathcal{N}')$, then $(\mathcal{M} \parallel \mathrm{sym}_{\mathcal{M}}(\mathcal{N})) \stackrel{\mu}{\longmapsto} (\mathcal{M}' \parallel \mathrm{sym}_{\mathcal{M}}(\mathcal{N}'))$. If $\mathcal{M} \parallel \mathcal{N} \stackrel{\mu}{\longmapsto} (\mathcal{M}' \parallel \mathcal{N}')$ then $(\mathcal{M} \between \mathrm{ext}_{\mathcal{M}}(\mathcal{N})) \stackrel{\mu}{\longmapsto} (\mathcal{M} \between \mathrm{ext}_{\mathcal{M}}(\mathcal{N}'))$. □

Remark 4.2.8. Let $\mathcal{M}$ be a network and define the sets $\mathscr{N} = \{\mathcal{N} \mid \mathcal{M} \between \mathcal{N} \text{ is defined }\}$, $\mathscr{L} = \{\mathcal{L} \mid \mathcal{M} \parallel \mathcal{L} \text{ is defined }\}$. It is possible to prove, by straightforward calculations, that the operator $\mathrm{sym}_{\mathcal{M}}(\cdot)$ is an homomorphism from $\langle \mathscr{N}, \between \rangle$ to $\langle \mathscr{L}, \between \rangle$. That is, whenever $\mathrm{sym}_{\mathcal{M}}(\mathcal{N} \between \mathcal{N}')$ is defined, then so are $\mathrm{sym}_{\mathcal{M}}(\mathcal{N})$, $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}')$; further we have that $\mathrm{sym}_{\mathcal{M}}(\mathcal{N} \between \mathcal{N}') = \mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \between \mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \between \mathrm{sym}_{\mathcal{M}}(\mathcal{N}')$. Similarly, we can show that $\mathrm{ext}_{\mathcal{M}}(\cdot)$ is an homomorphism from $\langle \mathscr{L}, \between \rangle$ to $\langle \mathscr{N}, \between \rangle$.

Now define a partial order (modulo structural equivalence) over networks by letting $\mathcal{N} \leq \mathcal{N}'$ if there exists a network $\mathcal{N}''$ such that $\mathcal{N} \between \mathcal{N}'' \equiv \mathcal{N}'$. Let $\mathcal{N} \in \mathscr{N}$, $\mathcal{L} \in \mathscr{L}$ be two networks such that $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \leq \mathcal{L}$. Note that $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \in \mathscr{L}$ because of Proposition 4.2.5. By definition there exists a network $\mathcal{N}'$ such that $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \between \mathcal{N}' \approx \mathcal{L}$; by straightworward calculations we obtain that

$$\begin{aligned}
\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \between \mathcal{N}' &\approx \mathcal{L} \\
\mathrm{ext}_{\mathcal{M}}(\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \between \mathcal{N}') &\approx \mathrm{ext}_{\mathcal{M}}(\mathcal{L}) \\
\mathrm{ext}_{\mathcal{M}}(\mathrm{sym}_{\mathcal{M}}(\mathcal{N})) \between \mathrm{ext}_{\mathcal{M}}(\mathcal{N}') &\approx \mathrm{ext}_{\mathcal{M}}(\mathcal{L}) \\
\mathcal{N} \between \mathrm{ext}_{\mathcal{M}}(\mathcal{N}') &\approx \mathrm{ext}_{\mathcal{M}}(\mathcal{L})
\end{aligned}$$

Therefore, by definition, we have proved that $\mathcal{N} \leq \mathrm{ext}_{\mathcal{M}}(\mathcal{L})$. Similarly, we can show that if $\mathcal{N} \leq \mathrm{ext}_{\mathcal{M}}(\mathcal{L})$, then $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) \leq \mathcal{L}$.

In practice, we have proved that there is a *Galois connection* [15] between the spaces $\mathscr{N}, \mathscr{L}$, where $\mathrm{sym}_{\mathcal{M}}(\cdot)$ is the *lower adjoint* and $\mathrm{ext}_{\mathcal{M}}(\cdot)$ is the *upper adjoint*.

4.2.1 Decompositional Results

In this Section we solve one of the questions addressed at the beginning of Section 4.2 for the symmetric operator $\parallel$. Specifically, given two testing networks $\mathcal{M}, \mathcal{N}$ such that $\mathcal{M} \parallel \mathcal{N}$ is defined, we show how it is possible to decompose an extensional transition of the form $\mathcal{M} \parallel \mathcal{N} \stackrel{\mu}{\longmapsto} \mathcal{L}$ in extensional transitions for its individual components.

Before proving these decompositional results, we need a technical lemma concerning the labelled transition semantics, defined in Figure 2.4.

Lemma 4.2.9 (Strengthening)**.** Let $\Gamma \rhd M$ be a (well-formed) network, and Γ' be a connectivity graph such that

- $\Gamma' \subseteq \Gamma$, that is $\Gamma'_V \subseteq \Gamma_V$ and $\Gamma_E \subseteq \Gamma_E$
- For any $\mathrm{nodes}(M) \subseteq \Gamma'_V$
- Whenever $\Gamma \vdash m \to n$ for some nodes m, n with either $m \in \mathrm{nodes}(M)$ or $n \in \mathrm{nodes}(M)$, then $\Gamma' \vdash m \to n$.

Then, whenever $\Gamma \rhd M \stackrel{\lambda}{\longrightarrow} M'$, it also holds $\Gamma' \rhd M \stackrel{\lambda}{\longrightarrow} M'$.

Proof. See the Appendix. □

Next we provide some definitions which, though they are not strictly necessary for the topic discussed in this Section, can help in keeping the proofs of compositional and decompositional results readable.

Definition 4.2.10 (Affected Sets of Nodes)**.** Let $\mathcal{M} = \Gamma_M \rhd M$ be a testing network.

For any arbitrary node name m, we define its *output interface* as the set $\mathsf{Out}_m(\mathcal{M}) = \mathsf{Int}(\mathcal{M}) \cap \{n \mid \Gamma_M \vdash m \rightarrow n\}$. Intuitively the set $\mathsf{Out}_m(\mathcal{M})$ contains all the output nodes of $\mathcal{M}$ which are affected by broadcasts performed by node m.

For any node m, its *input set* is the set of nodes $\mathsf{In}_m(\mathcal{M}) = \{n \mid \Gamma_M \vdash m \rightarrow n\}$. Intuitively, this set contains all the internal nodes of $\mathcal{M}$ which are affected by broadcasts performed by node m.

Lemma 4.2.11. Let $\mathcal{M}, \mathcal{N}$ be two networks such that $(\mathcal{M} \parallel \mathcal{N})$ is defined. Then, for any $m \in \mathrm{nodes}(\mathcal{M})$, it holds that $\mathsf{In}_m(\mathcal{N}) = \mathsf{Out}_m(\mathcal{M}) \cap \mathrm{nodes}(\mathcal{N})$.

Proof. . Suppose that $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$. Since $\mathcal{M} \parallel \mathcal{N}$ is defined, then $\mathscr{P}_s(\mathcal{M}, \mathcal{N}) = \mathsf{true}$.

Let m be an arbitrary node name in $\mathrm{nodes}(M)$, and let $n \in \mathsf{Out}_m(\mathcal{M})$, $n \in \mathrm{nodes}(N)$. Then it holds that $\Gamma_M \vdash m \rightarrow n$; since $n \in \mathrm{nodes}(N)$, and $\mathscr{P}_s(\mathcal{M}, \mathcal{N}) = \mathsf{true}$, it also holds $\Gamma_N \vdash m \rightarrow n$. Hence $n \in \mathsf{In}_m(\mathcal{N})$.

Now suppose that $m \in \mathsf{In}_m(\mathcal{N})$; by definition $\Gamma_N \vdash m \rightarrow n$, hence $\Gamma_N \vdash m$. Further, since $m \in \mathrm{nodes}(M)$, we have $m \notin \mathrm{nodes}(N)$, from which it follows $m \in \mathsf{Int}(\mathcal{N})$. For network $\mathcal{N}$ is a testing network, it follows that $n \in \mathrm{nodes}(N)$.

It remains to show that $n \in \mathsf{Out}_m(\mathcal{M})$. However, this is straightforward. In fact, since $\Gamma_N \vdash m \rightarrow n$, $m \in \mathrm{nodes}(M)$ and $\mathscr{P}(\mathcal{M}, \mathcal{N}) = \mathsf{true}$, it follows that $\Gamma_M \vdash m \rightarrow n$. Finally, since $n \in \mathrm{nodes}(N)$, we have that $n \notin \mathrm{nodes}(M)$, hence $n \in \mathsf{Int}(\mathcal{M})$. Since $\Gamma_M \vdash m \rightarrow n$, $n \in \mathsf{Int}(\mathcal{M})$, by definition of output interface we obtain that $n \in \mathsf{Out}_m(\mathcal{M})$.

Thus we have shown that $m \in \mathsf{Out}_m(\mathcal{M})$ if and only if $m \in (\mathsf{In}_m(\mathcal{M}) \cap \mathrm{nodes}(\mathcal{N}))$, which is equivalent to $\mathsf{Out}_m(\mathcal{M}) = \mathsf{In}_m(\mathcal{M}) \cap \mathrm{nodes}(\mathcal{N})$. □

We are ready to prove the decompositional properties of extensional transitions we will need throughout this Chapter. We first analyse the case where a network of the form $(\mathcal{M} \parallel \mathcal{N})$ performs an (extensional) internal transition. Intuitively, if $(\mathcal{M} \parallel \mathcal{N})$ can perform an internal transition, then either some internal activity is performed by one of its individual components, or they interact through a broadcast communication which cannot be detected by the external environment.

Proposition 4.2.12 (Decomposition of Internal Transitions)**.** Let $\mathcal{M}, \mathcal{N}, \mathcal{L}$ be three networks such that $\mathcal{M} \parallel \mathcal{N}$ is defined, and $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{L}$. Then either one of the following is true:

(i) $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{M}'$, and $\mathcal{L} \equiv \mathcal{M}' \parallel \mathcal{N}$ [1],

(ii) or $\mathcal{N} \overset{\tau}{\longmapsto} \mathcal{N}'$, and $\mathcal{L} \equiv \mathcal{M} \parallel \mathcal{N}'$,

(iii) or $\mathcal{M} \overset{c!v \rhd \eta}{\longmapsto}$ for some channel c, value v and set of nodes η such that $\eta \subseteq \mathrm{nodes}(\mathcal{N})$. Also, $\mathcal{N} \overset{m.c?v}{\longrightarrow} \mathcal{N}'$ for some network $\mathcal{N}'$, node m such that $\mathsf{In}_m(\mathcal{N}) = \eta$ and $\mathcal{L} \equiv \mathcal{M}' \parallel \mathcal{N}'$,

(iv) or $\mathcal{N} \overset{c!v \rhd \eta}{\longmapsto} \mathcal{N}'$ for some channel c, value v and set of nodes η, $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{M}'$ for some node m such that $\mathsf{In}_m(\mathcal{M}) = \eta$, and $\mathcal{L} \equiv \mathcal{M}' \parallel \mathcal{N}'$.

Proof. See the Appendix. □

Let us turn our attention to broadcast actions. When a network of the form $(\mathcal{M} \parallel \mathcal{N})$ can perform a broadcast transition, which can be detected by an external set of nodes η, then we expect one of its individual components to perform a broadcast; this affects both the nodes of η and some of the internal nodes of the other network being composed.

Proposition 4.2.13 (Decomposition of Broadcast Transitions)**.** Let $\mathcal{M}, \mathcal{N}, \mathcal{L}$ be three networks such that

$\mathcal{M} \parallel \mathcal{N}$ is defined, and $\mathcal{M} \parallel \mathcal{N} \overset{c!v \rhd \eta}{\longmapsto} \mathcal{L}$ for some channel c, value v and non-empty set of nodes η. Then either one of the following holds:

(i) $\mathcal{M} \overset{c!v \rhd \eta}{\longmapsto} \mathcal{M}'$ for some network $\mathcal{M}'$, and $\mathcal{L} \equiv \mathcal{M}' \parallel \mathcal{N}$,

[1] In practice, we obtain $\mathcal{L} = \mathcal{M}' \parallel \mathcal{N}$; however, proving this equality is more complex than proving $\mathcal{L} \equiv \mathcal{M}' \parallel \mathcal{N}$, and it is not needed for our purposes.

(ii) or $\mathcal{N} \overset{c!v\triangleright\eta}{\longmapsto} \mathcal{N}'$ for some network $\mathcal{N}'$, and $\mathcal{L} \equiv \mathcal{M} \parallel \mathcal{N}'$,

(iii) or $\mathcal{M} \overset{c!v\triangleright(\eta\cup\eta')}{\longmapsto} \mathcal{M}'$, for some set of nodes η' such that $\eta' \subseteq \text{nodes}(N)$; further $\mathcal{N} \overset{m.c?v}{\longrightarrow} \mathcal{N}'$ for some network $\mathcal{N}'$ and node m such that $\mathsf{In}_m(\mathcal{N}) = \eta'$ and $\mathcal{L} \equiv \mathcal{M}' \parallel \mathcal{N}'$,

(iv) or $\mathcal{N} \overset{c!v\triangleright(\eta\cup\eta')}{\longmapsto} \mathcal{N}'$, where $\eta' \subseteq \text{nodes}(M)$. Further, $\mathcal{M} \overset{n.c?v}{\longmapsto} \mathcal{M}'$ for some network $\mathcal{M}'$ and $\mathcal{N}'$ such that $\mathsf{In}_m(\mathcal{M}) = \eta'$, $\mathcal{L} \equiv \mathcal{M}' \parallel \mathcal{N}'$.

Proof. The proof is very similar in details to that of Proposition (4.2.12), in the case where the extensional transition $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{L}$ has been caused by a broadcast which cannot be detected by the external environment. □

Finally, we show how input transitions can be decomposed. If a network of the form $(\mathcal{M} \parallel \mathcal{N})$ can perform an extensional input transition, then it is induced by at least one of its components.

Proposition 4.2.14 (Decomposition of Input Transitions)**.** Let $\mathcal{M}, \mathcal{N}$ be two networks such that $\mathcal{M} \parallel \mathcal{N}$ is defined. Also, let $\mathcal{L}$ be a network such that $\mathcal{M} \parallel \mathcal{N} \overset{m.c?v}{\longmapsto} \mathcal{L}$. Then either one of the following is true:

(i) $m \notin \mathsf{Input}(\mathcal{N})$; in this case we have that $m \in \mathsf{Input}(\mathcal{M})$; further, $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{M}'$ for some network $\mathcal{M}'$ such that $\mathcal{L} = \mathcal{M}' \parallel \mathcal{N}$,

(ii) or $m \notin \mathsf{Int}(\mathcal{M})$, from which it follows $m \in \mathsf{Int}(\mathcal{N})$; in this case $\mathcal{N} \overset{m.c?v}{\longmapsto} \mathcal{N}'$ for some network $\mathcal{N}'$, and $\mathcal{L} = \mathcal{M} \parallel \mathcal{N}'$,

(iii) or $m \in \mathsf{Int}(\mathcal{M}), m \in \mathsf{Int}(\mathcal{N})$; here there exist two networks $\mathcal{M}', \mathcal{N}'$ such that $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{M}'$, $\mathcal{N} \overset{m.c?v}{\longmapsto} \mathcal{N}'$ and $\mathcal{L} = \mathcal{M}' \parallel \mathcal{N}'$.

Proof. See the Appendix. □

4.2.2 Compositional Results

In this section we turn our attention to compositional results for extensional transitions, with respect to the symmetric composition operator $\parallel$.

In general, given two networks $\mathcal{M}, \mathcal{N}$ such that $\mathcal{M} \overset{\mu_1}{\longmapsto} \mathcal{M}'$ for some extensional action μ_1, network $\mathcal{M}'$, $\mathcal{N} \overset{\mu_2}{\longmapsto} \mathcal{N}'$ for some extensional action μ_2 and network $\mathcal{N}'$, our aim is to find an extensional action μ, if possible, such that $(\mathcal{M} \parallel \mathcal{N}) \overset{\mu}{\longmapsto} (\mathcal{M}' \parallel \mathcal{N}')$.

First we prove some lemmas which will be very useful in the proofs of compositional properties for extensional transitions. The first one is a weakening property for the labelled transition semantics, whose rules are defined in Figure 2.4.

Lemma 4.2.15 (Weakening)**.** Let $\Gamma_M \triangleright M$ be a network; also, let Γ_N be a connectivity graph such that, whenever $\Gamma_N \vdash m \to n$ for some nodes m, n such that either $m \in \text{nodes}(M)$ or $n \in \text{nodes}(N)$, then it follows that $\Gamma_M \vdash m \to n$.

If $\Gamma_M \triangleright M \overset{\lambda}{\longrightarrow} M'$ for some λ, then $(\Gamma_M \cup \Gamma_N) \triangleright M \overset{\lambda}{\longrightarrow} M'$. Note that $(\Gamma_M \cup \Gamma_N \triangleright M)$ is not necessarily an element of CNets.

Proof. See the Appendix. □

The other lemma we need concerns the output interface of nodes in composed networks.

Lemma 4.2.16. Let $\mathcal{M}, \mathcal{N}$ be two testing networks such that $\mathcal{M} \parallel \mathcal{N}$ is defined.

Then, for any node $m \in \text{nodes}(\mathcal{M})$, it holds $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) \subseteq \mathsf{Out}_m(\mathcal{M})$.

Proof. Let $\mathcal{M} = \Gamma_M \triangleright M$, $\mathcal{N} = \Gamma_N \triangleright N$ be two testing networks; suppose $\mathcal{M} \parallel \mathcal{N}$ is defined, that is $P_s(\mathcal{M}, \mathcal{N}) =$ true.

Let m be a node in nodes(M) and consider another node $n \in \mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N})$. We need to show that $n \in \mathsf{Out}_m(\mathcal{M})$.

For $n \in \mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N})$, we have that $(\Gamma_M \cup \Gamma_N) \vdash m \to n$, and $n \in \mathsf{Int}(\mathcal{M} \parallel \mathcal{N})$, from which it follows $n \notin \mathrm{nodes}(M \mid N)$. More specifically, $m \notin \mathrm{nodes}(N)$. Since we are assuming that $m \notin \mathrm{nodes}(N)$, note that we have $\Gamma_N \vdash m \nrightarrow n$, since $\Gamma_N \triangleright N$ is a testing network by hypothesis.

Thus we have that $(\Gamma_M \cup \Gamma_N) \vdash m \to n$ and $\Gamma_N \vdash m \nrightarrow n$; this leads to $\Gamma_M m \to n$. Now, since $n \notin \mathrm{nodes}(M \mid N)$, we also have $n \notin \mathrm{nodes}(M)$, and hence $n \in \mathsf{Int}(\Gamma_M \triangleright M)$ (in fact, $\Gamma_M \vdash n$, for $\Gamma_M \vdash m \to n$).

By definition, $m \in \mathrm{nodes}(M)$, $\Gamma_M \vdash m \to n$ and $n \in \mathsf{Int}(\Gamma_M \triangleright M)$ is equivalent to $n \in \mathsf{Out}_m(\Gamma_M \triangleright M)$, as we wanted to prove. □

Corollary 4.2.17. Let $\mathcal{M}, \mathcal{N}$ be two testing networks such that $\mathcal{M} \parallel \mathcal{N}$ is defined. Let also $m \in \mathrm{nodes}(\mathcal{M})$ be a node such that $\mathsf{Out}_m(\mathcal{M}) \subseteq \mathrm{nodes}(N)$; then $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) = \emptyset$.

Proof. Let $\mathcal{M} = \Gamma_M \triangleright M$, $\mathcal{N} = \Gamma_N \triangleright N$ be two testing networks such that $\mathcal{M} \parallel \mathcal{N}$ is defined. If $m \in \mathrm{nodes}(M)$ is a node such that $\mathsf{Out}_m(\mathcal{M}) \subseteq \mathrm{nodes}(N)$, then by Lemma 4.2.16 we have that $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) \subseteq \mathrm{nodes}(N)$.

However, for any arbitrary node $n \in \mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N})$ it holds that $n \in \mathsf{Int}(\mathcal{M} \parallel \mathcal{N})$, from which it follows that $n \notin \mathrm{nodes}(M \mid N)$, and specifically $n \notin \mathrm{nodes}(N)$. Therefore, $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) \cap \mathrm{nodes}(N) = \emptyset$, and since $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) \subseteq \mathrm{nodes}(N)$, it also follows $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) = \emptyset$. □

We are now ready to prove the compositional properties we will need in the following of this Chapter. Unsurprisingly, these are the dual of the decomposition properties of Section 4.2.1. We first focus on (extensional) internal transitions of networks.

Proposition 4.2.18. Let $\mathcal{M}, \mathcal{N}$ be two networks such that $\mathcal{M} \parallel \mathcal{N}$ is defined. Then the following results hold:

(i) If $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{M}'$ for some network $\mathcal{M}'$, then $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}$,

(ii) If $\mathcal{N} \overset{\tau}{\longmapsto} \mathcal{N}'$ for some network $\mathcal{N}'$, then $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M} \parallel \mathcal{N}$,

(iii) If $\mathcal{M} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{M}'$ for some network $\mathcal{M}'$, channel c, value v and set of nodes η such that $\eta \subseteq \mathrm{nodes}(\mathcal{N})$, then there exists a node $m \in \mathsf{Int}(\mathcal{N})$ such that $\mathsf{In}_m(\mathcal{N}) = \eta$. Further, whenever $\mathcal{N} \overset{m.c?v}{\longmapsto} \mathcal{N}'$ then $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}'$,

(iv) If $\mathcal{N} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{N}'$ for some network $\mathcal{N}'$, channel c, value v and set of nodes η such that $\eta \subseteq \mathrm{nodes}(\mathcal{M})$, then there exists a node $m \in \mathsf{Int}(\mathcal{M})$ such that $\mathsf{In}_m(\mathcal{M}) = \eta$. Further, whenever $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{M}'$ then $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}'$.

Proof. See the Appendix. □

Let us now turn our attention to broadcast actions. Again, different cases are possible, depending on whether a broadcast performed by a network $\mathcal{M}$ can affect the internal nodes of $\mathcal{N}$ in the composed network $\mathcal{M} \parallel \mathcal{N}$, and vice-versa.

Proposition 4.2.19. Let $\mathcal{M}, \mathcal{N}$ be two networks such that $(\mathcal{M} \parallel \mathcal{N})$ is defined.

(i) If $\mathcal{M} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{M}'$ for some channel c, value v, network $\mathcal{M}'$ and non-empty set of nodes η such that $\eta \cap \mathrm{nodes}(N) = \emptyset$, then $\mathcal{M} \parallel \mathcal{N} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}$,

(ii) If $\mathcal{N} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{N}'$ for some channel c, value v, network $\mathcal{N}'$ and non-empty set of nodes η, where $\eta \cap \mathrm{nodes}(\mathcal{M}) = \emptyset$, then $\mathcal{M} \parallel \mathcal{N} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{M} \parallel \mathcal{N}'$

(iii) Suppose $\mathcal{M} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{M}'$ for some channel c, value v and non-empty set of nodes η. Also, suppose that $\eta = \eta_1 \cup \eta_2$, where $\eta_1 \cap \mathrm{nodes}(\mathcal{N}) = \emptyset, \eta_2 \subseteq \mathrm{nodes}(\mathcal{N})$.

Then there exists a node $m \in \mathrm{nodes}(\mathcal{M})$ such that $\mathsf{In}_m(\mathcal{N}) = \eta_2$. Further, whenever $\mathcal{N} \overset{m.c!v}{\longmapsto} \mathcal{N}'$, then $\mathcal{M} \parallel \mathcal{N} \overset{c!v \triangleright \eta_1}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}'$

(iv) Suppose $\mathcal{N} \xmapsto{c!v \triangleright \eta} \mathcal{N}'$ for some channel c, value v and non-empty set of nodes η, η_1, η_2 such that $\eta = \eta_1 \cup \eta_2$. Also, assume that $\eta_1 \cap \mathrm{nodes}(\mathcal{N}) = \emptyset, \eta_2 \subseteq \mathrm{nodes}(\mathcal{N})$.

Then there exists a node $m \in \mathrm{nodes}(\mathcal{N})$ such that $\mathsf{In}_m(\mathcal{M}) = \eta_2$. Further, whenever $\mathcal{M} \xmapsto{m.c!v} \mathcal{M}'$, then $\mathcal{M} \parallel \mathcal{N} \xmapsto{c!v \triangleright \eta_1} \mathcal{M}' \parallel \mathcal{N}'$

Proof. The proofs of statements (i) and (ii) are similar in style to those of propositions 4.2.18(i) and 4.2.18(ii), respectively; moreover, the proofs for statements (iii) and (iv) are analogous to those of propositions 4.2.18(iii) and 4.2.18(iv), respectively. □

The last kind of extensional transitions we need to consider are those related to input actions.

Proposition 4.2.20. Let $\mathcal{M}, \mathcal{N}$ be two networks such that $(\mathcal{M} \parallel \mathcal{N})$ is defined.

(i) If $\mathcal{M} \xmapsto{m.c?v} \mathcal{M}'$, and $n \notin \mathsf{Input}(\mathcal{N})$, then $\mathcal{M} \parallel \mathcal{N} \xmapsto{m.c?v} \mathcal{M}' \parallel \mathcal{N}$

(ii) If $\mathcal{N} \xmapsto{m.c?v} \mathcal{N}'$, and $n \notin \mathsf{Input}(\mathcal{M})$, then $\mathcal{M} \parallel \mathcal{N} \xmapsto{m.c?v} \mathcal{M} \parallel \mathcal{N}'$

(iii) If $\mathcal{M} \xmapsto{m.c?v} \mathcal{M}'$ and $\mathcal{N} \xmapsto{m.c?v} \mathcal{M}'$, then $\mathcal{M} \parallel \mathcal{N} \xmapsto{m.c?v} \mathcal{M}' \parallel \mathcal{N}'$.

Proof. See the Appendix. □

As we will see in sections 4.3.1 and 4.4.1, in the following we will also need to compose weak extensional transitions. In general, this problem cannot be solved due to the non-standard definition of weak extensional output transitions, Definition 4.1.5(3b). In fact, it is possible that a weak extensional output is obtained as the result of multiple strong extensional outputs, each of which has to be composed with a weak input action.

Luckily, for our purposes we only need to compose weak transitions of the form $\mathcal{M} \overset{\mu_1}{\Longmapsto} \mathcal{M}'$ with transitions of the form $\mathcal{G} \overset{\mu_2}{\Longmapsto} \mathcal{G}'$, where $\mathcal{G} \in \mathbb{G}$ (see Definition 3.2.5). In this case we are able to provide compositional results for weak extensional transitions.

The main result we need to accomplish this task is stated in the following Lemma.

Lemma 4.2.21. Let $\mathcal{M}$ be a network and $\mathcal{G}$ be a generating network such that $\mathcal{M} \parallel \mathcal{G}$ is defined.

If $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}'$ for some network $\mathcal{M}'$, then $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}$.

Conversely, if $\mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{G}'$ for some network $\mathcal{G}'$, then $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M} \parallel \mathcal{G}'$.

Proof. We only prove the first statement, for the proof for the second one can be obtained symmetrically.

If $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}'$, then by Definition 4.1.5 (1) we have that

$$\mathcal{M}_0 \xmapsto{\tau} \mathcal{M}_1 \xmapsto{\tau} \cdots \xmapsto{\tau} M_k$$

where $\mathcal{M}_0 = \mathcal{M}$, $\mathcal{M}_k = \mathcal{M}'$ and $k \geq 0$. The proof is performed by induction on k.

If $k = 0$, then $\mathcal{M} = \mathcal{M}'$, hence $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}$ trivially holds.

Let then $k > 0$, and suppose that if $\mathcal{M}_0 \xmapsto{\tau} \cdots \xmapsto{\tau} \mathcal{M}_{k-1}$ then $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}_{k-1} \parallel \mathcal{G}$. Since $\mathcal{M}_{k-1} \xmapsto{\tau} \mathcal{M}_k$, by Proposition (4.2.18)(i) it holds that $\mathcal{M}_{k-1} \parallel \mathcal{G} \xmapsto{\tau} \mathcal{M}_k \parallel \mathcal{G}$. Therefore, we have that

$$\mathcal{M}_0 \parallel \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}_{k-1} \parallel \mathcal{G} \xmapsto{\tau} \mathcal{M}_k \parallel \mathcal{G}$$

For $\mathcal{M} = \mathcal{M}_0$, $\mathcal{M}_k = \mathcal{M}'$ and $\overset{\tau}{\Longmapsto}$ is the transitive, reflexive closure of $\xmapsto{\tau}$ (see Definition 4.1.5(1)), it follows that $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}$, as we wanted to prove. □

We will also need the following Lemma.

Lemma 4.2.22 (Single Node Inputs)**.** Let $\mathcal{G} = \Gamma_G \triangleright l[\![P]\!]$, $\mathcal{G}' = \Gamma'_G \triangleright l[\![P]\!]$ be generating networks, and suppose that $\mathcal{G} \xmapsto{m.c?v} \Gamma_G \triangleright l[\![Q]\!]$ for some node m, channel c, value v and process Q.

Then, for any $n \in \mathsf{Input}(\mathcal{G}')$, it holds that $\mathcal{G}' \xmapsto{n.c?v} \Gamma'_G \triangleright l[\![Q]\!]$.

.

Proof. Let n be an arbitrary node in $\mathsf{Input}(\mathcal{G}')$. By definition we have that $\Gamma'_G \vdash n \to l$.

If $\mathcal{G} \xmapsto{m.c?v} \Gamma_G \triangleright l[\![Q]\!]$ for some node m, then by Definition (4.1.3)(2) it follows that $m \in \mathsf{Input}(\mathcal{G})$. Thus, $\Gamma_G m \to l$. Also, $\mathcal{G} \xrightarrow{m.c?v} \Gamma_G \triangleright l[\![Q]\!]$. The last transition can be derived in the intensional semantics (Figure 2.4) only via rules (B-DEAF) and (B-REC). In the first case we have that $P \xrightarrow{c?v} Q$ (recall that $\mathcal{G} = \Gamma_G \triangleright l[\![P]\!]$), while in the last one it follows that $\neg\mathrm{rcv}(P, c)$, and $P = Q$.

If $P \xrightarrow{c?v} Q$, since $\Gamma'_G \vdash n \to l$, by an application of Rule (B-REC) we can derive the transition $\Gamma'_G \triangleright l[\![P]\!] \xrightarrow{n.c?v} l[\![Q]\!]$, which is exactly the transition $\mathcal{G}' \xrightarrow{n.c?v} \Gamma'_G \triangleright l[\![Q]\!]$. For $n \in \mathsf{Input}(\mathcal{G})$, by Definition 4.1.3(2) it follows that $\mathcal{G} \xmapsto{n.c?v} \mathcal{G}'$.

If $\neg\mathrm{rcv}(P, c)$, by an application of Rule (B-DEAF) we obtain the transition $\Gamma_G \triangleright n[\![P]\!] \xrightarrow{n.c?v} n[\![P]\!]$, and since in this case $P = Q$ it follows that this transition can be rewritten as $\mathcal{G} \xrightarrow{n.c?v} \Gamma_G \triangleright l[\![Q]\!]$. Again, since $n \in \mathsf{Input}(\mathcal{G}')$ by hypothesis, it follows that $\mathcal{G}' \xmapsto{n.c?v} \Gamma'_G \triangleright l[\![Q]\!]$. □

Note that in Lemma 4.2.22 the networks $\mathcal{G}$ and $\mathcal{G}'$ do not have necessarily the same connectivity graph.

Composition of weak extensional transitions can now be obtained as a simple consequence of Lemma 4.2.21 and of the composition results we have already proved for strong extensional transitions.

Proposition 4.2.23 (Weak Composition)**.** Let $\mathcal{M}$ be a network, $\mathcal{G}$ be a generating network such that $(\mathcal{M} \parallel \mathcal{G})$ is defined. Suppose that $\mathcal{G} = \Gamma_G \triangleright n[\![P]\!]$ for some connectivity graph Γ_G, process P and node n. Then, for any testing network $\mathcal{M}'$ and generating network $\mathcal{G}'$ the following statements hold.

(i) If $\mathcal{M} \overset{\tau}{\Longrightarrow} \mathcal{M}'$ and $\mathcal{G} \overset{\tau}{\Longrightarrow} \mathcal{G}'$, then $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$

(ii) If $\mathcal{M} \overset{c!v\triangleright\{n\}}{\Longrightarrow} \mathcal{M}'$ and whenever $\mathcal{G} \overset{m.c?v}{\Longrightarrow} \mathcal{G}'$ for some $m \in \mathsf{Input}(\mathcal{G})$, then $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$

(iii) If $\mathcal{G} \overset{c!v\triangleright\eta}{\Longrightarrow} \mathcal{G}'$ for some channel c, value v and set of nodes η such that $\eta \subseteq \mathrm{nodes}(\mathcal{M})$, and $\mathcal{M} \overset{n.c?v}{\Longrightarrow} \mathcal{M}'$, then $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$

(iv) If $\mathcal{M} \overset{c!v\triangleright\eta}{\Longrightarrow} \mathcal{M}'$ for some value v, channel c and set of nodes η such that $n \notin \eta$, then $\mathcal{M} \parallel \mathcal{G} \overset{c!v\triangleright\eta}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}$

(v) If $\mathcal{G} \overset{c!v\triangleright\eta}{\Longrightarrow} \mathcal{G}'$ for some value v, channel c and set of nodes η such that $\eta \cap \mathrm{nodes}(\mathcal{M}) = \emptyset$, then $\mathcal{M} \parallel \mathcal{G} \overset{c!v\triangleright\eta}{\Longrightarrow} \mathcal{M} \parallel \mathcal{G}'$

(vi) If $\mathcal{M} \overset{c!v\triangleright\eta}{\Longrightarrow} \mathcal{M}'$, where $n \in \eta$, and $\mathcal{G} \xmapsto{m.c?v} \mathcal{G}'$ for some node $m \in \mathsf{Input}(\mathcal{G})$, then $\mathcal{M} \parallel \mathcal{G} \overset{c!v\triangleright\eta\setminus\{n\}}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$

(vii) Suppose $\mathcal{G} \overset{c!v\triangleright\eta}{\Longrightarrow} \mathcal{G}'$, where $\eta = \eta_1 \cup \eta_2$ for some η_1, η_2 such that $\eta_1 \cap \mathrm{nodes}(\mathcal{M}) = \emptyset$, $\eta_2 \subseteq \mathrm{nodes}(\mathcal{M})$. Then, whenever $\mathcal{M} \overset{n.c?v}{\Longrightarrow} \mathcal{M}'$, it follows that $\mathcal{M} \parallel \mathcal{G} \overset{c!v\triangleright\eta_1}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$.

(viii) If $\mathcal{M} \overset{m.c?v}{\Longrightarrow} \mathcal{M}'$ and $m \notin \mathsf{Input}(\mathcal{G})$, then $\mathcal{M} \parallel \mathcal{G} \overset{m.c?v}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}$

(ix) If $\mathcal{G} \overset{m.c?v}{\Longrightarrow} \mathcal{G}'$, and $m \notin \mathsf{Input}(\mathcal{M})$, then $\mathcal{M} \parallel \mathcal{G} \overset{m.c?v}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}$

(x) If $\mathcal{M} \overset{m.c?v}{\Longrightarrow} \mathcal{M}'$ and $\mathcal{G} \xmapsto{m.c?v} \mathcal{G}'$, then $\mathcal{M} \parallel \mathcal{G} \overset{m.c?v}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$.

Proof. Most of these proofs are easy to perform. However, some of them present some technical details, which arise as a consequence of the non-standard definition of output actions. See the Appendix for detailed proofs. □

4.3 Full Abstraction for May-testing

In this Section we provide a characterisation result for the may-testing preorder $\sqsubseteq_{\text{may}}$. Specifically, we show that this relation coincides with the inclusion over the set of extensional traces, which are defined later in this Section.

Correspondences between the may-testing preorder and the set of traces have already been proved, in a finitary setting, for well known process calculi [33, 17]. Here, the proofs of the full-abstraction results rely on composition and decomposition results for weak transitions.

The proof of the equivalence between traces inclusion and the may-testing preorder is split in two parts: Soundness and Completeness. Soundness, which states that the trace inclusion relation is included in the may-testing preorder, is proved by showing that the former is compositional. Completeness, which states that the may-testing preorder is a subset of traces inclusion, is proved by exhibiting a characteristic test for each trace.

In this Section we follow the same approach. First we give the formal definition of the set of traces for networks. Then we show that, if we focus on finitary networks, trace inclusion coincides with the may-testing preorder.

Given an alphabet of symbols A, we use the standard notation A^* to denote the set of finite words (or lists) whose occurrences are elements of A. *Traces* are elements of the set $(\mathsf{EAct}_\omega)^*$, where $\mathsf{EAct}_\omega = \mathsf{EAct} \cup \{\omega\}$. In other words, a trace is a finite sequence of symbols, each of which is either a visible extensional action (where visible means that it can be detected by the external environment of some network), or a success mark ω. In the following we use ε to denote the empty trace in the set $(\mathsf{EAct}_\omega)^*$ and t, t' to range over traces. For any $\mu \in \mathsf{EAct}$, $t, t' \in \mathsf{EAct}^*$, we use the notation $\mu::t$ to represent the trace constituted by the symbol μ followed by the trace t, and $t::t'$ as the concatenation of t and t'.

Next we define the set of traces for a network $\mathcal{M}$; the idea is that a trace $\mu_1::\cdots::\mu_k$ ($k \geq 0$) is a trace of $\mathcal{M}$ if the latter is equipped with a sequence of (weak) extensional transitions

$$\mathcal{M} \stackrel{\mu_1}{\Longmapsto} \mathcal{M}_1 \stackrel{\mu_2}{\Longmapsto} \cdots \stackrel{\mu_k}{\Longmapsto} \mathcal{M}_k$$

Further, we have that $\mu_1::\cdots::\mu_k::\omega$ is a trace of $\mathcal{M}$ if, in the sequence of weak extensional transition above, the configuration $\mathcal{M}_k$ is successful.

Definition 4.3.1 (Traces of a Network)**.** The set of traces of a testing network $\mathcal{M}$ is denoted by traces($\mathcal{M}$), and it is defined as the least set such that

(i) $\varepsilon \in \text{traces}(\mathcal{M})$

(ii) If $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{N}$, where $\mathcal{N}$ is a successful configuration, then $\omega \in \text{traces}(\mathcal{M})$

(iii) If $\mathcal{M} \stackrel{m.c?v}{\Longmapsto} \mathcal{N}$ for some node m, channel c, value v and network $\mathcal{N}$ such that $t \in \text{traces}(\mathcal{N})$, then $m.c?v\,t \in \text{traces}(\mathcal{M})$

(iv) If $\mathcal{M} \stackrel{c!v \triangleright \eta}{\Longmapsto} \mathcal{N}$ for some channel c, value v, non-empty set of nodes η and network $\mathcal{N}$ such that $t \in \text{traces}(\mathcal{N})$, then $c!v \triangleright \eta::t \in \text{traces}(\mathcal{M})$

Remark 4.3.2. Note that there are elements of $(\mathsf{EAct}_\omega)^*$ which can never appear in the set of traces of a network. For example, the trace $\omega::c!v \triangleright \eta$[2] does not belong to the set of traces traces($\mathcal{M}$) for any arbitrary network $\mathcal{M}$, as it can be quickly noted by looking at Definition 4.3.1.

In general, no trace of the form $t::\omega::t'$, where $t' \neq \varepsilon$, can appear in traces($\mathcal{M}$) for an any network $\mathcal{M}$.

It is straightforward to observe that traces are strictly related to the may-pass testing relation. Intuitively, a trace which does not contain any occurrence of input actions corresponds to a computation fragment (recall that, by Proposition 4.1.4, a reduction coincides up-to structural equivalence with either a τ-extensional transition or

[2] Formally, this is a shortcut for the trace $\omega::c!v \triangleright \eta::\varepsilon$.

a broadcast extensional transition). Thus, it is easy to note that a successful computation fragment corresponds to a trace which contains a sequence of output extensional actions, followed by the special symbol ω. In the following, we use the term *output trace* when referring to a trace of the form $\mu_1::\mu_k$, $k \geq 0$, where $\mu_i = c_i!v_i \triangleright \eta_i$ for some $c_i \in \mathbf{Ch}$, $v_i \in \mathbf{Val}$ and non-empty set of nodes η_i; note that if $k = 0$, then the trace t is exactly ε. The term *successful trace* is used to refer to traces of the form $t::\omega$, where t is an output trace.

Proposition 4.3.3. Any arbitrary network $\mathcal{M}$ has a successful computation if and only if there exists a success trace $t = c_1!v_1 \triangleright \eta_1::\cdots c_k!v_k \triangleright ::\eta_k::\omega$ such that $t \in \text{traces}(\mathcal{M})$.

In practice, the proof of this statement becomes easier if we reason over (strong) extensional transitions, rather than over their weak counter-part. To this end, the following definition turns out to be very useful.

Definition 4.3.4 (Simple Sets of Traces)**.** For any network $\mathcal{M}$, we define its *simple set of traces*, $\text{traces}_s(\mathcal{M})$, as the smallest set such that

(i) $\varepsilon \in \text{traces}_s(\mathcal{M})$

(ii) If $\mathcal{M}$ is a successful configuration, then $\omega \in \text{traces}_s(\mathcal{M})$

(iii) If $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{N}$ and $t \in \text{traces}_s(\mathcal{N})$, then $t \in \text{traces}_s(\mathcal{M})$

(iv) If $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{N}$ and $t \in \text{traces}_s(\mathcal{N})$ for some node m, channel c, value v and trace t, then $m.c?v :: t \in \text{traces}_s(\mathcal{M})$

(v) If $\mathcal{M} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{N}$ and $t \in \text{traces}_s(\mathcal{M})$ for some channel c, value v, non-empty set of nodes η and trace t, then $c!v \triangleright \eta :: t \in \text{traces}_s(\mathcal{M})$

(vi) If $\mathcal{M} \overset{c!v \triangleright \eta_1}{\longmapsto} \mathcal{N}$, and $c!v \triangleright \eta_2 :: t \in \text{traces}_s(\mathcal{N})$ for some non-empty set of nodes η_2 such that $\eta_1 \cap \eta_2 = \emptyset$, then $c!v \triangleright (\eta_1 \cup \eta_2) :: t \in \text{traces}_s(\mathcal{M})$

For any arbitrary network $\mathcal{M}$, its set of traces $\text{traces}(\mathcal{M})$ and its simple sets of traces $\text{traces}_s(\mathcal{M})$ coincide. This can be proved by using the following Lemma.

Lemma 4.3.5. Let $\mathcal{M}$, $\mathcal{N}$ be testing networks such that $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{N}$. Then, for any trace $t \in \text{traces}_s(\mathcal{N})$ it holds $t \in \text{traces}_s(\mathcal{M})$.

Proof. See the Appendix. □

Proposition 4.3.6. For any network $\mathcal{M}$, $\text{traces}(\mathcal{M}) = \text{traces}_s(\mathcal{M})$.

Proof. See the Appendix. □

Proof of Proposition 4.3.3 For the only if implication, we actually prove that a network $\mathcal{M}$ has a successful computation only if there exists an output trace t such that $t = c_1!v_1 \triangleright \eta_1::\cdots::c_j!v_j \triangleright \eta_j$ and $t::\omega \in \text{traces}_s(\mathcal{M})$. Then the result follows from Proposition 4.3.6.

Suppose that $\mathcal{M}$ has a successful computation fragment. We show that there exists a trace $t = c_1!v_1 \triangleright ::\cdots::c_j!v_j \triangleright \eta_j$ such that $t::\omega \in \text{traces}_s(\mathcal{M})$ by induction on the minimal length k of a successful computation for $\mathcal{M}$.

$k = 0$ Then $\mathcal{M}$ is a successful configuration. Further $\omega \in \text{traces}_s(\mathcal{M})$ by Definition 4.3.4 (ii). In this case it suffices to let $j = 0$, hence $t = \varepsilon$, to obtain $t::\omega \in \text{traces}_s(\mathcal{M})$.

$k > 0$ Suppose the statement is true for $k-1$. In this case we have that $\mathcal{M} \twoheadrightarrow \mathcal{N}$ for some network $\mathcal{N}$ which has a successful computation fragment of length $k-1$. By inductive hypothesis, there exists an index $j \geq 0$ and a trace $t = c_1!v_1 \triangleright \eta_1::\cdots::c_j!v_j \triangleright \eta_j::\omega$ such that $t :: \omega \in \text{traces}_s(\mathcal{N})$. Since $\mathcal{M} \twoheadrightarrow \mathcal{N}$, by Proposition 4.1.4 we have two possible cases.

(a) $\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{N}'$, for some network $\mathcal{N}' \equiv \mathcal{N}$.

It is not difficult to note that traces are preserved by structurally congruent networks[3]. Therefore, $t::\omega \in \text{traces}(\mathcal{N}')$. Now it follows from Definition 4.3.4(iii) that $t::\omega \in \text{traces}_s(\mathcal{M})$.

(b) $\mathcal{M} \stackrel{c_0!v_0 \triangleright \eta_0}{\longmapsto} \mathcal{N}'$ for some channel c_0, value v_0,set of nodes η_0 and network $\mathcal{N}'$ such that $\mathcal{N}' \equiv \mathcal{N}$. In this case we have that $t::\omega \in \text{traces}_s(\mathcal{N}')$, and by Definition 4.3.4(v) it follows that $c_0!v_0 \triangleright \eta_0::t::\omega \in \text{traces}_s(\mathcal{M})$.

It remains to note that the trace $c_0!v_0 \triangleright \eta_0::t$ is exactly $c_0!v_0 \triangleright \eta_0::\cdots::c_j!v_j \triangleright \eta_j$, which is an output trace.

For the if implication, suppose that there exists a (possibly empty) output trace $t = c_1!v_1 \triangleright \eta_1 :: \cdots :: c_j!v_j \triangleright \eta_j$ for some $j \geq 0$ (the case $j = 0$ corresponds to the empty trace ε) and $t::\omega \in \text{traces}(\mathcal{M})$. Then, it is not difficult to show that $\mathcal{M} \stackrel{c_1!v_1 \triangleright \eta_1}{\Longmapsto} \cdots \stackrel{c_j!v_j \triangleright \eta_j}{\Longmapsto} \mathcal{N}$, with $\mathcal{N}$ being a successful configuration. By Proposition 4.1.9 we have that $\mathcal{M} \rightarrow^* \mathcal{N}$, hence $\mathcal{M}$ has a successful computation fragment. □

We have defined traces formally and proved some of their properties. The rest of this Section is devoted to prove the following result.

Theorem 4.3.7 (Characterisation of May-Testing)**.** Let $\mathcal{M}, \mathcal{N}$ be two finitary composable networks; suppose that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$ and $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$.

Then $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ if and only if $\text{traces}(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$.

Proof. The proof of Theorem 4.3.7 is split in two parts; Soundness (Theorem 4.3.8) and Completeness (Theorem 4.3.19).

Let $\mathcal{M}, \mathcal{N}$ be two finitary networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$; if $\text{traces}(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$, then $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ as a consequence of Theorem 4.3.8. Conversely, if $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, then Theorem 4.3.19 ensures that $\text{traces}(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$. □

4.3.1 Soundness

This Section is devoted to the proof of the following Theorem:

Theorem 4.3.8 (Soundness for May-Testing)**.** Let $\mathcal{M}, \mathcal{N}$ be two testing networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$. Then $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$.

Before looking at the technical details needed to perform the proof of Theorem 4.3.8, let us look briefly how this result can be proved in simpler process calculi [33]; In this case the proof of soundness for the may-testing equivalence can be performed by showing that, in a finitary setting, trace inclusion is preserved by the parallel composition operator |, proper of the CCS calculus.

The proof of compositionality for this operator is provided by first defining how, for any two arbitrary processes P, Q, it is possible to infer a trace for the composite processes $P \mid Q$ from the traces of its individual components. This operation is known as *zipping*.

Conversely, if a trace for the composite process $P \mid Q$ is known, then it is possible to infer a set of couples of traces, $\langle t, t' \rangle$, which contains at least an element whose left and right projections are traces of P, Q, respectively. This operation is known as *unzipping*.

Compositionality of trace inclusion is then obtained by showing that, if we unzip a trace t to obtain a set of couples of traces $\langle t, t' \rangle$, and then we zip the two traces contained in one of these, we obtain the original trace t.

Specifically, suppose that P, Q, R are CCS processes, and the traces of P are included in the traces of Q; further, let t be an arbitrary trace of the composite process $P \mid R$. By unzipping t, we obtain a set $\{\langle t_i, t'_i \rangle\}_{i \in I}$ of couples of traces, where I is an index set. Now, we are ensured that, for at least one index $i \in I$ t_i is a trace of P

[3] This follows because extensional transitions are preserved by structurally congruent networks, which is itself a consequence of intensional transitions to be preserved by structurally congruent networks; see Proposition 2.4.6.

and t'_i is a trace of R. Further, for any index $i \in I$ with such a property it holds that t_i is also a trace of Q. Now, by zipping t_i and t'_i we obtain that t is a trace of the process $Q \mid R$.

In our calculus of networks, however, zipping and unzipping properties for traces are not enough to prove the compositionality of trace set inclusion, with respect to the extension operator $\between$. This is because the results concerning zipping and unzipping of traces rely on the compositional and decompositional properties of extensional transitions, which have been proved only for the symmetric composition operator $\|$.

In practice, when considering three networks $\mathcal{M}, \mathcal{N}, \mathcal{T}$ such that $\mathcal{M} \between \mathcal{T}$, $\mathcal{N} \between \mathcal{T}$ are defined and traces$(\mathcal{M}) \subseteq$ traces$(\mathcal{N})$, we have first to perform a change of the composition operator used in order to be able to use zipping and unzipping properties. The operators $\mathrm{sym}_{\mathcal{M}}(\cdot)$ and $\mathrm{sym}_{\mathcal{N}}(\cdot)$ serve to this purpose, as shown in Proposition 4.2.4. However, by applying these operators we obtain the networks $\mathcal{M} \| \mathrm{sym}_{\mathcal{M}}(\mathcal{T})$ and $\mathcal{N} \| \mathrm{sym}_{\mathcal{N}}(\mathcal{T})$; now note that $\mathrm{sym}_{\mathcal{M}}(\mathcal{T})$ and $\mathrm{sym}_{\mathcal{N}}(\mathcal{T})$ could be different networks, with different sets of traces[4]

In order to be able to prove Theorem 4.3.8, we need to be able to infer the traces of $\mathrm{sym}_{\mathcal{N}}(\mathcal{T})$ from those of $\mathrm{sym}_{\mathcal{M}}(\mathcal{T})$, where $\mathcal{M}, \mathcal{N}$ and $\mathcal{T}$ are the three networks considered above; we call this operation *switching*.

Now we proceed by presenting how traces can be zipped for networks. In our case zipping of traces is defined as a set of traces, and the definition of zipping is parametric in two networks. For the moment, we assume that the right hand side of a composition of the form $(\mathcal{M} \| \mathcal{G})$, is a generating network (Definition 3.2.5).

Definition 4.3.9 (Zipping)**.** Let $\mathcal{M} = \Gamma_M \rhd M, \mathcal{G} = \Gamma_G \rhd n[\![P]\!]$ be respectively a testing network and a generating network. For any two traces t, t', we define the set of traces $\mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, t')$ as the least set such that

(i) $\varepsilon \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon, \varepsilon)$

(ii) $\omega \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(\omega, t')$

(iii) $\omega \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, \omega)$

(iv) $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2)$ implies $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(c!v \rhd \{n\}::t_1, m.c?v::t_2)$

(v) If $\eta \subseteq \mathrm{nodes}(\mathcal{M})$ and $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2)$, then $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(n.c?v::t_1, c!v \rhd \eta::t_2)$

(vi) If $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t')$, and η is a non-empty set of nodes such that $n \notin \eta$, then $c!v \rhd \eta::t \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(c!v \rhd \eta::t_1, t')$

(vii) Let η be a non-empty set of nodes such that $\eta \cap \mathrm{nodes}(\mathcal{M}) = \emptyset$;
if $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, t_2)$ then $c!v \rhd \eta::s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, c!v \rhd \eta::t_2)$

(viii) If $n \in \eta$, $\eta \setminus \{n\} \neq \emptyset$ and $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2)$, then $c!v \rhd (\eta \setminus \{n\})::s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(c!v \rhd \eta::t_1, m.c?v::t_2)$

(ix) Let $\eta = \eta_1 \cup \eta_2$, where $\eta_1 \cap \mathrm{nodes}(\mathcal{M}) = \emptyset, \emptyset \subset \eta_2 \subseteq \mathrm{nodes}(\mathcal{M})$; if $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2)$ then $c!v \rhd \eta_1::s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(n.c?v::t_1, c!v \rhd \eta::t_2)$

(x) if $m \notin \mathsf{Input}(\mathcal{G})$ and $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t')$, then $m.c?v::s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(m.c?v :: t_1, t')$

(xi) if $m \notin \mathsf{Int}(\mathcal{M})$ and $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, t_2)$, then $m.c?v::s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, m.c?v::t_2)$

(xii) if $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2)$, then $m.c?v::s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(m.c?v :: t_1, m.c?v :: t_2)$

The main property of the function $\mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(\cdot, \cdot)$ that we are interested in, as we have already pointed out, is stated in Proposition 4.3.10.

Proposition 4.3.10 (Zipping)**.** Let $\mathcal{M} \in \mathsf{CNets}, \mathcal{G} \in \mathbb{G}$ be two networks such that $\mathcal{M} \| \mathcal{G}$ is defined. Then, for any traces $t \in \mathrm{traces}(\mathcal{M}), t' \in \mathrm{traces}(\mathcal{G})$ and $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, t')$ it holds $s \in \mathrm{traces}(\mathcal{M} \| \mathcal{G})$.

Proof. This statement follows directly from the weak/strong composition properties we have proved in Section 4.2.2. See the Appendix for a detailed proof. □

[4] Specifically, the input and output interface of these two networks can be different. This difference affects directly the sets of traces of the two networks.

Next we focus on the definition of unzipping.

Definition 4.3.11 (Unzipping)**.** Let $\mathcal{M},\mathcal{G}$ be respectively a testing network and a generating network such that $\mathcal{M} \parallel \mathcal{G}$ is defined. Further, let $\mathcal{M} = \Gamma_M \triangleright M$, $\mathcal{G} = \Gamma_G \triangleright n[\![P]\!]$.

For any trace s, $\text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ is the least set of pairs of traces $\langle t, t' \rangle$ which satisfies the following constraints:

(i) $\langle \varepsilon, \varepsilon \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon)$

(ii) for any trace t', $\langle \omega, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(\omega)$

(iii) for any trace t, $\langle t, \omega \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(\omega)$

(iv) For any non-empty set of nodes $\eta \subseteq \mathsf{Output}(\mathcal{M})$ and node $m \in \mathsf{Input}(G)$, if $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then $\langle c!v \triangleright \eta \cup \{n\} :: t_1, m.c?v :: t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s)$, for any node $m \in \mathsf{Input}(\mathcal{G})$

(v) For any set of nodes η and $\langle t, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then $\langle n.c?v :: t, c!v \triangleright \eta \cup \eta' :: t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s)$, provided that $\emptyset \subset \eta' \subseteq \text{nodes}(\mathcal{M})$

(vi) For any set of nodes η, if $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then $\langle c!v \triangleright \eta :: t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s)$

(vii) For any set of nodes η, if $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then $\langle t_1, c!v \triangleright \eta :: t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s)$

(viii) If $m \notin \mathsf{Input}(\mathcal{G})$ and $\langle t_1, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then $\langle m.c?v :: t_1, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(m.c?v :: s)$

(ix) If $m \notin \mathsf{Input}(\mathcal{M})$ and $\langle t, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then $\langle t, m.c?v :: t_2 \rangle \in \text{unzip}^{\mathcal{N}}_{\mathcal{M}}(m.c?v :: s)$

(x) If $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then $\langle m.c?v :: t_1, m.c?v :: t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(m.c?v :: s)$

(xi) If $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta_1 :: c!v \triangleright \eta_2 :: s)$, and $\eta_1 \cap \eta_2 = \emptyset$, then $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright (\eta_1 \cup \eta_2) :: s)$.

(xii) If $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then for any node $m \in \mathsf{Input}(\mathcal{G})$, $\langle c!v \triangleright \{n\} :: t_1, m.c?v :: t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$

(xiii) If $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, then for any $\eta \subseteq \text{nodes}(\mathcal{M})$, $\langle n.c?v :: t_1, c!v \triangleright \eta :: t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$.

Our aim for unzipping is to prove that, given a trace $s \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$, for some testing network $\mathcal{M}$ and generating network $\mathcal{G}$ for which $\mathcal{M} \parallel \mathcal{G}$ is defined, there exists a pair $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$. The proof of this statement relies on the following Lemma.

Lemma 4.3.12. Let $\mathcal{M},\mathcal{G}$ be respectively a testing network and a generating network such that $\mathcal{M} \parallel \mathcal{G}$ is defined. Also, suppose that $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\Longmapsto} (\mathcal{M}' \parallel \mathcal{G}')$ for some networks $\mathcal{M}', \mathcal{G}'$.

If $s \in \text{traces}(\mathcal{M}' \parallel \mathcal{G}')$, and $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ is a pair of traces such that $t_1 \in \text{traces}(\mathcal{M}'), t_2 \in \text{traces}(\mathcal{G}')$, then there exists a pair of traces $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$.

Proof. See the Appendix. □

Proposition 4.3.13 (Unzipping)**.** Let $\mathcal{M},\mathcal{G}$ be respectively a testing network and a generating network such that $\mathcal{M} \parallel \mathcal{G}$ is defined.

Then, for any trace $s \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$ there exists a pair $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$.

Proof. See the Appendix. □

As we have already discussed, in our calculus zipping and unzipping properties alone are not sufficient to show that trace inclusion is compositional with respect to the extension operator $\between$. To accomplish this goal, we need another definition, which is provided below.

Definition 4.3.14 (Switching)**.** Let Γ_1, Γ_2 be two connectivity graphs, P be a process and n be a node such that $\mathcal{G} = \Gamma_1 \triangleright n[\![P]\!]$, $\mathcal{H} = \Gamma_2 \triangleright n[\![P]\!]$

For any trace t, we define the set $\text{switch}_{\mathcal{H}}(t)$ as the smallest set such that

(i) $\varepsilon \in \text{switch}_{\mathcal{H}}(\varepsilon)$

(ii) $\omega \in \text{switch}_{\mathcal{H}}(\omega)$

(iii) $c!v \triangleright \eta' :: \text{switch}_{\mathcal{H}}(t) \in \text{switch}_{\mathcal{H}}(c!v \triangleright \eta :: t)$, where $\eta' = \mathsf{Output}(\mathcal{H})$

(iv) for any $m' \in \mathsf{Input}(\mathcal{H})$ and $t' \in \text{switch}_{\mathcal{H}}(t)$, $m'.c?v :: t' \in \text{switch}_{\mathcal{H}}(m.c?v :: t)$

For switching we will need the properties stated below.

Lemma 4.3.15. Let $\mathcal{G} = \Gamma_1 \triangleright n[\![P]\!]$, $\mathcal{H} = \Gamma_2 \triangleright n[\![P]\!]$ be two networks such that

- $\mathsf{Input}(\mathcal{G}) = \emptyset$ implies $\mathsf{Input}(\mathcal{H}) = \emptyset$,
- $\mathsf{Output}(\mathcal{G}) = \emptyset$ implies $\mathsf{Output}(\mathcal{H}) = \emptyset$.

Then, for any $t \in \text{traces}(\mathcal{G})$ and $t' \in \text{switch}_{\mathcal{H}}(t')$ it holds that $t' \in \text{traces}(\mathcal{H})$.

Proof. By induction on the trace t. We show the details only for the case $t = c!v \triangleright \eta :: t_1$, for the cases ε, ω are trivial, while the case $m.c?v :: t_1$ can be handled similarly.

If $t = c!v \triangleright \eta :: t_1$, then $\mathcal{G} \overset{c!v \triangleright \eta}{\Longrightarrow} (\Gamma_1 \triangleright n[\![Q]\!])$ for some process P, $\eta = \mathsf{Output}(\mathcal{G})$ is a non-empty set of nodes and $t_1 \in \text{traces}(\Gamma_1 \triangleright n[\![Q]\!])$.

By hypothesis we have that $\eta' = \mathsf{Output}(\mathcal{H})$ is non-empty. Further, it is straightforward to note that in this case we have that $\mathcal{H} \overset{c!v \triangleright \eta'}{\Longrightarrow} \Gamma_2 \triangleright n[\![Q]\!]$.

Since $t_1 \in \text{traces}(\mathcal{M})$, by inductive hypothesis every trace $t'' \in \text{switch}_{\mathcal{H}}(t_1)$ enjoys the property $t'' \in \text{traces}(\Gamma_2 \triangleright n[\![Q]\!])$ [5] Now note that, if $t' \in \text{switch}_{\mathcal{G}}(c!v \triangleright \eta :: t_1)$, then $t' = c!v \triangleright \eta' :: t''$ for some $t'' \in \text{switch}_{\mathcal{H}}(t_1)$. Since $t'' \in \text{traces}(\Gamma_2 \triangleright n[\![Q]\!])$, and $\mathcal{H} \overset{c!v \triangleright \eta'}{\Longrightarrow} (\Gamma_2 \triangleright n[\![Q]\!])$, it follows that $t' \in \text{traces}(\mathcal{H})$, as we wanted to prove. □

Proposition 4.3.16. [Switching] Let $\mathcal{M}, \mathcal{N}$ be two networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}), \mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$. Suppose also that $\mathcal{G}$ is a generating network such that both $\mathcal{M} \between \mathcal{G}$ and $\mathcal{N} \between \mathcal{G}$ are defined.

Let $\mathcal{H} = \text{sym}_{\mathcal{M}}(\mathcal{G})$, $\mathcal{K} = \text{sym}_{\mathcal{N}}(\mathcal{G})$. Then, for any trace s and $\langle t, t' \rangle \in \text{unzip}^{\mathcal{H}}_{\mathcal{M}}(s)$ such that $t \in \text{traces}(\mathcal{M})$, there exists a trace $t'' \in \text{switch}_{\mathcal{K}}(t')$ such that $s \in \text{zip}^{\mathcal{H}}_{\mathcal{N}}(t, t'')$.

Proof. See the Appendix. □

We have set up all the tools we require to prove that trace inclusion is preserved by network composition via the extension operator $\between$. We first prove this property under the assumption that the right hand side of a composite network $\mathcal{M} \between \mathcal{G}$ is a generating network; then we use the principle of network induction 3.2.7 to extend the result to arbitrary networks.

Proposition 4.3.17 (Single Node Compositionality for Traces)**.** Let $\mathcal{M}, \mathcal{N}$ be two testing network such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$ and $\text{traces}(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$. Then, for any generating network $\mathcal{G}$ such that $\mathcal{M} \between \mathcal{G}$ and $\mathcal{N} \between \mathcal{G}$ are defined.

Then $\text{traces}(\mathcal{M} \between \mathcal{G}) \subseteq \text{traces}(\mathcal{N} \between \mathcal{G})$.

Proof. Let $\mathcal{H} = \text{sym}_{\mathcal{M}}(\mathcal{G})$, $\mathcal{K} = \text{sym}_{\mathcal{N}}(\mathcal{G})$. By Proposition 4.2.4 we have that $\mathcal{M} \between \mathcal{G} = \mathcal{M} \parallel \mathcal{H}$, while $\mathcal{N} \between \mathcal{G} = \mathcal{N} \parallel \mathcal{K}$.

Let $s \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$. Then there exists a pair $\langle t, t' \rangle \in \text{unzip}^{\mathcal{H}}_{\mathcal{M}}(s)$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$, as stated in Proposition 4.3.13.

Further, we have that $t \in \text{traces}(\mathcal{N})$ by hypothesis; moreover, by Lemma 4.3.15 there exists a trace $t'' \in \text{switch}_{\mathcal{K}}(t')$ such that $t' \in \text{traces}(\mathcal{K})$. By Proposition 4.3.16 it also holds that $s \in \text{zip}^{\mathcal{K}}_{\mathcal{N}}(t, t'')$; finally, Proposition 4.3.10 ensures that $s \in \text{traces}(\mathcal{N} \parallel \mathcal{K})$.

Since $\mathcal{N} \parallel \mathcal{K} = \mathcal{N} \between \mathcal{G}$, we have proved that for any $s \in \text{traces}(\mathcal{M} \between \mathcal{G})$ it also holds $s \in \text{traces}(\mathcal{N} \between \mathcal{G})$. Hence $\text{traces}(\mathcal{M} \between \mathcal{G}) \subseteq \text{traces}(\mathcal{N} \between \mathcal{G})$. □

[5] In practice, the inductive hypothesis holds for any trace $t'' \in \text{switch}_{\mathcal{H}'}(t_1)$, where $\mathcal{H}' = \Gamma_2 \triangleright n[\![Q]\!]$. However, it is trivial to show that the latter set coincides with $\text{switch}_{\mathcal{H}}(t_1)$.

So far we have focused on the specific case in which networks are always extended with a generating network. However, we want to prove that trace inclusion is preserved by the operator $\between$ in the more general case that an arbitrary network $\mathcal{T} \in \mathsf{CNets}$ appears in the right hand side of a composition ($\mathcal{M} \between \mathcal{T}$).

Corollary 4.3.18. Let $\mathcal{M}, \mathcal{N}$ be two networks such that traces($\mathcal{M}$) $\subseteq$ traces($\mathcal{N}$).

Then, for any network $\mathcal{T}$ such that both $\mathcal{M} \between \mathcal{T}$ and $\mathcal{N} \between \mathcal{T}$ are defined, it holds that traces($\mathcal{M} \between \mathcal{T}$) $\subseteq$ traces($\mathcal{N} \between \mathcal{T}$).

Proof. The proof is performed by network induction (Theorem 3.2.7) on the network $\mathcal{T}$, noticing that trace inclusion is preserved by structurally congruent networks. This is because we have already pointed out that traces are preserved by structurally congruent networks.

If $\mathcal{T}$ is the identity network $\mathcal{O}$, then $\mathcal{M} \between \mathcal{O} \equiv \mathcal{M}, \mathcal{N} \between \mathcal{O} \equiv \mathcal{N}$. Since traces($\mathcal{M}$) $\subseteq$ traces($\mathcal{N}$), and trace inclusion is preserved by structurally congruent networks, then traces($\mathcal{M} \between \mathcal{O}$) $\subseteq$ traces($\mathcal{N} \between \mathcal{O}$).

Otherwise, $\mathcal{T} \equiv \mathcal{T}' \between \mathcal{G}$ for some network $\mathcal{T}'$ and generating network $\mathcal{G}$. By inductive hypothesis, we have that traces($\mathcal{M} \between \mathcal{T}'$) $\subseteq$ traces($\mathcal{N} \between \mathcal{T}'$). By Proposition 4.3.17 it follows that traces(($\mathcal{M} \between \mathcal{T}'$) $\between \mathcal{G}$) $\subseteq$ traces(($\mathcal{N} \between \mathcal{T}'$) $\between \mathcal{G}$).

Recall that the operator $\between$ is associative (Proposition 3.2.3), thus we have that that ($\mathcal{M} \between \mathcal{T}'$) $\between \mathcal{G} = \mathcal{M} \between (\mathcal{T}' \between \mathcal{G}) \equiv \mathcal{M} \between \mathcal{T}$. Similarly, we can prove that ($\mathcal{N} \between \mathcal{T}'$) $\between \mathcal{G} \equiv \mathcal{N} \between \mathcal{T}$. For trace inclusion is preserved by structurally congruent networks, it is easy to note that traces($\mathcal{M} \between \mathcal{G}$) $\subseteq$ traces($\mathcal{N} \between \mathcal{G}$). □

Now that we have proved that trace inclusion is compositional with respect to the operator $\between$, we can finally prove Theorem 4.3.8.

Proof of Theorem 4.3.8 Let $\mathcal{M}, \mathcal{N}$ be two networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$ and traces($\mathcal{M}$) $\subseteq$ traces($\mathcal{N}$); in order to show that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, it it sufficient to show that, for any arbitrary test $\mathcal{T}$ such that ($\mathcal{M} \between \mathcal{T}$) and ($\mathcal{N} \between \mathcal{T}$) are both defined, $\mathcal{M}$ **may-pass** $\mathcal{T}$ implies $\mathcal{N}$ **may-pass** $\mathcal{T}$.

To this end, let $\mathcal{T}$ be a test such that it can be used to extend both the networks $\mathcal{M}, \mathcal{N}$; further, suppose that $\mathcal{M}$ **may-pass** $\mathcal{T}$. By definition $\mathcal{M} \between \mathcal{T}$ has a successful computation; then, by Proposition 4.3.3 there exists a success trace t such that $t \in$ traces($\mathcal{M} \between \mathcal{T}$).

Since traces($\mathcal{M}$) $\subseteq$ traces($\mathcal{N}$) by hypothesis, it follows from Corollary 4.3.18 that $t \in$ traces($\mathcal{N} \between \mathcal{T}$). Therefore we have that $\mathcal{N} \between \mathcal{T}$ has a successful computation, from which we obtain $\mathcal{N}$ **may-pass** $\mathcal{T}$. □

4.3.2 Completeness

This Section is devoted to prove the following Theorem.

Theorem 4.3.19 (Completeness of May-testing)**.** Let $\mathcal{M}, \mathcal{N}$ be two finitary, proper networks. If $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, then traces($\mathcal{M}$) $\subseteq$ traces($\mathcal{N}$).

Remark 4.3.20. The reader could argue that the restriction to finitary networks in Theorem 4.3.2 could be a serious limitation for our proof technique. However, we recall that trace inclusion can still be used to prove whether two networks are may-testing related in an infinitary setting, for such a restriction was not used in the assumptions of soundness, Theorem 4.3.8.

Further, it is often the case that wireless networks and distributed applications are designed following a client/server model. Once a client sends a request to a distributed application, it waits to receive an answer; the application should be designed such that such a request is answered in a finite amount of time. Representing this kind of systems in our framework always leads to finitary networks.

The standard approach to prove Theorem 4.3.2 is to define a characteristic test $\mathcal{T}_t$ for each trace t, in a way such that $\mathcal{M}$ **may-pass** $\mathcal{T}_t$ if and only if $t \in \mathcal{M}$. Consider then any two arbitrary networks $\mathcal{M}, \mathcal{N}$ such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$ and $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$, and suppose that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$.

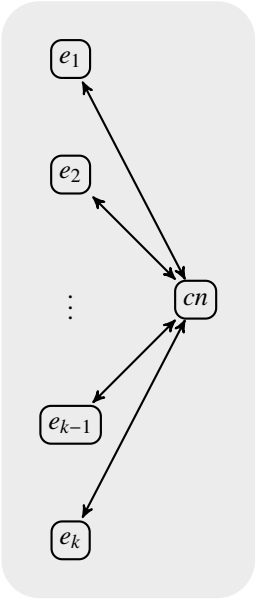


Figure 4.6: The connectivity graph Γ_T used to test networks; here $\{e_1, \cdots, e_n\} = \eta$, and cn is a fresh node name.

Then, for any trace $t \in \text{traces}(\mathcal{M})$ it holds that $\mathcal{M}$ **may-pass** $\mathcal{T}_t$, therefore we also have that $\mathcal{N}$ **may-pass** $\mathcal{T}_t$. At this point it is straightforward to note that we also have $t \in \text{traces}(\mathcal{N})$. Due to the arbitrariness of the trace t, this line of reasoning leads to establish that $\text{traces}(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$.

Note that, in a finitary setting, then the set of channels Ch and the set of values Val need to be finite, for otherwise every network $\mathcal{M}$ such that $\mathsf{Input}(\mathcal{M}) \neq \emptyset$ there would exist a transition of the form $\mathcal{M} \xmapsto{m.c?v} \mathcal{M}'$, where c is an arbitrary channel and v is an arbitrary value; then $\mathcal{M}$ would have an infinite number of (strong) extensional transitions, contradicting the hypothesis that such a network is finite branching.

The main topic of this section is that of providing a characteristic test for any possible trace. To this end, we introduce some shortcut notation for processes; while this is not strictly necessary, it will help to keep the definition of the tests clear.

Given a (possibly empty) list of variables $x_1, \cdots, x_n$, we define the process $c?(x_1, \cdots, x_n).P$ to be exactly P if the list above is empty, $c?(x_1).c?(x_2, \cdots, x_n).P$ otherwise. Further, we let the process **LOCK** to be exactly $\sum_{c \in \text{Ch}} c?(x).\mathbf{0}$; that is, **LOCK** terminate whenever it receives an input on any given channel. We have already noted that in a finitary setting the set Ch is finite, so that the process **LOCK** is well defined.

Similarly, we let $\mathbf{ALLOW}(c, x).P = c?(x).P + \sum_{\substack{d \in \mathbf{Ch} \\ d \neq c}} d?(x).\mathbf{0}$. This process allow a value to be received along channel c, but it deadlocks if it detects a value being transmitted along a different channel.

We are now ready to define the networks that we use for testing traces. For the moment we assume that the input and output interface of a network are fixed; given two sets of nodes $\eta_{\text{in}}, \eta_{\text{out}}$, we restrict our attention to networks $\mathcal{M}$ such that $\mathsf{Input}(\mathcal{M}) = \eta_{\text{in}}, \mathsf{Output}(\mathcal{M}) = \eta_{\text{out}}$. Further, we let $\mathsf{Int}(\mathcal{M}) = \eta_1 \cup \eta_2 = \{e_1, \cdots e_k\}$ and we assume that $k > 0$; that is, either $\eta_{\text{in}} \neq \emptyset$ or $\eta_{\text{out}} \neq \emptyset$.

First we define the network topology that we use for the characteristic tests of traces; then we provide a definition of the code that each node is running in these networks. This definition is given inductively over traces. Before proving that a test $\mathcal{T}_t$ we define is indeed a characteristic test for the trace t, we supply an informal explanation of its behaviour.

The connectivity graph that we use for networks is depicted in Figure 4.6. Here the nodes $e_1, \cdots, e_n$ correspond to the external nodes of the network $\mathcal{M}$ being tested, while cn is a fresh node, which we call *controller node*. Since the extensional behaviour of $\mathcal{M}$ is being observed by multiple nodes, there is the need to coordinate the way they interact with the external environment; this task is accomplished by the controller node cn.

Given a trace t, the network $\mathcal{T}_t = \Gamma_T \triangleright T_t$ which we use for testing whether t is a trace of a network is defined as follows; the system term T_t has the form $\prod_{i=1}^{k} e_i[\![P_t^i]\!] \,|\, cn[\![P_t]\!]$. For any node $e_1, \cdots, e_k$ and for the node cn,

the processes $P_t^i, i = 1, \cdots k$, as well as the process P_t, are defined inductively over the trace t. Note that, since we are assuming that the networks being tested are proper, there is no need to test for traces which contain the success symbol ω. In fact, the reason of this restriction is that the trace ω cannot be tested, for the success process ω has no extensional action associated.

In order to present the code that each node of the testing network $\mathcal{T}_t$ runs, we use some process definitions which make the analysis of the computations of experiments easier. Here we assume that i ranges over $1, \cdots k$.

$$\begin{aligned} P_{\text{next}}.Q &= cc!\langle \mathbf{PROCEED} \rangle .Q \\ P_{\text{check}}.Q &= cc!\langle \mathbf{CHECK} \rangle .cc?(x_1, \cdots, x_k).Q \end{aligned}$$

$$\begin{aligned} P'_{\text{next}}.Q &= cc?(x).Q + \mathbf{LOCK} \\ P'_{\text{check}}.Q &= cc?(x).(cc!\langle \mathbf{CLEAR} \rangle .Q + \mathbf{LOCK}) + \mathbf{LOCK} \end{aligned}$$

We can use these processes to define the code running at each node of a test; these are defined below.

(i) $t = \varepsilon$

$$\begin{aligned} P_\varepsilon^i &= P'_{\text{check}}.\mathbf{0} \\ P_\varepsilon &= P_{\text{check}}.\omega \end{aligned}$$

(ii) $t = d!v \triangleright \eta :: t'$;

$$P_t^i = \begin{cases} \mathbf{ALLOW}(d,x).\text{if } x = v \text{ then } (cc!\langle \mathbf{DETECTED} \rangle .(P'_{\text{check}}.P'\text{next}.P_{t'}^i) + \mathbf{LOCK}) \text{ else } \mathbf{0} & \text{if } e_i \in \eta \\ P'_{\text{check}}.P'_{\text{next}}.P_{t'}^i & \text{if } e_j \notin \eta \end{cases}$$

$$P_t = cc?\left(x_1, \cdots, x_{|\eta|}\right).P_{\text{check}}.P_{\text{next}}.P_{t'}$$

(iii) $e_j.d?v :: t'$

$$P_t^i = \begin{cases} (d!\langle v \rangle .cc!\langle \mathbf{SENT} \rangle .(P'_{\text{check}}.P'_{\text{next}}.P_{t'}^i + \mathbf{LOCK})) + \mathbf{LOCK} & \text{if } i = j, e_j \in \eta_{\text{in}} \\ P'_{\text{check}}.P'_{\text{next}}.P_{t'}^i & \text{otherwise} \end{cases}$$

$$P_t = cc?(x).cc!\langle \mathbf{CHECK} \rangle .P_{\text{check}}.P_{\text{next}}.P_{t'}$$

Let us explain, at least informally, how the tests we have defined work. The test for the trace ε is trivial; this always can reach a successful configuration after a sequence of τ-extensional transitions, in which each of the nodes $e_1, \cdots, e_k$ send an acknowledgement message to the controller node. Upon receiving the acknowledgement message from each of the nodes above, the controller node cn reports success. While the coordination between nodes $e_1, \cdots, e_k$ and the controller node is not strictly necessary in this case, it reveals to be helpful when dealing with the proofs of technical statements concerning the behaviour of characteristic tests.

In all the other cases we need to check whether the tested network can perform a sequence of weak extensional transitions which induces the trace being tested. To accomplish this task, the test checks each of these transitions in sequence. That is, a characteristic test of the form $\mathcal{T}_{\mu::t}$ first checks whether the testee $\mathcal{M}$ can perform a weak extensional action $\mathcal{M} \stackrel{\mu}{\Longrightarrow} \mathcal{M}'$, for some network $\mathcal{M}'$; if this is the case, then the test proceeds by checking whether the derivative of $\mathcal{M}$, $\mathcal{M}'$, is equipped with the trace t.

Since the nodes $e_1, \cdots, e_k$ can only interact with the tested network partially, the test $\mathcal{T}_t$ implements a coordination protocol in which the controller node cn collects the information about the extensional behaviour

observed by all the nodes above; such a coordination allows the node *cn* to infer the extensional activity performed by the tested network.

The protocol which the test uses for detecting a weak extensional action is composed of three stages, controlled by the node *cn*; in this protocol the nodes of a test use a fresh coordination channel *cc* for information exchange. The different stages of the coordination protocol are described below. Further, the protocol is designed to fail whenever an arbitrary node interacts with the tested network while coordination is in progress.

Detect In this stage only nodes which appear in the action at the head of the trace being tested can broadcast messages to the controller node. If the head of the trace has the form $c!v \triangleright \eta$, these are exactly the nodes included in η, while if the action being considered has the form $m.c?v$ then the only node which is able to broadcast messages to *cn* is exactly m. Such nodes interact with the tested network to ensure that its behaviour is consistent with the weak transition being tested, then they inform the controller node that it can start the next stage of the protocol,

Once the controller node has received a request to proceed from each of these nodes, it broadcasts a value to all the nodes in the testing network, notifying them that the protocol has entered in the second stage

Check In this stage the controller nodes waits to receive a feedback from all the nodes in $\{e_1, \cdot, e_k\}$. Nodes which had an active role in the Detect stage reply to this request only in the case they observed no activity performed by the tested network since they sent the acknowledgement message to the controller node in the first stage. All the other nodes, instead, reply only if they detected that the part of the tested network with which they can interact has exhibited no observational behaviour also in the detect stage. If the controller node receives a reply from all the nodes $e_1, \cdots, e_k$, then it enters in the final stage of the protocol,

Proceed If the protocol enters in this stage, then it is ensured that the tested network has performed the (weak) extensional action which the test $\mathcal{T}_t$ was designed for; thus, it notifies the nodes $e_1, \cdots, e_k$ that the test can start detecting another extensional action.

In this stage the message broadcast by node *cn* is received by every node in $\{e_1, \cdots, e_k\}$ only if the tested network has exhibited no observational behaviour after the Check stage had finished. If this is the case, the protocol terminates and the network start to check for the tail of the trace being tested.

Let us illustrate in deeper detail the behaviour of a characteristic test $\mathcal{T}_t$; here we assume that $t = d!v \triangleright \eta :: t'$, where η is an arbitrary set of nodes included in $\{e_1, \cdots, e_k\}$. Recall that *cc* is a fresh channel, which is not used by the tested network.

Detect Stage When the test starts its computation, each node e in the set η waits to receive a value along channel d, then it compares it with v. If this comparison is successful, then it sends an acknowledgement message to cc, informing it that the correct value has been detected at node e. If the comparison is not successful, or if the node e detects some other activity before sending the acknowledgement message, it deadlocks.

All the other nodes different from *cn* will not be able to perform any broadcast; further, if these nodes detect some activity performed by the network being tested they deadlock. Node *cn*, on the other hand, waits to receive exactly $|\eta|$ acknowledgements value; if this happens, then every node in η has received value v exactly once; this is ensured because only nodes in η are able to broadcast an action when the test $\mathcal{T}_t$ starts its computation.

Check Note that in the detect stage of the experiment $\mathcal{M} \between \mathcal{T}_t$, node *cn* has received no feedback from those nodes which are not included in η. However, such nodes could have been detected some activity performed by the testee; this could mean that the tested network has broadcast some value to some node $e' \notin \eta$ while some of the nodes in η where still waiting to detect the value v to node c.

In this case the network would have performed a sequence of extensional transitions $\mathcal{M} \stackrel{d!v\triangleright\eta'}{\Longrightarrow} \stackrel{d'!w\triangleright\eta^{\times}}{\Longrightarrow} \stackrel{d!v\triangleright\eta''}{\Longrightarrow} \mathcal{M}'$, where $\eta^{\times} \setminus \eta \neq \emptyset$. It is easy to check that this sequence of transitions does not correspond to an extensional transition of the form $\mathcal{M} \stackrel{d!v\triangleright\eta}{\Longrightarrow} \mathcal{M}'$.

To be sure that the scenario above did not happen in the computation, node *cn* broadcasts a message to all the other nodes in nodes($\mathcal{T}_t$), requesting a feedback from them.

At this stage of the computation the following holds:

- A node in η is not deadlocked only if it has only detected value v along channel d exactly once, while all the other nodes are not deadlocked if they detected no activity at all
- Each of the nodes in the network (excluded *cn*) which is still active replies to the controller node with another acknowledgement message. The latter then waits to receive exactly k acknowledgement messages, one for each external node in $\mathcal{M}$.

Note that the controller node *cn* receives exactly k acknowledgement messages if and only if the tested network $\mathcal{M}$ has performed a sequence of weak extensional transitions $\mathcal{M} \stackrel{d!v\triangleright\eta^1}{\Longrightarrow} \cdots \mathcal{M} \stackrel{d!v\triangleright\eta^j}{\Longrightarrow} \mathcal{M}'$, where the sets $\eta_1, \cdots, \eta_j$ are pairwise disjoint and their union corresponds to η. Our non-standard definition of weak extensional actions ensures that this corresponds to the weak extensional transition $\mathcal{M} \stackrel{d!v\triangleright\eta}{\Longrightarrow}$.

Proceed Stage At this point the controller node *cn* decrees that the transition which the test was designed to check has been performed by the tested network. The final step performed by the controller node in the experiment $\mathcal{M} \between \mathcal{T}_t$ is that of broadcasting a message to all the nodes $e_1, \cdots, e_k$, notifying them that they can start testing for the trace t'.

Note that in this stage nodes in $\{e_1, \cdots, e_k\}$ deadlock if they detect that the tested network has exhibited some observational behaviour. This is needed to ensure that no activity can be performed by a network whilst the test has finished detecting for an extensional transition, but it has not started to detect another yet.

The reader should now have a clear idea of the behaviour of characteristic tests of traces. In the remainder of this Section we give a formal proof that each of the tests $\mathcal{T}_t$ captures the associated trace t. First we prove that if $t \in \text{traces}(\mathcal{M})$ then $\mathcal{M} \between \mathcal{T}_t$ has a successful computation, then we prove the converse implication.

Before proving these statements, we will need some technical lemmas which are very useful when performing the proofs. First, let i range over $1, \cdots, k$, t be a trace and η be a set of nodes such that $\emptyset \subset \eta \subset \{e_1, \cdots, e_k\}$. We define the following networks:

$$\begin{aligned}
\mathcal{T}^t_{\text{next}} &= \Gamma_T \triangleright \prod_{i=1}^{k} e_i[\![P'_{\text{next}}.P^i_t]\!] \,|\, cn[\![P_{\text{next}}.P_t]\!] \\
\mathcal{T}^t_{\eta} &= \Gamma_T \prod_{i\in\eta} e_i[\![P'_{\text{next}}.P^i_t]\!] \prod_{i\notin\eta} e_i[\![(cc!\langle \mathbf{CLEAR}\rangle.P'_{\text{next}}.P^i_t) + \mathbf{LOCK}]\!] \\
&\quad |cn[\![cc?\big(x_{|\eta|+1}, \cdots, x_k\big).P_{\text{next}}.P_t]\!] \\
\mathcal{T}^t_{\text{check}} &= \Gamma_T \triangleright \prod_{i=1}^{k} e_i[\![P'_{\text{check}}.P'_{\text{check}}.P^i_t]\!] \,|\, cn[\![P_{\text{check}}.P_{\text{next}}.P_t]\!]
\end{aligned}$$

For any trace t, let Tests_t be the set of networks

$$\text{Tests}_t = \{\mathcal{T}^t_{\text{next}}, \mathcal{T}^t_{\eta}, \mathcal{T}^t_{\text{check}} \mid \emptyset \subset \eta \subset \{e_1, \cdots, e_k\}\}$$

Intuitively, the tests in Tests_t correspond to the different configurations that can be obtained while coordination between nodes is in progress in an experiment. For each of these configurations there are three possibilities:

- proceed to the next step of the coordination progress, which is a test in Tests_t,
- or complete the coordination activity, in which case the resulting test is $\mathcal{T}_t$
- or fail at some node e_i; this can happen only if in an experiment $\mathcal{M} \Vdash \mathcal{T}'$, where $\mathcal{T}' \in \text{Tests}_t$, the network $\mathcal{M}$ broadcasts some value which can be detected by some of the nodes $e_1, \cdots, e_k$, which are affected by the computation by making the code they are running evolve in the deadlocked process $\mathbf{0}$.

Note that, if the coordination protocol fails at some step of a computation, then this cannot correspond to a successful computation. Intuitively speaking, this is because, when testing for a trace, the code running at node *cn* is deterministic; since this is the only node that can enable the success process ω, in a successful computation for an experiment $\mathcal{M} \Vdash \mathcal{T}'$, $\mathcal{T}' \in \text{Tests}_t$, we have that node *cn* has to eventually run the code P_t. Further, for any trace t code P_t first waits to receive a value from each of the nodes in $\{e_1, \cdots, e_k\}$. However, this is possible only if the code running at this node is not the deadlocked process $\mathbf{0}$, that is the protocol has completed the coordination process without errors.

Formally, the following statements hold.

Lemma 4.3.21 (Coordination)**.** Let t be a trace, $\mathcal{M}_0 = \Gamma_M \rhd M_0$ be a composable network, and consider an experiment of the form $\mathcal{M} \Vdash \mathcal{T}_0$, where $\mathcal{T}_0 \in \text{Tests}_t$.

Let also $\mathcal{M}_0 \Vdash \mathcal{T}_0 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{M}_n \Vdash \mathcal{T}_n$ be a successful computation of minimal length for $\mathcal{M} \Vdash \mathcal{T}'$.

Then there exists an index j such that $\mathcal{T}_j = \mathcal{T}_t$, with $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_j$.

Proof. The proof of this statement is very technical in its details. See the Appendix for a detailed outline. □

Corollary 4.3.22. Let t be a trace, $\mathcal{M}_0 = \Gamma_M \rhd M_0$ be a proper network, and consider the experiment $\mathcal{M} \Vdash \mathcal{T}^t_{\text{check}}$. Let also $\mathcal{M}_0 \Vdash \mathcal{T}_0 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{M}_n \Vdash \mathcal{T}_n$ be a successful computation of minimal length for $\mathcal{M} \Vdash \mathcal{T}^t_{\text{check}}$; here we assume that $\mathcal{T}_0 = \mathcal{T}^t_{\text{check}}$.

Then there exists an index j such that $\mathcal{T}_j = \mathcal{T}_t$, with $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_j$.

Proof. This statement is an instance of Lemma 4.3.21, where $\mathcal{T}_0$ is chosen to be exactly $\mathcal{T}^t_{\text{check}}$. □

Lemma 4.3.21 has deep consequences. It says that coordination between nodes in a testing network succeeds only in the case that none of the nodes in the testee has performed some activity which can be detected by the external environment. When this happens, a computation of an experiment can no longer be successful.

Next we show that, when in an experiment of the form $\mathcal{M} \Vdash \mathcal{T}_t$ with $t \neq \varepsilon$ the testing component is in the detecting stage, then the coordination protocol can advance to the next stage without having the code at some of its nodes failing (that is, reaching a deadlocked configuration) only if the tested component $\mathcal{M}$ can perform the weak extensional action corresponding to the head of the trace.

First we focus on the case in which the trace being tested has the form $c!v \rhd \eta :: t'$; in this case, let $\eta^{\times}$, $\eta^{\downarrow}$ be three (possibly empty) sets of nodes which constitute a partition of η; for each possible choice of these sets define the test

$$\begin{aligned}\mathcal{T}^t_{\eta^{\times},\eta^{\rightarrow},\eta^{\downarrow}} \quad = \quad & \Gamma_T \rhd \prod_{i:e_i \in \eta^{\times}} e_i[\![P_t]\!] \mid \prod_{i:e_i \in \eta^{\rightarrow}} e_i[\![cc!\langle \mathbf{DETECTED} \rangle.P'_{\text{next}}.P'_{\text{check}}.P'_{t'}]\!] \mid \\ & \prod_{i:e_i \in \eta^{\rightarrow}} e_i[\![P'_{\text{next}}.P'_{\text{check}}.P'_{t'}]\!] \mid cn[\![cc?\big(x_1, \cdots, x_{|\eta^{\times} \cup \eta^{\rightarrow}|}\big).P_{\text{next}}.P_{\text{check}}.P_{t'}]\!]\end{aligned}$$

We let $\text{DTests}_{c!v \rhd \eta :: t'}$ be the collection of tests defined according to all the possible choices of $\eta^{\times}, \eta^{\rightarrow}$ and $\eta^{\downarrow}$.

Intuitively, these are the configurations that can be encountered in the testing component of a successful computation of an experiment of the form $\mathcal{M} \Vdash \mathcal{T}_{c!v \rhd \eta :: t'}$. Here the set $\eta^{\times}$ contains the nodes in η that are still waiting to detect value v along channel c, nodes in $\eta^{\rightarrow}$ are exactly those nodes which have detected the value, but still haven't sent any acknowledgement to the coordinating node *cn*, while nodes in $\eta^{\downarrow}$ are those nodes which have detected the value and sent the acknowledgement message to *cn*.

We can define a well-founded relation $\prec$ between elements of $\text{DTests}^{t}_{c!v\triangleright\eta}$, based on the contents of the sets $\eta^{\times},\eta^{\rightarrow}$ and $\eta^{\downarrow}$. This relation is defined as the least transitive relation such that if $e_i \in \eta^{\times}$ for some $i = 1,\cdots,k$, then $\langle\eta^{\times},\eta^{\rightarrow},\eta^{\downarrow}\rangle \prec \langle\eta^{\times}\setminus\{e_i\},\eta^{\rightarrow}\cup\{e_i\},\eta^{\downarrow}\rangle$, while if $e_i \in \eta^{\rightarrow}$ then $\langle\eta^{\times},\eta^{\rightarrow},\eta^{\downarrow}\rangle \prec \langle\eta^{\times},\eta^{\rightarrow}\setminus\{e_i\},\eta^{\downarrow}\cup\{e_i\}\rangle$.

The relation $\prec$, defined for triples of sets of nodes, can be now lifted to tests in $\text{DTests}_{c!v\triangleright\eta::t'}$ in the obvious manner. Intuitively, whenever $\mathcal{T} \prec \mathcal{T}'$ for two tests in $\text{DTests}_{c!v\triangleright\eta::t'}$, then if both $\mathcal{T}$ and $\mathcal{T}'$ appear in a successful computation of an experiment $\mathcal{M} \between \mathcal{T}_{c!v\triangleright\eta::t'}$, then $\mathcal{T}$ precedes $\mathcal{T}'$ in the considered computation. Note that the relation $\prec$ is well-founded. Also, it contains a bottom element, which is the test $\mathcal{T}^{t'}_{\eta,\emptyset,\emptyset}$ and a top element, which is the test $\mathcal{T}_{\emptyset,\emptyset,\eta}$. Further, it is straightforward to note that the last test is structurally congruent to $\mathcal{T}^{t'}_{\text{check}}$.

We prove the following useful Lemma for this collection of tests.

Lemma 4.3.23 (Detection)**.** Let t be a trace such that $t = c!v\triangleright\eta :: t'$ for some trace t' and set of nodes η such that $\emptyset \subset \eta \subseteq \{e_1,\cdots,e_k\}$.

Let $\mathcal{M}_0 = \Gamma_M \triangleright M_0$ be a proper network and $\mathcal{T}_0 = \mathcal{T}_{\eta^{\times},\eta^{\rightarrow},\eta^{\downarrow}}$ be a test in DTests_t. Suppose that the experiment $\mathcal{M}_0 \between \mathcal{T}_0$ has a successful computation of the form

$$\mathcal{M}_0 \between \mathcal{T}_0 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{M}_n \between \mathcal{T}_n$$

Then there exists an index $j > 0$ such that $\mathcal{T}_j = \mathcal{T}^{t}_{\text{check}}$, and $\mathcal{M} \overset{\mu}{\Longmapsto} \mathcal{M}_j$, where $\mu = \tau$ if $|\eta^{\times}| = \emptyset$, $c!v\triangleright\eta^{\times}$ otherwise.

Proof. The proof of this statement is similar in style to that of Lemma 4.3.21. However, it contains a significant amount of technical details. See the Appendix for a detailed outline of the proof. □

Corollary 4.3.24. Let t be a trace such that $t = c!v\triangleright\eta :: t'$ for some trace t', where $\emptyset \subset \eta \subseteq \{e_1,\cdots,e_k\}$.

Let $\mathcal{M}_0 = \Gamma_M \triangleright M_0$ be a proper network and $\mathcal{T}_0 = \mathcal{T}_{c!v\triangleright\eta::t'}$. Suppose that the experiment $\mathcal{M}_0 \between \mathcal{T}_0$ has a successful computation of the form

$$\mathcal{M}_0 \between \mathcal{T}_0 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{M}_n \between \mathcal{T}_n$$

Then there exists an index $j > 0$ such that $\mathcal{T}_j = \mathcal{T}^{t}_{\text{check}}$, and $\mathcal{M} \overset{\mu}{\Longmapsto} \mathcal{M}_j$, where $\mu = \tau$ if $|\eta^{\times}| = \emptyset$, $c!v\triangleright\eta^{\times}$ otherwise.

Proof. Note that we have that $\mathcal{T}_{c!v\triangleright\eta::t'} \equiv \mathcal{T}_{\eta,\emptyset,\emptyset} \in \text{DTest}_{c!v\triangleright\eta::t'}$.

Then the corollary is an instance of Lemma 4.3.23, when applied to the test $\mathcal{T}_{c!v\triangleright\eta::t'}$. □

We perform a last check for the family of tests we defined, this time dealing with input actions.

Lemma 4.3.25 (Sending)**.** Let t be a trace such that $t = m.c!v :: t'$ for some trace t', value v, channel c and node m.

Let $\mathcal{M}_0 = \Gamma_M \triangleright M_0$ be a proper network and $\mathcal{T}_0 = \mathcal{T}_{m.c!v::t'}$. Suppose that the experiment $\mathcal{M}_0 \between \mathcal{T}_0$ has a successful computation of the form

$$\mathcal{M}_0 \between \mathcal{T}_0 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{M}_n \between \mathcal{T}_n$$

Then there exists an index $j > 0$ such that $\mathcal{T}_j = \mathcal{T}^{t}_{\text{check}}$, and $\mathcal{M} \overset{m.c?v}{\Longmapsto} \mathcal{M}_j$.

Proof. The proof of this statement is similar in style to that of lemmas 4.3.21 and 4.3.23, and contains a significant less amount of technical details; therefore, it is skipped. □

We are now ready to show that each of the tests $\mathcal{T}_t$ we have defined captures the corresponding trace t. This is stated formally in Proposition 4.3.26

Proposition 4.3.26. Let $\mathcal{M}$ be a proper, finitary network such that $\mathsf{Input}(\mathcal{M}) = \eta_{\text{in}}$, $\mathsf{Output}(\mathcal{M}) = \eta_{\text{out}}$.

Then $t \in \text{traces}(\mathcal{M})$ if and only $\mathcal{M}$ **may-pass** $\mathcal{T}_t$.

Proof. Suppose that $t \in \text{traces}(\mathcal{M})$. We show that there exists a successful computation of $(\mathcal{M} \between \mathcal{T}_t)$ by induction on the trace t.

- $t = \varepsilon$. We have to show that $\mathcal{T}_\varepsilon$ has a successful computation; recall that $\mathcal{T}_\varepsilon = \prod_{i=1}^{k} e_i[\![P'_\varepsilon]\!] \,|\, cn[\![P_\varepsilon]\!]$, where

$$\begin{aligned} P'_\varepsilon &= P'_{\text{check}}.\mathbf{0} \\ P_\varepsilon &= P_{\text{check}}.\mathbf{0} \end{aligned}$$

  Recall that, by Lemma 4.2.5, $(\mathcal{M} \between \mathcal{T}_\varepsilon) = \mathcal{M} \parallel (\mathcal{T}'_\varepsilon)$, where $\mathcal{T}'_\varepsilon = \text{sym}_{\mathcal{M}}(\mathcal{T}_\varepsilon) = \Gamma'_T \rhd T_\varepsilon$. Note that, for any node e_j, $j = 1, \cdots, k$, we have that $\mathcal{T}'_\varepsilon \overset{\mu}{\longmapsto} \mathcal{T}'_1$, where $\mathcal{T}'_1 \equiv \Gamma'_T \rhd \prod_{i=1 i\neq j}^{k} e_i[\![P'_{\text{check}}.\mathbf{0}]\!] \,|\, e_j[\![\mathbf{0}]\!] \,|\, cn[\![cc?(x_2, \cdots, x_k).\omega]\!]$, where $\mu = \tau$ if $\mathsf{Out}_{e_j}(\mathcal{T}'_\varepsilon) = \emptyset$, $c!v \rhd \eta$ if $\emptyset \neq \eta = \mathsf{Out}_{e_j}(\mathcal{T}'_\varepsilon)$.

  In the first case we have that $\mathcal{M} \parallel \mathcal{T}'_\varepsilon \overset{\tau}{\longmapsto} \mathcal{M} \parallel \mathcal{T}'_1$ as a consequence of Proposition 4.2.18, while in the last case we have that $\mathcal{M} \overset{e_i.cc?v}{\longmapsto} \mathcal{M}$, since the coordination channel cc is not used by $\mathcal{M}$; then by Proposition 4.2.18 it follows that $\mathcal{M} \parallel \mathcal{T}'_\varepsilon \overset{\tau}{\longmapsto} \mathcal{M} \parallel \mathcal{T}'_\varepsilon$.

  Thus we have proved that $\mathcal{M} \parallel \mathcal{T}'_\varepsilon \overset{\tau}{\longmapsto} \mathcal{M} \parallel \mathcal{T}'_\varepsilon$. We can iterate this procedure for the remaining $k-1$ nodes in $\{e_1, \cdots, e_k\}$, leading to $\mathcal{M} \parallel \mathcal{T}'_\varepsilon \overset{\tau}{\longmapsto} \mathcal{T}'_\omega$, where $\mathcal{T}'_\omega \equiv \Gamma'_T \rhd \prod_{i=1}^{k} e_i[\![\mathbf{0}]\!] \,|\, cn[\![\omega]\!]$.

  Since $\mathcal{T}'_\omega$ is a successful configuration, it follows that $\mathcal{M} \between \mathcal{T}_\varepsilon$ has a successful configuration, that is $\mathcal{M}$ may-pass $\mathcal{T}_\varepsilon$.

- $t = c!v \rhd \eta :: t'$.

  Since $c!v \rhd \eta :: t' \in \text{traces}(\mathcal{M})$, then there exists a network $\mathcal{M}'$ such that $\mathcal{M} \overset{c!v \rhd \eta}{\Longmapsto} \mathcal{M}'$, and $t' \in \text{traces}(\mathcal{M}')$. By inductive hypothesis we have that $\mathcal{M}'$ may-pass $\mathcal{T}_{t'}$.

  We show that, for any test $\mathcal{T}^{t'}_{\eta^\times, \eta^\rightarrow, \eta^\downarrow} \in \text{DTests}_t$ and network $\mathcal{N}$, if $\mathcal{N} \overset{c!v \rhd \eta^\times}{\Longmapsto} \mathcal{N}'$, then $\mathcal{N} \between \mathcal{T}^{t'}_{\eta^\times, \eta^\rightarrow, \eta^D one} \overset{\tau}{\Longmapsto} \mathcal{N}' \between \mathcal{T}_{t'}$. Since $\mathcal{T}_t \equiv \mathcal{T}^{t'}_{\eta, \emptyset, \emptyset}$ and due to the arbitrariness of the network $\mathcal{N}$, it follows that $\mathcal{M} \between \mathcal{T}_t \overset{\tau}{\Longmapsto} \mathcal{M}' \between \mathcal{T}_{t'}$. For $\mathcal{M}'$ **may-pass** $\mathcal{T}_{t'}$, then it also holds that $\mathcal{M}$ **may-pass** $\mathcal{T}_t$.

  Let us then prove the statement above; this can be done by performing an induction on the proof of the transition $\mathcal{N} \overset{c!v \rhd \eta^\times}{\Longmapsto} \mathcal{N}'$.

  - $\mathcal{N} \overset{\tau}{\Longmapsto} \mathcal{N}_1 \overset{c!v \rhd \eta^\times}{\longmapsto} \mathcal{N}_2 \overset{\tau}{\Longmapsto} \mathcal{N}'$.

    In this case we have that $\mathcal{N} \between \mathcal{T}_{\eta^\times, \eta^\rightarrow, \eta^\downarrow}{}^{t'} \overset{\tau}{\Longmapsto} \mathcal{N}_1 \between \mathcal{T}_t$; note that in the testing component the nodes contained in $\eta^\times$ are waiting to detect value v along channel c; further, node cn is waiting to receive an acknowledgement value from both the nodes in $\eta^\times$ and the nodes in $\eta^\rightarrow$.

    It is easy to show that we have the extensional transition $\mathcal{N}_1 \between \mathcal{T}^{t'}_{\eta^\times, \eta^\rightarrow, \eta^\downarrow} \overset{\tau}{\longmapsto} \mathcal{N}_2 \between \mathcal{T}^{t'}_{\emptyset, (\eta^\times \cup \eta^\rightarrow), \eta^\downarrow}$; in the latter network no node in η is waiting to detect the value v along channel c. Further, the nodes in both the sets $\eta^\times$ and $\eta^\rightarrow$ can send an acknowledgement value (along channel cc) to the coordination node cn.

    Since broadcasts along channel cc do not affect the network $\mathcal{N}_2$, it is straightforward to infer the weak transition $\mathcal{N}_2 \between \mathcal{T}_{\emptyset, (\eta^\times \cup \eta^\rightarrow), \eta^\downarrow}{}^{t'} \overset{\tau}{\Longmapsto} \mathcal{N}_2 \between \mathcal{T}_{\emptyset, \emptyset, \eta}$. In the latter network the testing component is exactly $\mathcal{T}^{t'}_{\text{check}}$.

    At this point we can show that $\mathcal{N}_2 \between \mathcal{T}^{t'}_{\text{check}} \overset{\tau}{\Longmapsto} \mathcal{N}' \between \mathcal{T}^{t'}_{\text{check}}$; now it remains to show that $\mathcal{N}' \between \mathcal{T}^{t'}_{\text{check}} \overset{\tau}{\Longmapsto} \mathcal{N}' \between \mathcal{T}_{t'}$. This step is actually easy, and it can be proved by simply letting $\mathcal{T}'_t$ run the coordination protocol.

  - $\mathcal{N} \overset{c!v \rhd \eta_1}{\Longmapsto} \mathcal{N}'' \overset{c!v \rhd \eta_2}{\Longmapsto} \mathcal{N}'$, where $\eta_1 \cap \eta_2 = \emptyset$, $\eta_1 \cup \eta_2 = \eta^\times$. In this case we can provide an argument similar to the previous case to show that $\mathcal{N} \between \mathcal{T}^{t'}_{\eta^\times, \eta^\rightarrow, \eta^\downarrow} \overset{\tau}{\Longmapsto} \mathcal{N}'' \between \mathcal{T}^{t'}_{(\eta^\times \setminus \eta_1), (\eta^\rightarrow \cup \eta_1), \eta^\downarrow}$. Note

that $\eta^{\times} \setminus \eta_1 = \eta_2$; thus, we can apply the inductive hypothesis to infer that $\mathcal{N}'' \between \mathcal{T}^{t'}_{\eta'',(\eta^{\rightarrow} \cup \eta_1),\eta^{\downarrow}} \stackrel{\tau}{\Longmapsto} \mathcal{N}' \between \mathcal{T}_t$.

Therefore we have that $\mathcal{N} \between \mathcal{T}_{\eta^{\times},\eta^{\rightarrow},\eta^{\downarrow}} \stackrel{\tau}{\Longmapsto} \mathcal{N}' \between \mathcal{T}_t$.

- $t = m.c?v::t'$. In this case we have that $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{M}'$ for some $\mathcal{M}'$ such that $t \in \mathrm{traces}(\mathcal{M}')$. Further, by inductive hypothesis we have that $\mathcal{M}'$ **may-pass** $\mathcal{T}_{t'}$.

  In this case it is rather simple to show that $\mathcal{M} \between \mathcal{T}_t \stackrel{\tau}{\Longmapsto} \mathcal{M}' \between \mathcal{T}_t$; the style of the proof of this statement is similar to the previous cases, and details are therefore omitted.

It remains to prove the converse implication; that is, if $\mathcal{M}$ **may-pass** $\mathcal{T}_t$, then $t \in \mathrm{traces}(\mathcal{M})$. This can be proved by induction on the trace t.

- $t = \varepsilon$. This case is trivial, for $\varepsilon \in \mathrm{traces}(\mathcal{M})$ for any network $\mathcal{M}$.

- $t = c!v \triangleright \eta::t'$.

  Recall that we are assuming that $\mathcal{M}$ **may-pass** $\mathcal{T}_t$.

  By Corollary 4.3.24 it holds that $\mathcal{M} \between \mathcal{T}_t \stackrel{\tau}{\Longmapsto} \mathcal{M}' \between \mathcal{T}^{t'}_{\mathrm{check}}$, with $\mathcal{M} \stackrel{c!v \triangleright \eta}{\Longmapsto} \mathcal{M}'$ and $\mathcal{M}'$ **may-pass** $\mathcal{T}^{t'}_{\mathrm{check}}$. Further, by Corollary 4.3.22 we also have that $\mathcal{M}' \between \mathcal{T}^{t'}_{\mathrm{check}} \stackrel{\tau}{\Longmapsto} \mathcal{M}'' \between \mathcal{T}_{t'}$, and $\mathcal{M}''$ **may-pass** $\mathcal{T}_{t'}$.

  By inductive hypothesis, we have that $t' \in \mathrm{traces}(\mathcal{M}'')$. Further, since $\mathcal{M} \stackrel{c!v \triangleright \eta}{\Longmapsto} \mathcal{M}'' \stackrel{\tau}{\Longmapsto} \mathcal{M}'$, it follows that $t \in \mathrm{traces}(\mathcal{M})$.

- $t = m.c?v :: t'$. This case is similar to the previous one, this time using Lemma 4.3.25 and Corollary 4.3.22.

□

So far we have focused on networks whose input and output interface are fixed; further, we have assumed that the interface of a network $\mathsf{Int}(\mathcal{M})$ is not empty. However, even with these restriction we are now in a situation in which we can prove Theorem 4.3.19.

Proof of Theorem 4.3.19 First consider the case in which $\mathsf{Int}(\mathcal{M}) = \emptyset$. That is, $\mathsf{Input}(\mathcal{M}) = \emptyset$ and $\mathsf{Output}(\mathcal{M}) = \emptyset$.

Since $\mathcal{M} \sqsubseteq_{\mathrm{may}} \mathcal{N}$, it follows from Definition 3.3.6 that $\mathsf{Input}(\mathcal{N}) = \emptyset$, $\mathsf{Output}(\mathcal{M}) = \emptyset$. In this case it is easy to show that $\mathrm{traces}(\mathcal{M}) = \mathrm{traces}(\mathcal{N}) = \{\varepsilon\}$; this is because, in order for the network $\mathcal{M}$ to perform an extensional action of the form $m.c?v$, then there should exist a node $m \in \mathsf{Input}(\mathcal{M})$; conversely, if the network $\mathcal{M} \stackrel{c!v \triangleright \eta}{\Longmapsto} \mathcal{M}'$ for some channel c, value v and non-empty set of nodes η, then $\eta \subseteq \mathsf{Output}(\mathcal{M})$.

Now suppose that $\mathsf{Int}(\mathcal{M}) \neq \emptyset$. Let $\mathsf{Input}(\mathcal{M}) = \eta_{\mathrm{in}}$, $\mathsf{Output}(\mathcal{M}) = \eta_{\mathrm{out}}$. For this particular choice of input and output interface we can associate to each trace t the test $\mathcal{T}_t$ defined in this Section. Further, by Definition 3.3.6 and the assumption that $\mathcal{M} \sqsubseteq_{\mathrm{may}} \mathcal{N}$, we have that $\mathsf{Input}(\mathcal{N}) = \eta_{\mathrm{in}}$, $\mathsf{Output}(\mathcal{N}) = \eta_{\mathrm{out}}$.

Let t be a trace such that $t \in \mathrm{traces}(\mathcal{M})$. Then by Proposition 4.3.26 it follows that $\mathcal{M}$ **may-pass** $\mathcal{T}_t$. Since $\mathcal{M} \sqsubseteq_{\mathrm{may}} \mathcal{N}$, we also have that $\mathcal{N}$ **may-pass** $\mathcal{T}_t$[6]. A final application of Proposition 4.3.26 leads to $t \in \mathrm{traces}(\mathcal{N})$.

We have proved that, for any trace t, if $t \in \mathrm{traces}(\mathcal{M})$ then $t \in \mathrm{traces}(\mathcal{N})$, under the assumption that $\mathcal{M} \sqsubseteq_{\mathrm{may}} \mathcal{N}$; that is, $\mathrm{traces}(\mathcal{M}) \subseteq \mathrm{traces}(\mathcal{N})$. □

[6]Note that both $\mathcal{M} \between \mathcal{T}_t$ and $\mathcal{N} \between \mathcal{T}_t$ are defined

4.3.3 Simple Applications of Trace Inclusion

This Section contains simple applications of our proof technique for relating networks via the may-testing preorder. Before presenting them, we show that the restriction we imposed on composable networks is mandatory for trace inclusion to be sound with respect to the $\sqsubseteq_{\text{may}}$ preorder.

Example 4.3.27 (Violation of Soundness in Non Composable Networks)**.** Consider again the networks $\mathcal{M}, \mathcal{N}$ of Example 3.1.2. Note that we have $\mathcal{N} \notin \mathsf{CNets}$. Recall that $\mathcal{M} = \Gamma_1 \rhd m[\![c!v]\!], \mathcal{N} = \Gamma_2 \rhd n[\![c!v]\!]$; further, $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}) = \emptyset$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N}) = \{n, l\}$.

For network $\mathcal{M}$, it is easy to show that $\text{traces}(\mathcal{M}) = \{\varepsilon, c!v \rhd \{n, l\}\}$; on the other hand, we also have $\text{traces}(\mathcal{N}) = \{\varepsilon, c!v \rhd \{n, l\}\}$. Therefore, if Theorem 4.3.8 were true for well-formed networks in general, we would expect that $\mathcal{M} =_{\text{may}} \mathcal{N}$. This is not the case, for we have already exhibited in Example 3.1.2 a test $\mathcal{T}$ such that $\mathcal{M}$ **may-pass** $\mathcal{T}$, whereas $\neg(\mathcal{N}$ **may-pass** $\mathcal{T})$.

Therefore, if we dropped the assumption that we are working with composable networks, Theorem 4.3.8 would not holds anymore. □

We have already mentioned that, in order to ensure completeness, our definition of weak extensional actions has to be non standard. The following example shows that if we used the standard definition of weak extensional actions it would be possible to exhibit two networks $\mathcal{M}, \mathcal{N}$ such that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, but $\text{traces}(\mathcal{M}) \not\subseteq \text{traces}(\mathcal{N})$.

Example 4.3.28 (Broadcast versus Multicast Revisited)**.** Consider again the networks $\mathcal{M}, \mathcal{N}$ of Example 4.1.6. Recall that there network $\mathcal{N}$ can broadcast value v along channel c to both the external nodes o_1, o_2; on the other hand, network $\mathcal{M}$ can perform two different broadcasts of the same value along channel c; one of them can be detected by node o_1, the other by node o_2.

We have already supplied details that lead to the intuition that $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$ in Example 4.1.6. Here we prove formally this statement.

For network $\mathcal{N}$, it holds $\text{traces}(\mathcal{N}) = \{\varepsilon, c!v \rhd \{o_1, o_2\}$; on the other hand, we have that $\text{traces}(\mathcal{M}) = \{\varepsilon, c!v \rhd \{o_1\}, c!v \rhd \{o_2\}, c!v \rhd \{o_1, o_2\}\}$. Therefore, we have $\text{traces}(\mathcal{N}) \subseteq \text{traces}(\mathcal{M})$; it follows from Soundness (Theorem 4.3.8) that $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$.

Note that the trace $c!v \rhd \{o_1, o_2\}$ in $\text{traces}(\mathcal{M})$ can be derived because $\mathcal{M} \stackrel{c!v \rhd \{o_1, o_2\}}{\Longrightarrow} \mathcal{M}_0$, where $\mathcal{M}_0 = \Gamma_M \rhd m[\![\mathbf{0}]\!] \mid n[\![\mathbf{0}]\!]$. The above transition can be only inferred by using our non-standard definition of weak extensional outputs.

In fact, suppose that we employed the standard definition of weak extensional transitions; that is, $\mathcal{M} \stackrel{c!v \rhd \eta}{\Longrightarrow} \mathcal{M}'$ iff $\mathcal{M} \stackrel{\tau}{\Longrightarrow} \stackrel{c!v \rhd \eta}{\longrightarrow} \stackrel{\tau}{\Longrightarrow} \mathcal{M}'$, and that the definition of $\text{traces}(\mathcal{M})$ had been changed accordingly. In this case it would not be difficult to show that $\text{traces}(\mathcal{M}) = \{\varepsilon, c!v \rhd \{o_1\}, c!v \rhd \{o_2\}\}$, $\text{traces}(\mathcal{N}) = \{\varepsilon, c!v \rhd \{o_1, o_2\}\}$. Here we have that $c!v \rhd \{o_1, o_2\} \in \text{traces}(\mathcal{N})$, but $c!v \rhd \{o_1, o_2\} \notin \text{traces}(\mathcal{M})$; equivalently, $\text{traces}(\mathcal{N}) \not\subseteq \text{traces}(\mathcal{M})$.

Since we already proved the inequality $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$, it follows that in a framework where a standard definition of weak extensional transitions had been employed Completeness, Theorem 4.3.19, would not hold anymore. □

4.4 Full Abstraction for Must-testing

In this Section we provide a characterisation result for the must-testing preorder. It is well known that, in standard process calculi like CCS, in a finitary setting this preorder is characterised by the Smith preorder over *acceptance sets*, quantified over all traces. Roughly speaking, an acceptance set for a process P consists is a collection of (possibly empty) sets of actions; each of such sets take the name of weak ready set. Intuitively, a ready set is defined only for those processes which cannot perform any internal action, and it coincides with the set of visible actions that such a set can perform. Intuitively speaking, when the process P is tested via a

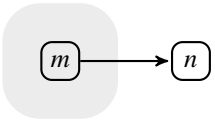


Figure 4.7: A connectivity graph Γ

test T, a computation for $P|T$ can proceed only if the testee can synchronise with the test via a visible action. See [17, 33] for a detailed discussion of the definition of ready sets and acceptance sets.

However, in broadcast calculi, acceptance sets cannot be used to characterise the must-testing preorder. This has already been shown in [24]; here the authors note that, in process-calculi with broadcast communication, it is not necessary for a process which can output a value along a channel to synchronise with a receiver to guarantee that the computation can proceed. This is because, as we have already mentioned in Chapter 2, broadcast is a non-blocking action.

What needs to be taken into account, when providing a characterisation for the must-testing preorder in broadcast process calculi, is the set of processes which cannot evolve autonomously; in our calculus of networks, these are exactly those networks in which no internal activities and no visible broadcasts can be performed. We call them *deadlock networks*.

Another issue that we need to consider, when focusing on must testing, is that of divergent networks; these are networks in which the computation can progress indefinitely. Note that, as we have already shown in Proposition 4.1.4, in our framework computations coincide (up-to structural congruence) with maximal sequences of transitions whose individual actions can correspond to either internal activities or outputs to the external environment. This remark leads to a slightly non-standard definition of diverging networks, which has already been introduced for broadcast calculi in [24].

Using these intuitions, we provide a characterisation of the must-testing preorder in terms of deadlock traces; basically, these are traces in which the possibility of reaching a deadlock network is taken into account. However, our characterisation result holds only for strongly convergent networks, which are defined later in this Section; intuitively a network is strongly convergent if the extensional pLTS it generates does not contain any diverging state.

Before stating the definition of deadlock traces, it is necessary to provide some definitions. First, throughout this Section we only focus on *proper transitions*; a strong transition $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{N}$ is said to be proper if $\mathcal{M}$ is not successful; a weak transition $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{N}$ is proper if and only if we can provide a proof of its derivation by only using strong proper transitions[7].

Next we define the notion of convergence for networks.

Definition 4.4.1 (Convergent Networks)**.** A network $\mathcal{M}$ is *convergent* if there exists no infinite sequence of reductions

$$\mathcal{M}_0 \rightarrow\!\triangleright\, \mathcal{M}_1 \rightarrow\!\triangleright \cdots \rightarrow\!\triangleright\, \mathcal{M}_n \rightarrow\!\triangleright\, \mathcal{M}_{n+1} \rightarrow\!\triangleright \cdots$$

where $\mathcal{M} = \mathcal{M}_0$, and for every index $i \geq 0$ the network $\mathcal{M}_i$ is not successful.

We write $\mathcal{M}\downarrow$ to denote that a network $\mathcal{M}$ is convergent; if the network $\mathcal{M}$ is not convergent we say that it is a *divergent network*, written $\mathcal{M}\uparrow$.

Remark 4.4.2. Note that our definition of divergence uses the reduction relation $\rightarrow\!\triangleright$. In other process calculi divergence is defined as an infinite sequence of silent transitions in the labelled transition semantics; for example, in *CCS* a process P is said to be divergent if an only if there exists an infinite sequence of the form $P \overset{\tau}{\longrightarrow} P_0 \overset{\tau}{\longrightarrow} P_1 \overset{\tau}{\longrightarrow} \cdots$. However, in such process calculi a silent action corresponds, modulo structural congruence, to a reduction $\rightarrow\!\triangleright$. This is not true in our framework; since broadcast is a non-blocking action, a reduction corresponds to either an extensional τ-transition or to an extensional broadcast.

[7] That is, the sequence of strong transitions $\mathcal{M} \overset{\mu_1}{\longmapsto} \cdots \overset{\mu_k}{\longmapsto} \mathcal{N}$ corresponding to the weak transition $\mathcal{M} \overset{\mu}{\Longmapsto} \mathcal{N}$ is made only of proper transitions.

At least intuitively, a network is divergent if it has a computation (in which inputs received from the external environment are not involved) which can procede indefinitely, that is it has an infinite sequence of reductions. Again, since broadcast is a non-blocking action in our framework, and firing a broadcast does not require to involve the external environment, it should be included in the set of possible transitions that can be used to obtain a divergent computation.

Remark 4.4.3 (Distinguishing diverging networks)**.** It is well known that, in CCS and CSP, for any diverging process P and test T it holds that P **must-pass** T if and only if T is successful. Thus, for any process P' it holds that P' **must-pass** T. As a consequence, for any diverging process P and process Q it holds that $P \sqsubseteq_{\text{must}} Q$.

This statement is not true in our calculus of networks. In fact, consider the networks $\mathcal{M} = \Gamma \rhd m[\![c!v.P]\!]$ and $\mathcal{N} = m[\![\mathbf{0}]\!]$, where Γ is the connectivity graph depicted in Figure 4.7 and P is the diverging process $P \Leftarrow \tau.P$. By Definition 4.4.1 it follows that $\mathcal{M}\uparrow$. However, we have that $\mathcal{M} \not\sqsubseteq_{\text{must}} \mathcal{N}$.

In fact, let Γ_n be the least connectivity graph such that $\Gamma_n \vdash n$, and $(\Gamma_n)_E = \emptyset$; consider the network $\mathcal{T} = \Gamma_n \rhd n[\![c?(x).\omega]\!]$. It is straightforward to prove that, despite $\mathcal{M}\uparrow$ and $\mathcal{T}$ is not successful, it holds that $\mathcal{M}$ **must-pass** $\mathcal{T}$. On the other hand, we also have that $\neg(\mathcal{N}$ **must-pass** $\mathcal{T})$, for $\mathcal{N} \nparallel \mathcal{T}$ is unsuccessful and it has no possible reduction.

A slight modification of the Example above shows that in our calculus of networks it is possible to distinguish two diverging networks via the must-pass testing relation. For example, let $\mathcal{M}$ be the network above, and $\mathcal{N}' = \Gamma \rhd m[\![P]\!]$; it is easy to note that both $\mathcal{M}\uparrow$ and $\mathcal{N}'\uparrow$. However $\mathcal{N}' \sqsubseteq_{\text{must}} \mathcal{M}$, whereas $\mathcal{M} \not\sqsubseteq_{\text{must}} \mathcal{N}'$.

□

The remark above states that not all kind of divergences are equivalent in our must-testing framework. Therefore, if we want to provide a general characterisation result for the preorder $\sqsubseteq_{\text{must}}$ via a relation $\mathcal{R}$, we need to define the latter so that it distinguishes between diverging networks according to some criteria; for example, Remark 4.4.3 shows that the pair $\langle \mathcal{M}, \mathcal{N}' \rangle$ has to be included in $\mathcal{R}$, but $\langle \mathcal{N}', \mathcal{M} \rangle$ has not.

At the current state of the art, it remains to be found a relation $\mathcal{R}$ which characterises the must-testing preorder; while we managed to identify a preorder $\leq$ for networks for which a difference with the testing preorder $\sqsubseteq_{\text{must}}$ has not yet been found, we still lack a definitive proof that the two relations coincide. The definition of such a relation and the intuitions which lie behind it are defined in Section 4.4.3.

However, we are able to provide a characterisation result for the $\sqsubseteq_{\text{must}}$ preorder if we focus on strongly convergent networks; these are defined below.

Definition 4.4.4 (Strongly Convergent Networks)**.** Let $\mathcal{M}$ be a testing network; we say that

(i) $\mathcal{M} \Downarrow_\varepsilon$ if $\mathcal{M}\downarrow$,

(ii) $\mathcal{M} \Downarrow_{\mu::t}$, where $\mu \in \mathsf{EAct}$, if $\mathcal{M}\downarrow$ and whenever $\mathcal{M} \stackrel{\mu}{\Longrightarrow} \mathcal{N}$ it holds that $\mathcal{N} \Downarrow_t$.

We write $\mathcal{M} \Uparrow_t$ for $\mathcal{M} \not\Downarrow_t$. We say that a testing network is strongly convergent if, for any finite sequence of extensional actions t, it holds that $\mathcal{M} \Downarrow_t$.

The next notion we need to provide a characterisation of must-testing for networks is that of deadlock networks.

Definition 4.4.5 (Deadlock Networks)**.** A network $\mathcal{M}$ is a *deadlock Network* (or *deadlocked*) if it is not successful and $\mathcal{M} \not\rightarrow_\rhd$.

In other words, a network is deadlocked if it cannot perform any internal activity, nor it can broadcast a value to an arbitrary, non-empty set of external nodes. As we will see, deadlock networks have a crucial role in our characterisation result for the must-testing preorder.

We are now ready to define deadlock traces for networks; this definition is similar in style to Definition 4.3.1, except for the fact that we include the possibility for a network to be unable to evolve autonomously; these are exactly deadlock networks.

Definition 4.4.6 (Deadlock traces of a network)**.** A *Deadlock trace* is an element in the set $(\mathsf{EAct}_\delta)^*$, where $\mathsf{EAct}_\delta = \mathsf{EAct} \cup \{\delta\}$ and δ is a fresh symbol.

For any network $\mathcal{M}$, its set of deadlock traces $\mathrm{DTraces}(\mathcal{M})$ is defined to be the least set such that

1. $\varepsilon \in \mathrm{DTraces}(\mathcal{M})$,
2. If $\mathcal{M}\downarrow$, and $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{M}'$ for some deadlock configuration $\mathcal{M}'$, then $\delta \in \mathrm{DTraces}(\mathcal{M})$
3. If $\mathcal{M}\downarrow$, and $\mathcal{M} \stackrel{c!v\triangleright\eta}{\Longmapsto} \mathcal{M}'$, for some $\mathcal{M}'$ such that $t \in \mathrm{DTraces}(\mathcal{M}')$, then $c!v \triangleright \eta :: t \in \mathrm{DTraces}(\mathcal{M})$
4. If $\mathcal{M}\downarrow$, and $\mathcal{M} \stackrel{m.c?v}{\Longmapsto} \mathcal{M}'$ for some $\mathcal{M}'$ such that $t \in \mathrm{DTraces}(\mathcal{M}')$, then $m.c?v :: t \in \mathrm{DTraces}(\mathcal{M})$

As we already did for traces, we can give a simpler definition of the deadlock traces of a network.

Definition 4.4.7 (Simple deadlock traces of a network)**.** The set of *simple deadlock traces* of a testing network $\mathcal{M}$, denoted as $\mathrm{DTraces}_s(\mathcal{M})$, is the least set such that

(i) $\varepsilon \in \mathrm{DTraces}_s(\mathcal{M})$,

(ii) If $\mathcal{M}$ is deadlocked, then $\delta \in \mathrm{DTraces}_s(\mathcal{M})$

(iii) If $\mathcal{M}\downarrow$ and $\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{M}'$ for some $\mathcal{M}'$ such that $t \in \mathrm{DTraces}_s(\mathcal{M}')$, then $t \in \mathrm{DTraces}_s(\mathcal{M})$

(iv) If $\mathcal{M}\downarrow$ and $\mathcal{M} \stackrel{m.c?v}{\longmapsto} \mathcal{M}'$ for some $\mathcal{M}'$ such that $t \in \mathrm{DTraces}_s(\mathcal{M}')$, then $m.c?v :: t \in \mathrm{DTraces}_s(\mathcal{M})$.

(v) If $\mathcal{M}\downarrow$ and $\mathcal{M} \stackrel{c!v\triangleright\eta}{\longmapsto} \mathcal{M}'$ with $t \in \mathrm{DTraces}_s(\mathcal{M}')$, then $c!v \triangleright \eta :: t \in \mathrm{DTraces}_s(\mathcal{M})$

(vi) If $\mathcal{M}\downarrow$ and $\mathcal{M} \stackrel{c!v\triangleright\eta_1}{\longmapsto} \mathcal{M}'$ for some network $\mathcal{M}'$, non-empty set of nodes η_2 and simple deadlock trace t such that $\eta_1 \cap \eta_2 = \emptyset$ and $c!v \triangleright \eta_2 :: t \in \mathrm{DTraces}_s(\mathcal{M}')$, then $c!v \triangleright (\eta_1 \cup \eta_2) :: t \in \mathrm{DTraces}_s(\mathcal{M})$

As it could be expected, the definition of deadlock traces and simple deadlock traces coincide.

Proposition 4.4.8. For any testing network $\mathcal{M}$, $\mathrm{DTraces}(\mathcal{M}) = \mathrm{DTraces}_s(\mathcal{M})$.

Outline. The proof is analogous to that of Proposition 4.3.6. □

Deadlock traces are closely connected with must-testing. In the may-testing scenario, we have proved that an experiment has a successful computation if and only if it is equipped with a trace of the form $t :: \omega$, where t is an output trace. Similarly, we can relate unsuccessful computations of experiments with either divergent behaviour or with some failure trace. In the following Proposition the notion of output deadlock trace is straightforward; these are exactly those output traces whose all occurrences coincide with broadcast actions. Also, an unsuccessful deadlock trace is a trace of the form $t :: \delta$, where t is an output trace.

Proposition 4.4.9 (Unsuccessful computation)**.** Any testing network $\mathcal{M}$ has an unsuccessful computation iff either $\mathcal{M}\uparrow$ or there exists an unsuccessful trace $t \in \mathrm{DTraces}(\mathcal{M})$.

Outline. The proof of this proposition is similar in style to that of Proposition 4.3.3. Here we only provide high-level details of the complete proof.

Let $\mathcal{M}_0$ be a network with an unsuccessful computation

$$\mathcal{M}_0 \twoheadrightarrow \mathcal{M}_1 \cdots \twoheadrightarrow \mathcal{M}_n \twoheadrightarrow \mathcal{M}_{n+1} \twoheadrightarrow \cdots$$

There are two possible cases:

- The computation above has infinite length. Further, since we are assuming that it is an unsuccessful computation, we have that for any index $i \geq 0$, $\mathcal{M}_i$ is not successful. By Definition 4.4.1, it follows that $\mathcal{M}_0 \uparrow$.

- The length of the computation above is finite; that is, we can rewrite it as

$$\mathcal{M}_0 \rightarrow_{\triangleright} \cdots \mathcal{M}_k$$

for some index $k \geq 0$. Since a computation is a maximal computation fragment, we have that $\mathcal{M}_k \not\rightarrow_{\triangleright}$; further, since by hypothesis the computation above is not successful, we have that $\mathcal{M}_k$ is not a successful configuration. It follows from Definition 4.4.5 that $\mathcal{M}_k$ is a deadlock configuration, hence $\delta \in \mathrm{DTraces}(\mathcal{M}_k)$. Now we can proceed as in the proof of Proposition 4.3.3 to show that there exists an output deadlock trace t' such that $t'::\delta \in \mathrm{DTraces}(\mathcal{M}_0)$

For the converse implication, suppose first that $\mathcal{M}_0$ is a diverging network. By Definition 4.4.1 we have that there exists an infinite computation

$$\mathcal{M}_0 \rightarrow_{\triangleright} \cdots \mathcal{M}_n \rightarrow_{\triangleright} \mathcal{M}_{n+1} \rightarrow_{\triangleright} \cdots$$

such that, for any index $i \geq 0$, $\mathcal{M}_i$ is not successful. This is obviously an unsuccessful computation.

Now suppose that there exists an output deadlock trace t' such that $t'::\delta \in \mathrm{DTraces}(\mathcal{M}_0)$. Recall also that deadlock traces are defined by using only proper transitions. We can proceed as in the proof of Proposition 4.3.3 to show that the trace $t'::\delta$ is associated with a computation fragment

$$\mathcal{M}_0 \rightarrow_{\triangleright} \cdots \rightarrow_{\triangleright} \mathcal{M}_k$$

such that $\mathcal{M}_k$ is a deadlock configuration. Since we are only considering proper transitions, it follows that for any index i such that $0 \leq i < k$ then $\mathcal{M}_i$ is not successful. Further, $\mathcal{M}_k$ is not successful by definition of deadlock configuration. It remains to note that this computation fragment is maximal, for $\mathcal{M}_k \not\rightarrow_{\triangleright}$, hence it is an unsuccessful computation. □

Note that we have related deadlock traces with unsuccessful computations. Further, Proposition 4.4.9 also remarks that a divergent network always has an unsuccessful computation. These two facts have deep consequences on our characterisation result for the $\sqsubseteq_{\mathrm{must}}$ preorder; in fact, the presence of divergence or of a trace of the form $t::\delta$ in an experiment $\mathcal{M} \between \mathcal{T}$ coincide with the the predicate $\neg(\mathcal{M}$ **must-pass** $\mathcal{T})$, rather than with $\mathcal{M}$ **must-pass** $\mathcal{T}$. Therefore, when focusing on finitary, strongly convergent networks, we expect that deadlock traces inclusion characterise the inverse of the may-testing preorder, which we denote as $\sqsupseteq_{\mathrm{must}}$, or equivalently that the preorder $\sqsubseteq_{\mathrm{must}}$ is characterised by the relation $\supseteq$ over deadlock traces.

Theorem 4.4.10 (Characterisation of Must-Testing)**.** Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent proper networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$ and $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$.

Then $\mathcal{M} \sqsubseteq_{\mathrm{must}} \mathcal{N}$ if and only if $\mathrm{DTraces}(\mathcal{M}) \supseteq \mathrm{DTraces}(\mathcal{N})$.

Proof. We actually prove that $\mathrm{DTraces}(\mathcal{M}) \supseteq \mathrm{DTraces}(\mathcal{N})$ if and only if $\mathcal{M} \sqsupseteq_{\mathrm{must}} \mathcal{N}$. As we already did in the may-testing case, the proof of the Theorem is split in two parts; Soundness, Theorem 4.4.1 and Completeness, Theorem 4.4.2. □

4.4.1 Soundness

The aim of this Section is to prove the following Proposition.

Theorem 4.4.11 (Soundness for Must-testing)**.** For any two finitary, strongly convergent networks $\mathcal{M}, \mathcal{N}$ such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}), \mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$ and $\mathrm{DTraces}(\mathcal{M}) \subseteq \mathrm{DTraces}(\mathcal{N})$, then $\mathcal{M} \sqsupseteq_{\mathrm{must}} \mathcal{N}$.

The proof of this result is similar in style to that of soundness for the may-testing preorder, Theorem 4.3.8. We prove that in a strongly convergent setting deadlock trace inclusion is compositional with respect to the extension operator $\between$ by exploiting the compositional and decompositional results for extensional actions to

define zipping, unzipping and switching results for the symmetric operator $\|$; however, some complications arise for strong convergence of networks is not preserved by the testing operator $\natural$, as Example 4.4.12 shows.

Example 4.4.12 (Non-compositionality of Strong Convergent Networks)**.** Let Γ be the least connectivity graph such that $\Gamma \vdash m,n$ and $\Gamma \vdash m \leftrightarrow n$. Consider the network $\mathcal{M} = \Gamma \rhd m[\![P]\!]$, where P is the recursive process $P \Leftarrow c?(x).c!\langle v\rangle.P$. It is immediate to notice that $\mathcal{M}$ is a strongly convergent network.

Consider also the network $\mathcal{T} = \Gamma_T \rhd n[\![c!\langle v\rangle.P]\!]$, where Γ_T is the connectivity graph consisting of the sole node n; again, this network is strongly convergent. Further, the composite network $\mathcal{M} \natural \mathcal{T}$ is defined.

However, the latter network is not strongly convergent, for $(\mathcal{M} \natural \mathcal{T})\uparrow$; in fact, it is possible to derive the infinite sequence of transitions

$$(\mathcal{M} \natural \mathcal{T}) \stackrel{\tau}{\Longmapsto} (\mathcal{M}' \natural \mathcal{T}') \stackrel{\tau}{\Longmapsto} (\mathcal{M} \natural \mathcal{T}) \stackrel{\tau}{\Longmapsto} (\mathcal{M}' \natural \mathcal{T}') \stackrel{\tau}{\Longmapsto} \cdots$$

where $\mathcal{M}' = \Gamma \rhd m[\![c!\langle v\rangle.P]\!]$ and $\mathcal{T}' = \Gamma_T \rhd n[\![P]\!]$. □

Though strong convergence is not preserved by the extension operator $\natural$, we can rely on an auxiliary result to prove Theorem 4.4.11.

Proposition 4.4.13. Let $\mathcal{M},\mathcal{N}$ be two finitary, strongly convergent networks such that $\mathrm{DTraces}(\mathcal{M}) \supseteq \mathrm{DTraces}(\mathcal{N})$; also, assume that $\mathcal{T}$ is a finitary, testing network such that $(\mathcal{N} \natural \mathcal{T})\uparrow$.

Then, if $(\mathcal{N} \natural \mathcal{T})$ is defined, it also holds $(\mathcal{N} \natural \mathcal{T})\uparrow$.

The assumptions that networks $\mathcal{M},\mathcal{N}$ and $\mathcal{T}$ are finitary is necessary in Proposition 4.4.13.

In order to prove Proposition 4.4.13, we need some technical lemmas. The first one is a result from Graph Theory; this is *König's Lemma*, which is stated below.

Lemma 4.4.14 (König's Lemma)**.** Let G be a connected graph with infinitely many vertices, each of which has finite degree. Then G contains an infinite simple path.

Proof. See [41], Lemma (2.3). □

In particular, we are interested in the more specific case in which the graph G is a directed tree rooted in some vertex v.

Corollary 4.4.15. Let G be a connected tree with infinitely many vertices, each of which has finite degree, and let v be the root of such a tree; then G has an infinite simple path rooted in v.

Proof. In this case König's Lemma ensures that the directed tree G contains an infinite simple path p rooted in some vertex v'; since G is a connected tree, there exists also a finite path p' from its root v to the vertex v'; now it is possible to concatenate the paths p' and p to obtain an infinite simple path of G rooted in v. □

The main use of Corollary 4.4.15 lies in the following Lemma.

Lemma 4.4.16. Let $\mathcal{M}$ be a network; for any extensional action $\mu \in \mathsf{EAct}_\tau$ we define the length of an extensional transition of the form $\mathcal{M} \stackrel{\mu}{\Longmapsto} \mathcal{M}'$ as the number of strong extensional transitions used in its derivation, and we denote it as $\mathrm{length}(\mathcal{M} \stackrel{\mu}{\Longmapsto} \mathcal{M}')$. A similar definition is given for sequences of reductions of the form $\mathcal{M} \twoheadrightarrow^* \mathcal{M}'$.

(i) if $\mathcal{M}$ is finitary and strongly convergent, for any extensional action μ the quantity $\sup\{\mathrm{length}(\mathcal{M} \stackrel{\mu}{\Longmapsto} \mathcal{M}')$ for some $\mathcal{M}'\}$ is defined and finite

(ii) if $\mathcal{M}$ is finitary and convergent, the quantity $\sup\{\mathrm{length}(\mathcal{M} \twoheadrightarrow^* \mathcal{M}')$ for some $\mathcal{M}'\}$ is defined and finite

Proof. The main idea used to prove the Lemma is that of building the *transition tree* for the network $\mathcal{M}$; in the case of a weak τ-transition, this is a directed tree whose vertices are weak extensional transitions for the network $\mathcal{M}$, and an edge from a transition $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{M}'$ and $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{M}_1$ is defined if $\mathcal{M}' \stackrel{\tau}{\longmapsto} \mathcal{M}_1$. Then the statement follows from a direct application of Corollary 4.4.15; note that the application of this corollary require that each of the vertices in the tree that we have defined has finite degree. This is true, for we are focusing on finitary networks. See Appendix A, Section A.5 for details. □

The second technical statement we need to prove Proposition 4.4.13 concerns finite sequences of extensional transitions.

Lemma 4.4.17. Let $\mathcal{M}$ be a finitary, strongly convergent network, and $\{\mathcal{N}_j\}_{j\in J}$ be a finite collection of finitary, strongly convergent networks such that, for any $j, j' \in J$, $\mathcal{N}_j = \Gamma_N \rhd N_j$, $\mathcal{N}_{j'} = \Gamma_N \rhd N_{j'}$ and $\mathrm{nodes}(N_j) = \mathrm{nodes}(N_{j'})$. Assume that for any $j \in J$, $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}_j)$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N}_j)$; further, assume that $\mathrm{DTraces}(\mathcal{M}) \subseteq \bigcup_{j\in J} \mathrm{DTraces}(\mathcal{N}_j)$. Finally, suppose that $\mathcal{G} = \Gamma_G \rhd n[\![P]\!]$ is generating network such that $(\mathcal{M} \Vdash \mathcal{T})$ is defined and $\mathcal{N}_j \Vdash \mathcal{T}$ is defined for any $j \in J$.

Then, if there exists a sequence of extensional transitions of the form

$$(\mathcal{M} \Vdash \mathcal{T}) \stackrel{\mu_1}{\longmapsto} (\mathcal{M}^1 \Vdash \mathcal{T}_1) \stackrel{\mu_2}{\longmapsto} \cdots \stackrel{\mu_k}{\longmapsto} (\mathcal{M}^k \Vdash \mathcal{T}^k)$$

there also exists an index $j \in J$ such that $\mathcal{N}_j \Vdash \mathcal{T}$ such that

$$(\mathcal{N}_j \Vdash \mathcal{T}) \stackrel{\mu_1}{\Longmapsto} (\mathcal{N}_j^1 \Vdash \mathcal{T}_1) \stackrel{\mu_2}{\Longmapsto} \cdots \stackrel{\mu_n}{\Longmapsto} (\mathcal{N}_j^k \Vdash \mathcal{T}^k)$$

Proof. The proof of this statement can be obtained by performing an induction over the length n of the computation fragment of the network $\mathcal{M}_i \Vdash \mathcal{T}$, by using the (strong) decomposition, (weak) composition and connectivity change results proved in Section 4.2. However, some complications arise for we are considering finite collections of networks. See the Appendix for a detailed outline of the proof. □

Corollary 4.4.18. Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$, and let $\mathcal{T}$ be a finitary network such that $(\mathcal{M} \Vdash \mathcal{T})$ and $(\mathcal{N} \Vdash \mathcal{T})$ are defined.

If there exists a sequence of extensional transitions of the form

$$(\mathcal{M} \Vdash \mathcal{T}) \stackrel{\mu_1}{\longmapsto} (\mathcal{M}^1 \Vdash \mathcal{T}_1) \stackrel{\mu_2}{\longmapsto} \cdots \stackrel{\mu_k}{\longmapsto} (\mathcal{M}^k \Vdash \mathcal{T}^k)$$

of length $n_{\mathcal{M}}$, then there also exists a sequence of weak extensional transitions of the form

$$(\mathcal{N} \Vdash \mathcal{T}) \stackrel{\mu_1}{\Longmapsto} (\mathcal{N}^1 \Vdash \mathcal{T}_1) \stackrel{\mu_2}{\Longmapsto} \cdots \stackrel{\mu_n}{\Longmapsto} (\mathcal{N}^k \Vdash \mathcal{T}^k)$$

whose length is $n_{\mathcal{N}}$.

Proof. First, suppose that $\mathcal{T}$ is a generating network. In this case the statement is a specific case of Lemma 4.4.17, where the collection of networks $\{\mathcal{N}_j\}_{j\in J}$ is the singleton set $\{\mathcal{N}\}$.

In the case where $\mathcal{T}$ is not a generating network, the proof is performed by using the network induction principle, Theorem 3.2.7. □

Remark 4.4.19. The reader could argue that Corollary 4.4.18 could have been proved directly, rather than relying on the more complicated Lemma 4.4.17. However, the proof of the latter statement is performed by induction on the length of the sequence of extensional transitions for the network $\mathcal{M}$, and we need the assumption that $\mathrm{DTraces}(\mathcal{M}) \subseteq \bigcup_{j\in J} \mathrm{DTraces}(\mathcal{N}_j)$ to apply the inductive hypothesis.

If we tried to prove directly Corollary 4.4.18, we should have proved that, whenever $\mathcal{M} \stackrel{\mu}{\Longmapsto} \mathcal{M}^1$ then there exists a network $\mathcal{N}^1$ such that $\mathcal{N} \stackrel{\mu}{\Longmapsto} \mathcal{N}^1$ and $\mathrm{DTraces}(\mathcal{M}^1) \subseteq \mathrm{DTraces}(\mathcal{N}^1)$. This statement, however, is not true.

Let in fact Γ be the least connectivity graph such that $\Gamma \vdash m, n$ and $\Gamma \vdash m \to n$, and consider the networks $\mathcal{M} = \Gamma \rhd m[\![c!v.(c!w + c!u)]\!]$, $\mathcal{N} = \Gamma \rhd m[\![c!v.c!w + c!v.c!u]\!]$. Fur such networks it holds that $\mathrm{DTraces}(\mathcal{M}) \subseteq \mathrm{DTraces}(\mathcal{N})$, and the only possible transition for $\mathcal{M}$ is given by $\mathcal{M} \xmapsto{c!v \rhd n} \mathcal{M}'$, where $\mathcal{M}' = \Gamma \rhd m[\![c!w + c!u]\!]$. For network $\mathcal{N}$, there are two possible transitions: $\mathcal{N} \xmapsto{c!v \rhd n} \mathcal{N}'$ and $\mathcal{N} \xmapsto{c!v \rhd n} \mathcal{N}''$, where $\mathcal{N}' = \Gamma \rhd m[\![c!w]\!]$ and $\mathcal{N}'' = \Gamma \rhd m[\![c!w]\!]$. Now it is trivial to note that $\mathrm{DTraces}(\mathcal{M}') \not\subseteq \mathrm{DTraces}(\mathcal{N}')$ and $\mathrm{DTraces}(\mathcal{M}') \not\subseteq \mathrm{DTraces}(\mathcal{N}'')$. However, $\mathrm{DTraces}(\mathcal{M}') \subseteq (\mathrm{DTraces}(\mathcal{N}') \cup \mathrm{DTraces}(\mathcal{N}''))$.

Proof of Proposition 4.4.13 Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent networks; let $\mathcal{T}$ be a finitary network such that $(\mathcal{M} \between \mathcal{T})$ and $(\mathcal{N} \between \mathcal{T})$ are defined, and $(\mathcal{M} \between \mathcal{T}) \uparrow$. Suppose that $\mathrm{DTraces}(\mathcal{M}) \subseteq \mathrm{DTraces}(\mathcal{N})$.

Consider a diverging computation of the form

$$(\mathcal{M} \between \mathcal{T}) \twoheadrightarrow (\mathcal{M}_1 \between \mathcal{T}_1) \twoheadrightarrow \cdots$$

which exists by hypothesis. Since a reduction coincides (up-to structural congruence) with either an extensional τ-transition or an output transition, we can employ both propositions 4.1.4 and 4.4.18 to infer that, for any fragment of the above computation

$$(\mathcal{M} \between \mathcal{T}) \twoheadrightarrow (\mathcal{M}_1 \between \mathcal{T}_1) \twoheadrightarrow \cdots \twoheadrightarrow (\mathcal{M}_k \between \mathcal{T}_k)$$

there exists a computation fragment of the form

$$(\mathcal{N} \between \mathcal{T}) \twoheadrightarrow^* (\mathcal{N}_1 \between \mathcal{T}_1) \twoheadrightarrow^* \cdots \twoheadrightarrow^* (\mathcal{N}_k \between \mathcal{T}_k) \tag{4.5}$$

For each index k, choose a computation fragment c_k as in Equation 4.5; note that it is easy to show that such c_k can be chosen in a way such that the sequence $\{\mathrm{length}(c_k)\}_{k \geq 0}$ is increasing monotone. This is because it is always possible to choose c_{k+1} is a way such that c_k is a prefix for it. Next we show that such a sequence is *strictly* increasing monotone. In fact, let $k \geq 0$, and consider the computation fragment

$$(\mathcal{M} \between \mathcal{T}) \twoheadrightarrow (\mathcal{M}_1 \between \mathcal{T}_1) \twoheadrightarrow \cdots \twoheadrightarrow (\mathcal{M}_k \between \mathcal{T}_k)$$

It follows that there exists an index $k' \geq k$ such that

$$(\mathcal{M} \between \mathcal{T}) \twoheadrightarrow (\mathcal{M}_1 \between \mathcal{T}_1) \twoheadrightarrow \cdots \twoheadrightarrow (\mathcal{M}_k \between \mathcal{T}_k) \twoheadrightarrow \cdots \twoheadrightarrow (\mathcal{M}_{k'} \between \mathcal{T}_{k'})$$

such that, for some index k'' with $k \leq k'' < k'$ it holds that $\mathcal{M}_{k''} \xmapsto{m.c?v} \mathcal{M}_{k'}$. If this were not the case we would have that for any $k' \geq k$ either $\mathcal{M}_{k'} \xmapsto{\tau} \mathcal{M}_{k'+1}$ or $\mathcal{M}_{k'} \xmapsto{c!v \rhd \eta} \mathcal{M}_{k'+1}$, which leads to $\mathcal{M}_{k'} \uparrow$, thus contradicting the assumption that $\mathcal{M}$ is strongly convergent. Thus, in the computation fragment

$$(\mathcal{N} \between \mathcal{T}) \twoheadrightarrow^* (\mathcal{N}_1 \between \mathcal{N}_1) \twoheadrightarrow^* \cdots \twoheadrightarrow^* (\mathcal{N}_k \between \mathcal{T}_k) \twoheadrightarrow^* \cdots \twoheadrightarrow^* (\mathcal{N}_{k'} \between \mathcal{T}_{k'})$$

we have that $\mathcal{N}_{k''} \stackrel{m.c?v}{\Longmapsto} \mathcal{N}_{k''+1}$; informally speaking, in the computation fragment above the network $\mathcal{N}_{k''}$ performs at least one transition before reaching the state $\mathcal{N}_{k'}$. That is, $\mathrm{length}(c_k) < \mathrm{length}(c_{k'})$.

Thus we have shown that the sequence $\{\mathrm{length}(c_k)\}_{k \geq 0}$ is unbounded, that is $\sup\{\mathrm{length}(c_k)\}_{k \geq 0} = \infty$. Now it follows from Corollary 4.4.16 that $(\mathcal{N} \between \mathcal{T}) \uparrow$. □

In the following we focus on compositional properties of deadlock traces in the case that the composition of two finitary, strongly convergent network gives rise to a network which is again both finitary and strongly convergent. In particular, we show that in this case deadlock trace inclusion is preserved by the testing operator $\between$.

Proposition 4.4.20 (Preservation of Deadlock Traces Inclusion)**.** Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent networks; suppose that $\mathcal{M} \between \mathcal{T}$ and $\mathcal{N} \between \mathcal{T}$ are defined for some finitary, strongly convergent network $\mathcal{T}$. Finally,

assume that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}), \mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$ and $\mathrm{DTraces}(\mathcal{M}) \subseteq \mathrm{DTraces}(\mathcal{N})$.

Then it follows that $\mathrm{DTraces}(\mathcal{M} \divideontimes \mathcal{T}) \subseteq \mathrm{DTraces}(\mathcal{N} \divideontimes \mathcal{T})$

The proof of this statement requires the definition of zipping, unzipping and switching functions for deadlock traces. These are similar to the respective functions defined for traces 4.3.1; indeed, they can be seen as an extension of such functions, where the cases for handling the element δ, which is not present in traces, are introduced. Further, all the constraints dealing with the symbol ω, which can appear in traces but not in deadlock traces, are dropped. For each of these function we prove the same results we have stated for their respective definition of Section 4.3.1; as we could expect, these allow us to prove Proposition 4.4.20. In the following we always assume that networks are both finitary and strongly convergent; further, we also assume that the composition of two networks lead to a finitary, strongly convergent network.

First we provide the definition of zipping for deadlock traces; as we did in Section 4.3.1, for the moment we focus on the case where only generating networks are composed with a network $\mathcal{M}$ via the $\|$ operator.

Definition 4.4.21 (Zipping of Deadlock Traces)**.** Let $\mathcal{M}$ be a testing network, $\mathcal{G}$ be a generating network such that $\mathcal{M} \| \mathcal{G}$ is defined. For any deadlock traces t, t', the function $\mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, t')$ is defined by letting

(i) $\delta \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(\delta, \delta)$

(ii) in all the other cases, $\mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, t')$ is defined as in Definition 4.3.9

Proposition 4.4.22 (Zipping)**.** Let $\mathcal{M}$ be a strongly convergent, finitary network, $\mathcal{G}$ be a finitary, generating network. Suppose that $\mathcal{M} \| \mathcal{G}$ is defined and strongly convergent. Then, for any deadlock traces $t \in \mathrm{DTraces}(\mathcal{M}), t' \in \mathrm{DTraces}(\mathcal{G})$, and $s \in \mathrm{zip}^{\mathcal{G}}_{\mathcal{M}}(t, t')$, it holds $s \in \mathrm{DTraces}(\mathcal{M} \| \mathcal{G})$.

Outline. The proof of this statement is similar to that of Proposition 4.3.10. We only provide the details for the symbol δ, which is not included in the alphabet used for traces.

Suppose that $\delta \in \mathrm{DTraces}(\mathcal{M}), \delta \in \mathrm{DTraces}(\mathcal{G})$. We need to show that $\delta \in \mathrm{DTraces}(\mathcal{M} \| \mathcal{G})$. By definition, it follows that $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}'$ for some deadlock configuration $\mathcal{M}'$, and $\mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{G}'$ for some deadlock configuration $\mathcal{G}'$. We can employ Lemma 4.2.21 twice to obtain the transition $\mathcal{M} \| \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \| \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \| \mathcal{G}'$.

It remains to note that $\mathcal{M}' \| \mathcal{G}'$ is a deadlock configuration, from which it follows by definition that $\delta \in \mathrm{DTraces}(\mathcal{M} \| \mathcal{G})$. To this end, note that if $\mathcal{M}' \| \mathcal{G}' \overset{\tau}{\longmapsto} \mathcal{N}$ for some network $\mathcal{N}$, we would be able to apply Proposition 4.2.12 to infer one of the following: $\mathcal{M}' \overset{\tau}{\longmapsto}$, $\mathcal{G}' \overset{\tau}{\longmapsto}$, $\mathcal{M}' \overset{c!v \triangleright \eta}{\longmapsto}$ or $\mathcal{G}' \overset{c!v \triangleright \eta}{\longmapsto}$. Each of the statements above contradicts the hypothesis that both $\mathcal{M}'$ and $\mathcal{G}'$ are deadlock configurations; hence it is not possible that $\mathcal{M}' \| \mathcal{G}' \overset{\tau}{\longmapsto} \mathcal{L}$ for some network $\mathcal{L}$. Similarly, it can be proved that $\mathcal{M}' \| \mathcal{G}'$ cannot perform an extensional output action. □

Next we turn our attention to the unzipping function. Again, we have only to supply the details to handle the new element δ introduced for failure traces

Definition 4.4.23 (Unzipping of Deadlock Traces)**.** Let $\mathcal{M}$ be a network, $\mathcal{G}$ be a generating network, and suppose $(\mathcal{M} \| \mathcal{G})$ is defined. For any deadlock trace t, we define the set of pairs of deadlock traces $\mathrm{unzip}^{\mathcal{G}}_{\mathcal{M}}(t)$ to be the least set which satisfies the following conditions:

(i) $\langle \delta, \delta \rangle \in \mathrm{unzip}^{\mathcal{G}}_{\mathcal{M}}(\delta)$,

(ii) all the other cases are defined as in Definition 4.3.11

Proposition 4.4.24 (Unzipping)**.** Let $\mathcal{M}, \mathcal{G}$ be respectively a testing network and a generating network such that $\mathcal{M} \| \mathcal{G}$ is defined.

Then, for any trace $s \in \mathrm{traces}(\mathcal{M} \| \mathcal{G})$ there exists a pair $\langle t, t' \rangle \in \mathrm{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t \in \mathrm{traces}(\mathcal{M}), t' \in \mathrm{traces}(\mathcal{G})$.

Proof. Practically analogous to that of Proposition 4.3.13, where it has to be noted that if $\mathcal{M} \parallel \mathcal{G}$ is a deadlock configuration then both $\mathcal{M}$ and $\mathcal{G}$ are deadlocked.

In fact, if it were $\mathcal{M} \overset{\tau}{\longmapsto}$, it would be easy to infer $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\longmapsto}$; similarly, if $\mathcal{G} \overset{\tau}{\longmapsto}$ it would follows $\mathcal{M} \parallel \mathcal{G} \overset{\tau}{\longmapsto}$; in both cases the hypothesis that $\mathcal{M} \parallel \mathcal{G}$ is deadlocked is contradicted.

Further, if we assume that $\mathcal{M} \overset{c!v \triangleright \eta}{\longmapsto}$, it would be possible to show that $(\mathcal{M} \parallel \mathcal{G}) \overset{\tau}{\longmapsto}$ (in the case $\eta = \text{nodes}(\mathcal{G}) = \{n\}$) or $\mathcal{M} \parallel \mathcal{G} \overset{c!v \triangleright (\eta \setminus \{n\})}{\longmapsto}$ (in the case $\eta \neq \{n\}$). See Proposition 4.2.19 for details. A similar analysis can be applied if we assume that $\mathcal{G} \overset{c!v \triangleright \eta}{\longmapsto}$. Again, both these statements contradict the hypothesis that $\mathcal{M} \parallel \mathcal{G}$ is a deadlock configuration. □

The last function that we need to define to prove Proposition 4.4.20 is switching.

Definition 4.4.25. Let Γ_1, Γ_2 be two connectivity graphs, P be a process and n be a node such that $\mathcal{G} = \Gamma_1 \triangleright n[\![P]\!]$, $\mathcal{H} = \Gamma_2 \triangleright n[\![P]\!]$

For any trace t, we define the set $\text{switch}_{\mathcal{H}}(t)$ as the smallest set such that:

(i) $\delta \in \text{switch}_{\mathcal{H}}(\delta)$

(ii) in all the other cases, the function is defined as in Definition 4.3.14

Proposition 4.4.26. [Switching] Let $\mathcal{M}, \mathcal{N}$ be two networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}), \mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$. Suppose also that $\mathcal{G}$ be a generating network such that both $\mathcal{M} \between \mathcal{G}$ and $\mathcal{N} \between \mathcal{G}$ are defined.

Let $\mathcal{H} = \text{sym}_{\mathcal{M}}(\mathcal{G})$, $\mathcal{K} = \text{sym}_{\mathcal{N}}(\mathcal{G})$. Then, for any trace s and $\langle t, t' \rangle \in \text{unzip}^{\mathcal{H}}_{\mathcal{M}}(s)$ such that $t \in \text{DTraces}(\mathcal{M})$, there exists a trace $t'' \in \text{switch}_{\mathcal{K}}(t')$ such that $s \in \text{zip}^{\mathcal{H}}_{\mathcal{N}}(t, t'')$.

Proof. Identical to that of Proposition 4.4.26 □

We have concluded the list of definitions, together with their properties, that we need to prove Proposition 4.4.20.

Proof of Proposition 4.4.20 The proof of this statement is identical to that of Corollary 4.3.18. □

The two propositions that we have proved in this Section, namely Proposition 4.4.13 and Proposition 4.4.20, enable us to prove Soundness of deadlock traces inclusion with respect to the opposite of the testing preorder $\sqsubseteq_{\text{must}}$.

Proof of Theorem 4.4.11 Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent networks, and suppose that $\text{DTraces}(\mathcal{N}) \subseteq \text{DTraces}(\mathcal{M})$. Let also $\mathcal{T}$ be a finitary, strongly convergent network such that both $\mathcal{M} \between \mathcal{T}$, $\mathcal{N} \between \mathcal{T}$ are defined.

Finally, suppose that $\mathcal{N} \between \mathcal{T}$ has an unsuccessful computation. By Proposition 4.4.9 there are two possible cases

(i) $(\mathcal{N} \between \mathcal{T})\uparrow$; it follows from Proposition 4.4.13 that $(\mathcal{M} \between \mathcal{T})\uparrow$. Again, by Proposition 4.4.9 it follows that $\mathcal{M} \between \mathcal{T}$ has an unsuccessful computation.

(ii) there exists an output deadlock trace t such that $t :: \delta \in \text{DTraces}(\mathcal{N})$. By Assumption, $t :: \delta \in \text{DTraces}(\mathcal{M})$; it follows from Proposition 4.4.9 that $(\mathcal{M} \between \mathcal{T})$ has an unsuccessful computation.

Thus we have proved that if $\neg(\mathcal{N}$ **must-pass** $\mathcal{T})$ it also holds $\neg(\mathcal{M}$ **must-pass** $\mathcal{T})$. By quantifying over all tests we obtain that $\mathcal{N} \sqsupseteq_{\text{must}} \mathcal{M}$.

Therefore we have proved that $\text{DTraces}(\mathcal{N}) \subseteq \text{DTraces}(\mathcal{M})$ implies $\mathcal{N} \sqsupseteq_{\text{must}} \mathcal{N}$; by taking the contrapositive statement, we obtain that $\text{DTraces}(\mathcal{M}) \supseteq \text{DTraces}(\mathcal{N})$ implies $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$. □

4.4.2 Completeness

In this Section we prove that, in a strongly convergent, finitary setting, failure inclusion is complete with respect to the $\sqsupseteq_{\text{must}}$ preorder. To be precise, the statement of the Theorem we prove is provided below.

Theorem 4.4.27 (Completeness for the Must-testing Preorder)**.** Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent, composable networks such that $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$; then $\text{DTraces}(\mathcal{M}) \supseteq \text{DTraces}(\mathcal{N})$.

The proof of Theorem 4.4.27 can be obtained by exhibiting, for every deadlock trace t, a test $\mathcal{T}_t$ such that for any network $\mathcal{M}$ it holds $\neg(\mathcal{M} \text{ must-pass } \mathcal{T}_t)$ if and only if $t \in \text{DTraces}(\mathcal{M})$. Such tests are parametric in both the input and output interface of a network; this does not constitute a problem, for two networks can be compared via the preorder $\sqsubseteq_{\text{must}}$ only in the case that they share the same input and output interface.

Given a network $\mathcal{M}$, the collection of tests $\mathcal{T}_t$ which can be used to test whether a deadlock trace t is included in $\text{DTraces}(\mathcal{M})$ (with respect to the must-pass testing relation) can be obtained by dualising the tests we have provided in Section 4.3.2. Specifically, the dualisation of these tests consist in replacing each deadlocked configuration with a successful one, and vice versa; we will see shortly how this task can be accomplished. Further, we have to define a test $\mathcal{T}_\delta$ that can be used to check whether a network $\mathcal{M}$ can reach a deadlock configuration.

We only present the characteristic tests for any deadlock traces t, together with the statement of the main results that are needed to prove Theorem 4.4.27. The proof of such statements is analogous in style to their corresponding result for the may-testing preorder, stated in Section 4.3.2, and are therefore omitted.

For the moment we assume that the input and output interface of a network $\mathcal{M}$ are fixed and possibly empty; specifically, we assume that $\mathsf{Input}(\mathcal{M}) = \eta_i$ and $\mathsf{Output}(\mathcal{M}) = \eta_o$ for some finite sets of nodes η_i, η_o; for the sake of simplicity, we assume that $\mathsf{Int}(\mathcal{M}) = \{\eta_1, \cdots, \eta_j\}$. For every deadlock trace t, we define the testing network $\mathcal{T}_t = \Gamma_T \triangleright T_t$, where Γ_T is the connectivity graph depicted in Figure 4.6 and $T_t = \prod_{i=1}^{k} e_i[\![P_t^i]\!] \,|\, cn[\![P_t]\!]$; for each node e_i, $i = 1, \cdots, k$, the process P_t^i is defined inductively over t; the same applies for the code P_t running at node cn in $\mathcal{T}_t$.

Let us turn our attention to the inductive definition for such processes. First, for any finite list of variables $x_1, \cdots, x_k$ we define the process $c?(x_1, \cdots, x_k)_\omega.P$ to be exactly P in the case the list $x_1, \cdots, x_k$ is empty, and $c?(x_1, \cdots, x_k)_\omega.P = c?(x_1).(c?(x_2, \cdots, x_k).P) + \tau.\omega$ otherwise. This new construct can be used to define the following process, which makes the definition of the code running at node.

$$P^{\omega}_{\text{check}}.Q = cc!\langle \mathbf{CHECK} \rangle.cc?(x_1, \cdots, x_k)_\omega.Q$$

When providing the definition of the code running at each node of a test $\mathcal{T}_t$, we also make use of the processes $\mathbf{LOCK}, \mathbf{ALLOW}(c, x), P_{\text{next}}.Q, P'_{\text{next}}.Q$ and $P'_{\text{check}}.Q$, defined in Section 4.3.2. We are now ready to define the processes P_i^t, $i = 1, \cdots k$, and P_t. We assume that cc is a fresh channel, not contained in the set $\mathbf{Ch}$.

(i) $t = \varepsilon$

$$\begin{aligned} P^i_\varepsilon &\Leftarrow P'_{\text{check}}.\mathbf{0} \\ P_\varepsilon &\Leftarrow P^{\omega}_{\text{check}}.\mathbf{0} \end{aligned}$$

(ii) $t = \delta$

$$\begin{aligned} P^i_\delta &\Leftarrow \textstyle\sum_{c \in \mathbf{Ch}} c?(x).cc!\langle \mathbf{DETECTED} \rangle \\ P_\delta &\Leftarrow cc?(x)_\omega.\mathbf{0} \end{aligned}$$

(iii) $t = d!v \triangleright \eta :: t'$;

$$P_t^i \Leftarrow \begin{cases} \mathbf{ALLOW}(d,x).\text{if } x \neq v \text{ then } \mathbf{0} \text{ else} \\ (cc!\langle \mathbf{DETECTED} \rangle.(P'_{\text{check}}.P'\text{next}.P_{t'}^i) + \mathbf{LOCK}) & \text{if } e_i \in \eta \\ \\ P'_{\text{check}}.P'_{\text{next}}.P_{t'}^i & \text{if } e_j \notin \eta_i \end{cases}$$

$$P_t \Leftarrow cc_\omega?\big(x_1,\cdots,x_{|\eta|}\big).P^\omega_{\text{check}}.P_{\text{next}}.P_{t'}$$

(iv) $e_j.d?v :: t'$

$$P_t^i \Leftarrow \begin{cases} (d!\langle v \rangle.cc!\langle \mathbf{SENT} \rangle.(P'_{\text{check}}.P'_{\text{next}}.P_{t'}^i + \mathbf{LOCK})) + \mathbf{LOCK} & \text{if } i = j, e_j \in \eta_{\text{in}} \\ \\ P'_{\text{check}}.P'_{\text{next}}.P_{t'}^i & \text{otherwise} \end{cases}$$

$$P_t \Leftarrow cc_\omega?(x).cc!\langle \mathbf{CHECK} \rangle.P^\omega_{\text{check}}.P_{\text{next}}.P_{t'}$$

This definition is similar in style to that of the processes P_i^t, $i = 1,\cdots,k$ and P_t given in Section 4.3.2. However, there are three main differences.

(i) in the process P_ε, run at the controller node cn, the successful state ω is replaced with the deadlocked process $\mathbf{0}$

(ii) whenever the controller node cn is waiting for an input to be received along the channel cc, then it can non-deterministically choose to evolve in a successful state; this ensures that if one of the external nodes deadlock then the coordination protocol is forced to enter in a successful configuration

(iii) a new test for handling the case $t = \delta$ is introduced; intuitively, in this test each node $e_i, i = 1,\cdots,k$ waits to detect a value along an arbitrary channel. In this case, it notifies the controller node, whose code immediately evolves in a non-successful configuration. If the controller node is never notified, then it can only reach a successful state.

In other words, the processes $P_t^i, i = 1,\cdots,k$ and P_t have been defined by dualising those provided in Section 4.3.2. Thus, for the collection of processes $\mathcal{T}_t$ we have the following result.

Proposition 4.4.28. Let $\mathcal{M}$ be a composable, finitary and strongly convergent network such that $\mathsf{Input}(\mathcal{M}) = \eta_i$, $\mathsf{Output}(\mathcal{M}) = \eta_o$.

Then $t \in \text{DTraces}(\mathcal{M})$ if and only $\neg(\mathcal{M}$ must-pass $\mathcal{T}_t)$.

Proof. The proof is analogous to that of Proposition 4.3.26; the assumption that $\mathcal{M}$ is strongly convergent is necessary for ensuring that Proposition 4.4.28 is valid. In fact, if it held $\mathcal{M}\uparrow$, it would not be difficult to show that for any deadlock trace $t \neq \varepsilon$ we would have $t \notin \text{DTraces}(\mathcal{M})$. $(\mathcal{M} \between \mathcal{T}_t)\uparrow$, leading to $\neg(\mathcal{M}$ **must-pass** $\mathcal{T}_t)$. □

Proof of Theorem 4.4.27 Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent networks such that $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$. It follows from Definition 3.3.6 that $\mathsf{Input}(\mathcal{N}) = \mathsf{Input}(\mathcal{M})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{M})$.

Let t be a trace such that $t \in \text{DTraces}(\mathcal{M})$. Then by Proposition 4.4.28 it follows that $\neg\mathcal{M}$ **must-pass** $\mathcal{T}_t$. Since $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$, we also have that $\neg\mathcal{M}$ **must-pass** $\mathcal{T}_t$. A final application of Proposition 4.3.26 leads to $t \in \text{DTraces}(\mathcal{M})$.

We have proved that, for any trace t, if $t \in \text{DTraces}(\mathcal{N})$ then $t \in \text{DTraces}(\mathcal{M})$, under the assumption that $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$; that is, $\text{DTraces}(\mathcal{M}) \supseteq \text{traces}(\mathcal{N})$. □

4.4.3 Towards a Characterisation for Non Strongly Convergent Networks

We have already mentioned in Section 4.4 that our characterisation result for the Must-testing preorder holds only for strongly convergent networks. In fact, we have pointed out in Remark 4.4.3 that, in contrast with standard process calculi, there are cases in which diverging networks can be distinguished by a test via the must-pass testing relation. This is due to the non-blocking nature of broadcast transitions.

In practice, we have shown that a network $\mathcal{M}$ that diverges only after a broadcast transition has been performed can be distinguished from another network $\mathcal{N}$ which diverges via an infinite sequence of τ-extensional transitions. However, for such networks, it holds $\mathcal{N} \sqsubseteq_{\text{must}} \mathcal{N}$. Remark 4.4.3 suggests that the output transitions that a network *must* perform in a divergent computation should be taken into account when providing a characterisation of the must-testing preorder in the general case; in particular, we believe that the following result is true.

Conjecture 4.4.29. Let $\mathcal{M}, \mathcal{N}$ be two finitary networks such that $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$ and $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$

Then

(i) if $\mathcal{M} \Uparrow_t$ for some sequence t of extensional output actions, and $\mathcal{N} \Uparrow_{t'::t'}$, where t' is another sequence of extensional output actions, then $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$

(ii) if $\mathcal{M} \Uparrow_\varepsilon$, then for any network $\mathcal{N}$ it holds $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$.

This conjecture suggests that, to characterise the Must-testing preorder, we need a modified version of deadlock traces which takes into account the sequences of actions that have to be performed by a network before reaching a diverging state. Specifically, we define the set of divergence traces $\text{Div}(\mathcal{M})$ for an arbitrary network $\mathcal{M}$ as follows:

- if $\mathcal{M} \Uparrow_\varepsilon$ then $\uparrow \in \text{Div}(\mathcal{M})$,
- if $\mathcal{M} \Uparrow_t$ then $t ::\uparrow \in \text{Div}(\mathcal{M})$

The enhanced traces of a network $\mathcal{M}$, denoted by $\text{eTraces}(\mathcal{M})$, is defined as $\text{DTraces}(\mathcal{M}) \cup \text{Div}(\mathcal{M})$. This set can be equipped with a partial order $\leq$, which basically reflects the implications of the Conjecture above. Specifically, for any deadlock trace t we let $\uparrow \leq t$, and for any sequence of (possibly empty) extensional actions t we let $t ::\uparrow \leq t' ::\uparrow$, where t' is a suffix of t. We believe that the Smith's preorder induced by the partial order $\leq$ over enhanced traces can be used to characterise the must-testing preorder. Specifically, our Conjecture is stated below.

Conjecture 4.4.30. Let $\mathcal{M}, \mathcal{N}$ be two finitary, composable networks. Then $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$ if and only if

- $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N}), \mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$,
- for any enhanced trace $t' \in \text{eTraces}(\mathcal{N})$ there exists an enhanced trace $t \in \text{eTraces}(\mathcal{M})$ such that $t' \leq t$.

Note that, if two network $\mathcal{M}, \mathcal{N}$ are strongly convergent, then the conjecture above reduces to showing that $\text{DTraces}(\mathcal{M}) \supseteq \text{DTraces}(\mathcal{N})$. We suspect the conjecture above to be true; however, at the current state of the art we are not able to provide a proof of its statement.

4.4.4 Simple Applications of Deadlock Trace Inclusion

In this Section we prove some simple results related to failure trace inclusion and we show some simple application of our proof principle.

For the calculus CCS it is well known [16] that, in a strongly convergent setting, the behavioural equivalences $=_{\text{must}}$ and $\simeq$ coincide. Here we prove that the same result holds for our calculus of networks.

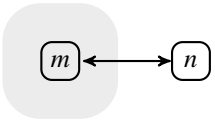


Figure 4.8: A connectivity graph Γ consisting of a fully connected node m and an external node n

Theorem 4.4.31 (Equivalence of $=_{\text{must}}$ and $\simeq$)**.** Let $\mathcal{M}, \mathcal{N}$ be two strongly convergent, proper, finitary networks. Then $\mathcal{M} =_{\text{must}} \mathcal{N}$ iff $\mathcal{M} \simeq \mathcal{N}$.

The proof of Theorem 4.4.31 is rather simple, as it is a simple consequence of the following, rather trivial, statement.

Proposition 4.4.32. Let $\mathcal{M}$ be a finitary, proper, strongly convergent networks. For any trace $t \in \mathsf{EAct}^*$ it holds $t \in \text{traces}(\mathcal{M})$ iff $t \in \text{DTraces}(\mathcal{M})$.

Proof. Note that the hypothesis $t \in \mathsf{EAct}^*$ implies that both the special symbols ω, δ (used in traces and deadlock traces, respectively), do not occur in t.

The proof of the statement is then a trivial consequence of definitions 4.3.1 and 4.4.6, recalling that in a finitary network the predicate $\mathcal{M}' \downarrow$ is satisfied for any network $\mathcal{M}'$. □

Corollary 4.4.33. Let $\mathcal{M}, \mathcal{N}$ be two finitary, strongly convergent, proper networks. If $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$, then $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$.

Proof. Let $t \in \text{traces}(\mathcal{N})$; since $\mathcal{N}$ is a proper network, it holds that $t \in \mathsf{EAct}^*$ (that is, ω does not occur in t). By Proposition 4.4.32 it follows that $t \in \text{DTraces}(\mathcal{N})$. For $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$, Theorem 4.4.11 ensures that $t \in \text{DTraces}(\mathcal{M})$. A final application of Proposition 4.4.32 (recall that $t \in \mathsf{EAct}^*$) leads to $t \in \text{traces}(\mathcal{M})$.

Therefore, for any trace $t \in \text{traces}(\mathcal{N})$ it holds $t \in \text{traces}(\mathcal{M})$; equivalently, $\text{traces}(\mathcal{N}) \subseteq \text{traces}(\mathcal{M})$, hence $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$ as a consequence of Theorem 4.3.8. □

Proof of Theorem 4.4.31 The if implication is trivial. For the only if implication, suppose $\mathcal{M} =_{\text{must}} \mathcal{N}$. By definition it holds that $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$ and $\mathcal{N} \sqsubseteq_{\text{must}} \mathcal{M}$. We can apply Corollary 4.4.33 to the two inequalities above to prove that $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}, \mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, respectively.

Thus we have that, $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$, $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, from which it follows $\mathcal{M} \subseteq \mathcal{N}$. Similarly, we can prove that $\mathcal{N} \subseteq \mathcal{M}$. The last two inequalities imply that $\mathcal{M} \simeq \mathcal{N}$. □

Theorem 4.4.31 will be very useful in Chapter 5. In fact, to check whether two networks are testing equivalent, we will need to check the equivalence between their sets of deadlock traces, without any need to prove that their sets of traces coincide.

As one could expect, the converse of Corollary 4.4.33 does not hold, as the following Example shows.

Example 4.4.34. Let Γ be the connectivity graph of Figure 4.8. Consider the networks $\mathcal{M} = \Gamma \triangleright m[\![c!\langle v\rangle]\!]$ and $\mathcal{N} = \Gamma \triangleright m[\![c?(x).c!\langle v\rangle]\!]$.

It is straightforward to note that $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$. In fact, note that t is a trace included in $\text{traces}(\mathcal{N})$ if and only if it has the form

$$n.d_1?v_1 :: \cdots :: n.d_i?v_i :: n.c?w :: n.c_{i+1}?v_{i+1} :: \cdots :: n.c_{i+j}?v_{i+j} :: \cdots c!v \triangleright \{n\} :: n.c_{i+j+1}?v_{i+j+1} :: \cdots :: n.c_{i+j+k}?v_{i+j+k}$$

where the channels $d_1, \cdots\cdots, d_{i+j+k}$ and the values $v_1, \cdots, v_{i+j+k}, w$ are arbitrary. It is easy to check that such a trace is also included in $\text{traces}(\mathcal{M})$; to this end, note that $\mathcal{M} \overset{n.d?w}{\Longrightarrow} \mathcal{M}$ for any channel $d \in \mathbf{Ch}$ and value $w \in \mathbf{Val}$.

On the other hand, we have that $\mathcal{M} \not\sqsubseteq_{\text{must}} \mathcal{N}$. In fact, it suffices to note that $\mathcal{N}$ is deadlocked, hence $\delta \in \text{DTraces}(\mathcal{N})$. On the other hand, $\mathcal{M}$ can only reach a deadlock configuration only after it has broadcast the

value v along channel c; such a broadcast can be detected by the external node n, hence it does not correspond to an internal activity in the reduction semantics. Thus we have that $\delta \notin \mathrm{DTraces}(\mathcal{M})$.

We have proved that $\mathrm{DTraces}(\mathcal{N}) \not\subseteq \mathrm{DTraces}(\mathcal{M})$; it follows from Theorem 4.4.27 that $\mathcal{M} \not\sqsubseteq_{\mathrm{must}} \mathcal{N}$. We could have proved the statement above also by exhibiting a test $\mathcal{T}$ such that $\mathcal{M}$ **must-pass** $\mathcal{T}$, whereas $\neg(\mathcal{N}$ **must-pass** $\mathcal{T})$. To this end, it suffices to consider the test which places the code $c?(x).\mathbf{0} + \tau.\omega$ at the external node n. □

The last example to which we apply our proof principle takes into account the non-standard definition of extensional actions.

Example 4.4.35. Consider again the networks $\mathcal{M}, \mathcal{N}$ of Figure 4.3. We have already proved in Section 4.3.3 the inequality $\mathcal{N} \sqsubseteq_{\mathrm{may}} \mathcal{M}$. Here we show that the stronger result $\mathcal{M} \sqsubseteq_{\mathrm{must}} \mathcal{N}$ also holds.

The proof of the statement above is rather trivial. It suffices to note that $\mathrm{DTraces}(\mathcal{N}) = \{\varepsilon, c!v \rhd \{o_1, o_2\}, c!v \rhd \{o_1, o_2\}::\delta\}$, while $\mathrm{DTraces}(\mathcal{M}) = \{\varepsilon, c!v \rhd \{o_1\}, c!v \rhd \{o_2\}, c!v \rhd \{o_1, o_2\}, c!v \rhd \{o_1, o_2\}::\delta\}$. Now it is trivial to note that $\mathrm{DTraces}(\mathcal{M}) \supseteq \mathrm{DTraces}(\mathcal{N})$, hence it follows from Theorem 4.4.11 that $\mathcal{M} \sqsubseteq_{\mathrm{must}} \mathcal{N}$. □

Chapter 5

Applications

In this Chapter we analyse case studies in which we apply the theory developed through chapters 2-4.

First we focus on rather simple networks, showing how our characterisation results for the testing preorders can be used to prove that two networks are either may or must testing equivalent. In this case we consider the non-trivial case in which the extensional LTS generated by a network contains recursive states. This topic is covered in Section 5.1.

Then we move to more practical case studies; we focus on desired behaviours of both wired and wireless networks which have been widely addressed in the literature. Given an informal description of the behaviour that we require from a network, or *specification*, we define a formal model for it. A model corresponds to a (usually simple) network for which it is easy, if not trivial, to check that its behaviour is consistent with the specification being considered. Having a model which captures the behaviour we require from a network, we can check whether an arbitrary network is consistent with it by proving that it is testing equivalent to the model we have defined.

We recall that *any verification using model-based techniques is only as good as the model of the system* [3]. In other words, in order to ensure that the behaviour of a network is consistent with its informal specification, we rely on the fact that the model provided to formalise such a specification is not faulty. In general, it is common practice to define a model for a specification to be very simple, so that it is easy to provide an informal argument for stating that the behaviour of the model is consistent with the considered specification.

Sometimes we will work with partially defined networks. That is, we only define a set of formal properties that we assume a network satisfies. Such properties concern both the code that internal nodes of a network are running as well as the connectivity graph of the network. Intuitively, one could see the code run by the internal nodes of a partially defined network as a formal description of a protocol, while the constraints imposed on the network topology can be seen as assumptions that needs to be satisfied to ensure that such a protocol behaves as required.

The first application we consider concerns *connectionless routing* of values between from a node to another in a network. Here the term connectionless means that no assumption is made on the order in which values broadcast by a source are delivered at the receiver. First, we define a model for routing, then we provide an implementation for it; the latter can be seen as an abstraction of a network whose internal nodes are running a *distance vector routing protocol*. This case study is analysed in Section 5.2.

In Section 5.3 we consider a more challenging situation, in which we consider the policy of *connection-oriented routing*. In contrast with connectionless routing, here we require that values are delivered at a destination node in the same order they have been originally broadcast by the source. We show two different implementations of the proposed model; the first one consists of an abstraction of connection-oriented routing protocols at the *internet layer*, while the second one can be seen as an abstraction of connection-oriented protocols running at the *transport layer* of the *TCP/IP standard*.

The case studies considered in sections 5.2 and 5.3 use point-to-point communication between nodes. Thus, the networks we will model in such sections could have been defined using simpler process calculi, such as

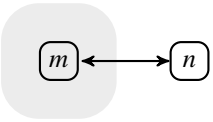


Figure 5.1: A connectivity graph for two recursive networks

CCS; however, one of the advantages that we have by using our framework for networks is that we are able to state explicitly the assumptions that we need to make over network topologies in order to ensure the correct behaviour of a protocol.

Next we move to providing case studies in which protocols use broadcast communication; in Section 5.4 we consider the problem in which a source node wants to multicast a message from a source node to a group of receivers. This problem has been widely studied in the literature, for realizations of multicast protocols are at the basis of data streaming applications [11, 25, 4, 45].

Finally, in Section 5.5 we consider networks that implement a Virtual Shared Memory protocol. These kind of protocols are widely used in distributed systems to simulate a random access memory of a computer in a setting where data can be distributed over different machines [64].

5.1 Relating Networks: Dealing with Recursion

This section contains a simple application of the theory developed in chapters 2- 4. Our aim here is that of showing how our characterisation results of the testing preorders can be used to prove the equivalence of two networks; in particular, we deal with recursive networks, that is networks whose generated extensional LTS have states containing loops.

In this Section we consider the networks $\mathcal{M} = \Gamma \triangleright M$, $\mathcal{N} = \Gamma \triangleright N$, where the connectivity graph Γ is depicted in Figure 5.1 and

$$\begin{aligned} P &\Leftarrow c!\langle v\rangle.(d!\langle w\rangle.P + d'!\langle w'\rangle.P) \\ Q &\Leftarrow (c!\langle v\rangle.d!\langle w\rangle.Q) + (c!\langle v\rangle.d'!\langle w'\rangle.Q) \end{aligned}$$

$$\begin{aligned} M &= m[\![P]\!] \\ N &= m[\![Q]\!] \end{aligned}$$

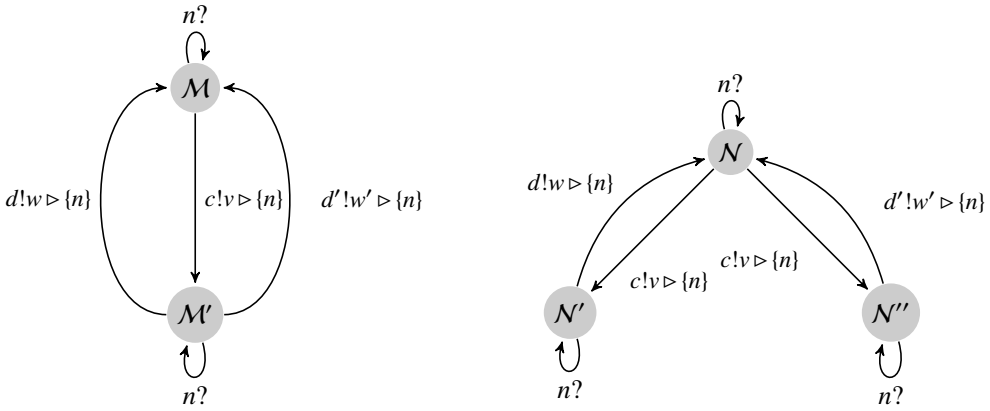
Note that the networks $\mathcal{M}, \mathcal{N}$ are divergent; since our characterisation results for the must-testing preorder are valid only in a strongly convergent setting, for this example we only consider the may-testing preorder and trace inclusion. However, the proof strategy we apply in this Section can also be used when dealing with the must-testing preorder and deadlock traces inclusion.

Our aim is to prove that $\mathcal{M} =_{\text{may}} \mathcal{N}$. To this end, we first give a graphical representation of the extensional LTS generated by these two networks; these are depicted in Figure 5.2. Here we have
$\mathcal{M}' = \Gamma \triangleright m[\![d!\langle w\rangle.P + d'!\langle w'\rangle.P]\!]$, $\mathcal{N}' = \Gamma \triangleright m[\![d!\langle w\rangle.Q]\!]$ and $\mathcal{N}'' = \Gamma \triangleright m[\![d'!\langle w'\rangle.Q]\!]$.

Note that in $\mathcal{M}, \mathcal{N}$ node n is fully connected. That is, in every state of the extensional LTSs generated by such networks the transition $n.c'?v'$ is enabled for any channel c' and value v'. With an abuse of notation, in Figure 5.2 we used the label $n?$ to denote an arbitrary input transition performed by a state.

By looking at the LTSs of Definition 5.2, it is easy to note that an application of Definition 4.3.1 allows us to calculate the set traces($\mathcal{M}$); this is the smallest set such that

- $\varepsilon \in \text{traces}(\mathcal{M})$
- if $t \in \text{traces}(\mathcal{M})$ then $n.c'?v'::t \in \text{traces}(\mathcal{M})$ for any channel c' and value v'.



Extensional LTS generated by $\mathcal{M}$ Extensional LTS generated by $\mathcal{N}$

Figure 5.2: The extensional LTSs generated by $\mathcal{M}$, $\mathcal{N}$

- if $t \in \text{traces}(\mathcal{M}')$ then $c!v \triangleright \{n\} \in \text{traces}(\mathcal{M}')$

The definition of the set $\text{traces}(\mathcal{M})$ depends on that of the set $\text{traces}(\mathcal{M}')$, which is defined as the smallest set such that

- $\varepsilon \in \text{traces}(\mathcal{M}')$
- If $t \in \text{traces}(\mathcal{M}')$ then $n.c'?v'::t \in \text{traces}(\mathcal{M}')$ for any channel c' and value v'
- If $t \in \text{traces}(\mathcal{M})$, then $d!w \triangleright \{n\}::t \in \text{traces}(\mathcal{M}')$
- If $t \in \text{traces}(\mathcal{M})$, then $d'!w' \triangleright \{n\}::t \in \text{traces}(\mathcal{M}')$

Let us turn our attention to the set of traces for the networks $\mathcal{N}, \mathcal{N}', \mathcal{N}''$; these are the smallest sets such that

- $\varepsilon \in \text{traces}(\mathcal{N}), \varepsilon \in \text{traces}(\mathcal{N}'), \varepsilon \in \text{traces}(\mathcal{N}'')$
- for any $\mathcal{L} \in \{\mathcal{N}, \mathcal{N}', \mathcal{N}''\}$, channel c' and value v', if $t \in \text{traces}(\mathcal{N})$ then $n.c'?v'::t \in \text{traces}(\mathcal{L})$
- If $t \in \text{traces}(\mathcal{N}')$ then $c!v \triangleright \{n\}::t \in \text{traces}(\mathcal{N})$
- If $t \in \text{traces}(\mathcal{N}'')$ then $c!v \triangleright \{n\}::t \in \text{traces}(\mathcal{N})$
- If $t \in \text{traces}(\mathcal{N})$ then $d!w \triangleright \{n\}::t \in \text{traces}(\mathcal{N}')$
- If $t \in \text{traces}(\mathcal{N})$ then $d'!w' \triangleright \{n\}::t \in \text{traces}(\mathcal{N}'')$

We are now ready to prove that the two networks are may-testing equivalent.

Proposition 5.1.1. $\mathcal{M} =_{\text{may}} \mathcal{N}$.

Proof. We prove the statements $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ and $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$ separately.

Proving that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ amounts to showing that $\text{traces}(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$. Then the result follows from Theorem 4.3.8. To this end, we prove a more general statement; specifically, we show that

1. If $t \in \text{traces}(\mathcal{M})$, then $t \in \text{traces}(\mathcal{N})$
2. If $t \in \text{traces}(\mathcal{M}')$, then either $t \in \text{traces}(\mathcal{N}')$ or $t \in \text{traces}(\mathcal{N}'')$

The two statements above are proved simultaneously by performing an induction on the trace t.

- The case $t = \varepsilon$ is trivial.

- Suppose $t = n.c'?v'::t'$ for some channel c', value v' and trace t'. If $t \in \text{traces}(\mathcal{M})$, then the only possibility is that $t' \in \text{traces}(\mathcal{M})$, for whenever $\mathcal{M}$ performs an extensional input transition, then the resulting configuration is again $\mathcal{M}$. By inductive hypothesis it holds that $t' \in \text{traces}(\mathcal{N})$; now it is not difficult to show that $n.c'?v'::t' \in \text{traces}(\mathcal{N})$.

  A similar argument can be used to prove that, in this case, if $t \in \text{traces}(\mathcal{M}')$ then either $t \in \text{traces}(\mathcal{N}')$ or $t \in \text{traces}(\mathcal{N}'')$

- If $t = c!v \rhd \{n\}::t'$, then we only have to consider the case $t \in \text{traces}(\mathcal{M})$. This is because the network $\mathcal{M}'$ has no (weak) extensional output transition of value v along channel c. In this case it is not difficult to show that the only possibility is that $t' \in \text{traces}(\mathcal{M}')$. By inductive hypothesis, we have that either $t' \in \text{traces}(\mathcal{N}')$ or $t' \in \text{traces}(\mathcal{N}'')$. If $t' \in \text{traces}(\mathcal{N}')$, it follows by definition that $c!v \rhd \{n\}::t' \in \text{traces}(\mathcal{N})$; a similar argument can be used in the case $t' \in \text{traces}(\mathcal{N}'')$.

- If $t = d!w \rhd \{n\}::t'$, then the only interesting case we have to check is given by $t' \in \text{traces}(\mathcal{M}')$. In this case it is not difficult to show that $t' \in \text{traces}(\mathcal{M})$, hence by inductive hypothesis $t' \in \text{traces}(\mathcal{N})$. Now it is easy to note that $d!w \rhd \{n\}::t' \in \text{traces}(\mathcal{N}')$.

- If $t = d'!w' \rhd \{n\}::t'$ then the only interesting case is that in which $t \in \text{traces}(\mathcal{M}')$. In this case it holds $t' \in \text{traces}(\mathcal{M})$, hence by inductive hypothesis $t' \in \text{traces}(\mathcal{N})$. Now it is not difficult to note that $d'!w' \rhd \{n\}::t' \in \text{traces}(\mathcal{N}'')$.

- The only case which it remains to check is that in which t is a trace whose head is an arbitrary extensional output action different from those already considered. However, this case is vacuous, for such a trace does not belong to $\text{traces}(\mathcal{M})$, nor to $\text{traces}(\mathcal{N})$.

Let us prove now that $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$. In this case it suffices to show that

- If $t \in \text{traces}(\mathcal{N})$ then $t \in \text{traces}(\mathcal{M})$,

- If $t \in \text{traces}(\mathcal{N}')$ then $t \in \text{traces}(\mathcal{M}')$,

- If $t \in \text{traces}(\mathcal{N}'')$ then $t \in \text{traces}(\mathcal{M}')$.

The three statements above are proved simultaneously by induction over a trace t; the proof is virtually identical to the inclusion above. □

Note that, when proving Proposition 5.1.1 we related the traces of the network $\mathcal{M}'$ with those of two different networks, namely $\mathcal{N}', \mathcal{N}''$. More formally, we proved the equivalence $\text{traces}(\mathcal{M}') = \text{traces}(\mathcal{N}') \cup \text{traces}(\mathcal{N}'')$. This is not surprising, for testing preorders lead to a linear theory rather than to a branching one.

5.2 Connectionless Routing

In this Section we consider the task of delivering messages from one source node to a destination one in a network. This problem is well-known in the literature of networks as *routing*.

Routing is the central task that has to be accomplished at the *internet layer* of the *TCP/IP Standard* in both wired and wireless networks. The services that are provided at the network layer in the protocol suite of the TCP/IP standard are well described in [65]: *The internet layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. [..] The internet layer is the lowest layer that deals with end-to-end transmission.* Here the term router corresponds to a node in our framework.

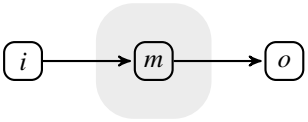


Figure 5.3: The connectivity graph of the model we use to define our routing model

5.2.1 A Faulty Attempt

Given the description above, the behaviour that we require from a network in this Section can be summarised as follows:

Specification 5.2.1 (Connection-less Routing)**.** Given an input node i and an output node o, messages broadcast from the former along a fixed channel c are eventually detected by the latter along the same channel. □

Note that we have made some assumptions in Specification 5.2.1; first, we assumed that the sender and receiver nodes are fixed, as well as the channel used by them to broadcast/detect a message. Second, we have placed no constraints regarding the order in which messages broadcast by the source node i are received at the destination o.

In few words, the last constraints means that the behaviour we wish to model corresponds to *connectionless routing*, which is best described in [65]: *the Internet community argues that the routers' job is moving packets around and nothing else. [..] This viewpoint leads quickly to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else. In particular, no packet ordering and flow control should be done.*

Now that we have given an informal description of the behaviour we wish to model, let us turn our attention at the task of defining a formal model for it. Recall that, when accomplishing this task, we want our model to be as simple as possible. A first attempt to define a model for connectionless routing could be that of considering the network $\mathcal{M}_0 = \Gamma_M \rhd M$, where Γ_M is the connectivity graph of Figure 5.3, M is the system term $m[\![P]\!]$ and P is the process definition $P \Leftarrow c?(x).c!\langle x\rangle.P$.

However, it is rather easy to show that this model is not consistent with Specification 5.2.1. To this end, note that the network $\mathcal{M}_0$, upon receiving a message v along channel c broadcast by node i, evolves into the network $\mathcal{M}_0^v = \Gamma_M \rhd m[\![c!\langle v\rangle.P]\!]$. Such a network is not waiting to receive any value anymore; if a second value w is broadcast by node i, node m in network $\mathcal{M}_0^v$ will ignore it and therefore it will never be able to forward it to node m; this behaviour for the network $\mathcal{M}_0$ is obviously inconsistent with the informal specification we are considering.

A second attempt to define a model for Specification 5.2.1 requires modifying the network $\mathcal{M}_0$ so that, at any given time, the internal node m can either receive a value from the input node i or forward one of the values it has previously detected to the external node o. In order for this to be possible, node m has to be able to keep store all the values broadcast by node i which have not yet been forwarded to node o in an internal buffer.

This task can be accomplished by using *multisets*; a multiset is a set which can contain multiple occurrences of an element [5]. In the following we use the symbols $\mathscr{M}, \mathscr{N}, \cdots$ to range over multisets. Further, we use the notation $\{\!|v_1, v_2, \cdots|\!\}$ to denote the multiset containing the (not necessarily distinct) elements $v_1, v_2, \cdots$.

The number of occurrences of an element v in a multiset $\mathscr{M}$ is called the *multiplicity* of v in $\mathscr{M}$ and we denote it as $\mathsf{mult}(v, \mathscr{M})$. We use the notation $\emptyset$ for the empty multi set, that is the multiset such that for any element v it holds $\mathsf{mult}(v, \emptyset) = 0$. If $\mathsf{mult}(v, \mathscr{M}) > 0$ for some element v and multiset $\mathscr{M}$, we say that $v \in \mathscr{M}$, while if $\mathsf{mult}(v, \mathscr{M}) = \mathsf{mult}(v, \mathscr{N})$ for any value v and two given multisets $\mathscr{M}, \mathscr{N}$, we say that $\mathscr{M} = \mathscr{N}$.

The union of two multisets $\mathscr{M}, \mathscr{N}$ is the multiset $\mathscr{M} \cup \mathscr{N}$ such that for any value v $\mathsf{mult}(v, (\mathscr{M} \cup \mathscr{N})) = \mathsf{mult}(v, \mathscr{M}) + \mathsf{mult}(v, \mathscr{N})$. The difference of two multisets $\mathscr{M}, \mathscr{N}$ is the multiset $\mathscr{M} \setminus \mathscr{N}$ such that, for any element v, $\mathsf{mult}(v, (\mathscr{M} \setminus \mathscr{N})) = \min\{0, (\mathsf{mult}(v, \mathscr{M}) - \mathsf{mult}(v, \mathscr{N}))\}$.

Intuitively, we can use multisets of values to let a node m in a network whose connectivity graph is the same as in Figure 5.3 keep track of the messages it has received from the input node i, but it has not forwarded to the

output node o; more specifically, multisets can be used to associate a (unsorted) memory buffer to the node m. Note that we needed to use multisets for there is the possibilities that more copies of a given value have to be stored in the buffer associated with the node m.

In fact, given a finite set of values **Val** we can assume, for any finite multiset of values $\mathscr{M}$, the process definition

$$P_{\mathscr{M}} \Leftarrow (c?(x).P_{(\mathscr{M}\cup\{x\})}) + \left(\sum_{v\in\mathscr{M}} c!\langle v\rangle.P_{(\mathscr{M}\setminus\{v\})}\right)$$

Then we could consider the network $\mathcal{M}' = \Gamma_M \rhd m[\![P_\emptyset]\!]$ as a possible model for Specification 5.2.1. The behaviour of such a network is described informally below.

- Initially, the buffer of node m is empty; that is, there is no value that has to be forwarded to the output node o. The only activity that node m can perform consists in receiving a message broadcast by the input node i along channel c. Upon receiving a message v, the code of node m evolves in $P_{\{v\}}$, meaning that the value v has been stored in the internal buffer of node m.

- At any given time, node m can either receive a value w broadcast by node i, or it can decide to broadcast one of the values (if any) w' which are stored in its buffer. In the first case, the value w is stored in the buffer of node m, while in the latter the value w' is removed from it.

Despite node m never ignores values broadcast by the input node i in network $\mathcal{M}'$, its behaviour is still inconsistent with Specification 5.2.1. In fact, it is easy to note that the code of node m suffers from *starvation*, meaning that there exists at least a sequence of transitions in which a value broadcast by node i is never delivered to node o. In fact, the network $\mathcal{M}' = \Gamma_M \rhd m[\![P_\emptyset]\!]$ is equipped with the sequence of extensional transitions

$$(\Gamma_M \rhd m[\![P_\emptyset]\!]) \xmapsto{i.c?v_0} (\Gamma_M \rhd m[\![P_{\{v_0\}}]\!]) \xmapsto{i.c?v_1} (\Gamma_M \rhd m[\![P_{\{v_0,v_1\}}]\!]) \xmapsto{i.c?v_2} \cdots$$

Further, it is straightforward to check that the extensional LTS induced by the network $\mathcal{M}'$ has an infinite number of states; since the proof methods we have proved in Chapter 4 can only be used with finitary networks, they cannot be applied to the network $\mathcal{M}'$.

5.2.2 A Second Attempt

In is very unlikely that we can find a finitary network $\mathcal{M}$ whose behaviour is consistent with Specification 5.2.1. In fact we are explicitly assuming that node i is external, hence there is no way to control the rate at which it broadcasts values to node m; for example, we cannot impose the constraint that node i always waits for node m to forward a value before broadcasting another one.

In contrast, we can slightly modify our Specification.

Specification 5.2.2 (Finite Connection-less Routing)**.** Given an input node i and an output node o and a positive integer k, the first k messages broadcast from node i along a fixed channel c are eventually detected by node o along the same channel. □

Specification 5.2.2 explicitly states that we only consider the first k values broadcast by the input node i, while subsequent values can be ignored. Note that this specification is parametric in the number k of values which have to be routed from node i to node o.

For any positive integer k, it is easy to define a network whose behaviour is consistent with Specification 5.2.2. In fact, it is sufficient to equip node m with a counter which keeps track of the number of values it still needs to receive from the input node i.

To this end, we assume the following process definitions, parametric in a multiset $\mathscr{M}$ and any integer $l \geq 0$:

$$P^0_{\mathscr{M}} \Leftarrow \sum_{v \in \mathscr{M}} c!\langle v \rangle . P^0_{(\mathscr{M} \setminus \{\!|v|\!\})}$$

$$P^{l+1}_{\mathscr{M}} \Leftarrow (c?(x) . P^l_{\mathscr{M} \cup \{\!|v|\!\}}) + \left(\sum_{v \in \mathscr{M}} c!\langle v \rangle . P^{l+1}_{(\mathscr{M} \setminus \{\!|v|\!\})} \right)$$

and let $\mathcal{M}_k = m[\![P^k_\emptyset]\!]$. Intuitively, network $\mathcal{M}_k$ starts ignoring inputs fired from node i after it has received exactly k values, and it deadlocks after such values have been forwarded to the output node o. This network constitutes a model for Specification 5.2.2[1].

Let us turn our attention to the formal behaviour of the network $\mathcal{M}_k$. For the sake of simplicity, for any integer $l \geq 0$ and finite multiset of values $\mathscr{M}$, we let $\mathcal{M}^l_{\mathscr{M}}$ be the network $\Gamma_M \triangleright m[\![P^l_{\mathscr{M}}]\!]$. Therefore, $\mathcal{M}_k = \mathcal{M}^k_\emptyset$.

Proposition 5.2.3. Let $j \geq 0$, $\mathscr{M}$ be a finite multiset of values and $v \in \mathbf{Val}$ be a value.

- For any channel $d \in \mathbf{Ch}$ (possibly equal to c) $\mathcal{M}^0_{\mathscr{M}} \xmapsto{i.d?v} \mathcal{M}'$ if and only if $\mathcal{M}' \equiv \mathcal{M}^0_{\mathscr{M}}$,
- $\mathcal{M}^{l+1}_{\mathscr{M}} \xmapsto{i.c?v} \mathcal{M}'$ if and only if $\mathcal{M}' \equiv \mathcal{M}^l_{(\mathscr{M} \cup \{\!|v|\!\})}$.
- Let $d \in \mathbf{Ch}, d \neq c$; then $\mathcal{M}^{l+1}_{\mathscr{M}} \xmapsto{i.d?v} \mathcal{M}'$ if and only if $\mathcal{M}' \equiv \mathcal{M}^{l+1}_{\mathscr{M}}$.
- $\mathcal{M}^l_{\mathscr{M}} \xmapsto{c!v \triangleright \{o\}} \mathcal{M}'$ if and only if $v \in \mathscr{M}$ and $\mathcal{M}' \equiv \mathcal{M}^l_{(\mathscr{M} \setminus \{\!|v|\!\})}$.
- $\mathcal{M}^l_{\mathscr{M}}$ is deadlocked if and only if $\mathscr{M} = \emptyset$.

Proof. The proof of this statement is straightforward. □

We can use Proposition 5.2.3 to calculate the set of deadlock traces for the network $\mathcal{M}_k$ and, more generally, the set $\mathrm{DTraces}(\mathcal{M}^l_{\mathscr{M}})$ for any integer $l \geq 0$ and finite multiset of values $\mathscr{M}$.

Corollary 5.2.4. For any $l \geq 0$, finite multiset of values $\mathscr{M}$ the set $\mathrm{DTraces}(\mathcal{M}^l_{\mathscr{M}})$ is the least set such that

- $\varepsilon \in \mathrm{DTraces}(\mathcal{M}^l_{\mathscr{M}})$
- If $\mathscr{M} = \emptyset$ then $\delta \in \mathcal{M}^l_{\mathscr{M}}$
- For any channel $d \in ch$ and value $v \in \mathbf{Val}$, if $t \in \mathcal{M}^0_{\mathscr{M}}$ then $i.d?v::t \in \mathcal{M}^0_{\mathscr{M}}$
- For any value $v \in \mathbf{Val}$, if $v \in \mathscr{M}$, and $t \in \mathrm{DTraces}(\mathcal{M}^l_{(\mathscr{M} \setminus \{\!|v|\!\})})$ then $c!v \triangleright \{o\}::t \in \mathrm{DTraces}(\mathcal{M}^l_{\mathscr{M}})$
- For any channel $d \in \mathbf{Ch}, d \neq c$ and value $v \in \mathbf{Val}$, if $t \in \mathrm{DTraces}(\mathcal{M}^{l+1}_{\mathscr{M}})$ then $i.d?v::t \in \mathrm{DTraces}(\mathcal{M}^l_{\mathscr{M}})$
- For any value $v \in \mathbf{Val}$, if $t \in \mathrm{DTraces}(\mathcal{M}^l_{(\mathscr{M} \cup \{\!|v|\!\})})$ implies $i.c?v::t \in \mathrm{DTraces}(\mathcal{M}^{l+1}_{\mathscr{M}})$

Proof. Trivial, by using Proposition 5.2.3 and Definition 4.4.7. □

5.2.3 The Implementation

So far we have described an informal specification for connection-less routing of a finite number k of messages and exhibited a model, a network $\mathcal{M}_k$, whose behaviour is consistent with such a specification. Our aim in this Section is to exhibit an implementation of the network $\mathcal{M}_k$, that is a finitary network $\mathcal{N}_k$ which is testing equivalent to our model; in order to show the equivalence $\mathcal{M}_k \simeq \mathcal{N}_k$, it is sufficient to show that $\mathrm{DTraces}(\mathcal{M}_k) = \mathrm{DTraces}(\mathcal{N}_k)$, as stated in Theorem 4.4.31.

In practice, we work with partially defined networks; that is, given an integer $k > 0$, we state the properties that we require from a network $\mathcal{N}_k$ to satisfy in order to ensure that $\mathrm{DTraces}(\mathcal{M}_k) = \mathrm{DTraces}(\mathcal{N}_k)$. As a consequence, the equivalence $\mathcal{M}_k \simeq \mathcal{N}_k$ will hold for any network $\mathcal{N}_k$ which satisfies such properties.

[1] To be very pedantic the network $\mathcal{M}_k$ still suffers from starvation; however, in this case starvation is caused by input transitions which do not affect the internal node m of $\mathcal{M}_k$, therefore they can be ignored

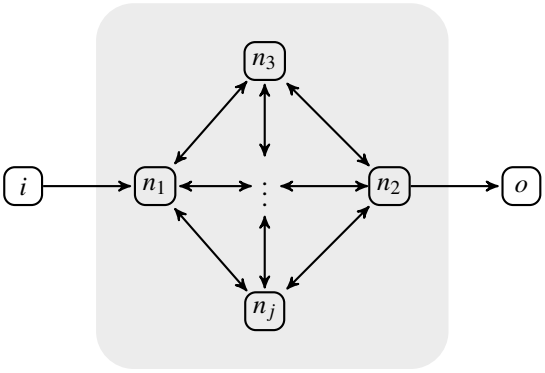


Figure 5.4: The parametrised connectivity graph Γ_N used in the Implementation of connectionless routing

Fixed an integer $k \geq 0$, let us discuss the properties that we require from an implementation $\mathcal{N}_k$ of our routing model; we assume that this network has the form $\Gamma_N \triangleright N_k$, where

- $\mathsf{nodes}(N_k) = \{n_1, \cdots, n_j\}$ for some index $j \geq 2$
- $\mathsf{Input}(\mathcal{N}_k) = \{i\}, \mathsf{Output}(\mathcal{N}_k) = \{o\}$; further, we assume that whenever $\Gamma_N \vdash i \rightarrow n_h$ for the only index $h = 1$. Similarly, if $\Gamma_N \vdash n_h \rightarrow o$ for the only index $h = 2$
- $\Gamma_N \vdash n_1 \twoheadrightarrow n_2$
- For any node in $n \in \mathsf{nodes}(N_k)$ there exists a directed path from the internal node n to the output node o; further, for such nodes we let $\mathsf{length}(n, \Gamma_N)$ be the length of the minimal path between the node n and the output node o and $\mathsf{next}(h, \Gamma_N) = \{h' \mid \Gamma_N \vdash n_h \rightarrow n_{h'} \text{ and } \mathsf{length}(n_{h'}, \Gamma_N) = \mathsf{length}(n_h, \Gamma_N) - 1\}$
- for any node $h = 2, \cdots, j$ it holds $\mathsf{length}(n_h, \Gamma_N) \leq \mathsf{length}(n_1, \Gamma_N)$,
- For any index $h = 2, \cdots, j$, we assume the existence of a channel $c_h \in \mathbf{Ch}$, different from the channel c used in the routing model $\mathcal{M}_k$
- Let h, l be two indexes ranging over $3, \cdots, j$ and $0, \cdots, k$,respectively; further, let $\mathscr{M}$ be a finite multiset of values. We make use of the following process definitions

$$
\begin{array}{rcl}
Q^2_{\mathscr{M}} & \Leftarrow & c_2?(x).Q^2_{(\mathscr{M} \cup \{x\})} + \left(\sum_{v \in \mathscr{M}} c!\langle v \rangle .Q^2_{(\mathscr{M} \setminus \{x\})} \right) \\
Q^h_{\mathscr{M}} & \Leftarrow & (c_h?(x).Q^h_{(\mathscr{M} \cup \{x\})}) + \left[\sum_{h' \in \in \mathsf{next}(h, \Gamma_N)} \left(\sum_{v \in \mathscr{M}} c_{h'}!\langle v \rangle .Q^h_{(\mathscr{M} \setminus \{v\})} \right) \right] \\
R^0_{\mathscr{M}} & \Leftarrow & \left[\sum_{h \in \mathsf{next}(1, \Gamma_N)} \left(\sum_{v \in \mathscr{M}} c_h!\langle v \rangle .R^0_{(\mathscr{M} \setminus \{v\})} \right) \right] \\
R^{l+1}_{\mathscr{M}} & \Leftarrow & (c?(x).R^l_{(\mathscr{M} \cup \{x\})}) + \left[\sum_{h \in \mathsf{next}(1, \Gamma_N)} \left(\sum_{v \in \mathscr{M}} c_h!\langle v \rangle .R^{l+1}_{(\mathscr{M} \setminus \{v\})} \right) \right]
\end{array}
$$

to define the system term N_k as

$$N_k = n_1[\![R^k_\emptyset]\!] \mid \prod_{h=2}^{j} n_h[\![Q^h_\emptyset]\!]$$

An informal description of the behaviour of the (parametric) network $\mathcal{N}_k$ is mandatory; its (partially defined) connectivity graph Γ_N is depicted in Figure 5.4.

Note that we assumed a channel c_h for any internal node n_h, where $h = 2, \cdots, j$. By looking at the code of the system term N_k, it is easy to note that the node n_h is the only one that can detect values broadcast along the

associated channel c_h. That is, if a node in the network $\mathcal{N}_k$ broadcasts a value along a channel c_h, then such a value is intended to be received only by the node n_h; further, at any given time the node n_h is listening for values incoming at channel c_h, hence it is ensured that it will receive values broadcast along such a channel by any node which is directly connected to it.

For any index $h = 3, \cdots, j$, if the node n_h detects a value, it will store it in its local buffer; this is represented as a multiset $\mathscr{M}$. At any given time, node n_h can decide to non-deterministically select a value in its buffer (if any) and an index h' such that the node n'_h can detect the broadcasts fired by n_h (that is, $\Gamma_N \vdash n_h \to n_{h'}$) and which is less distant to the output node o (that is, the minimal length of a path from node $n_{h'}$ to the output node o is strictly lower than the minimal length of a path from n_h to o); once the index h' and the value v have been selected, node n_h broadcasts value v along channel $c_{h'}$. Since the node $n_{h'}$ is the only one that can receive values broadcast along channel $c_{h'}$, it is ensured that it will be the only node that will detect such a broadcast. In other words, at any given time node n_h can either

- receive a message and store it in its buffer, or
- select a message stored in its own buffer, and forward it to one of its neighbouring node which is less distant to the output node o. Note that such a neighbouring node exists, for we are assuming that any internal node in $\mathcal{N}_k$ has a directed path from it to node o (therefore the minimal length of a path from such a node to o is defined and finite).

It remains to describe the informal behaviour of the nodes n_1, n_2. This is similar to that of the other nodes n_h, $h = 3, \cdots, j$. However, there are some crucial differences:

- node n_1 can only receive messages broadcast along channel c; since the only internal node that can broadcast values along such a channel is n_2, and we are explicitly assuming that n_1 is not in the range of transmission of n_2, we are ensured that whenever n_1 receives a message along channel c then it is because of a broadcast fired by the input node i,
- node n_1 can only receive k messages along channel c; this is unsurprising, for we have placed the same constraint in the model $\mathcal{M}_k$,
- node n_2 can only broadcast values along channel c; the only internal node that can receive messages broadcast along channel c is n_1; however, we are assuming that $\Gamma_N \vdash n_2 \nrightarrow n_1$. Therefore, the only node that can detect values broadcast by node n_2 is the output node o.

At this point the reader should have a clear idea of the behaviour of the network $\mathcal{N}_k$. Next, we examine the strong extensional transitions that can be performed by the states of the extensional LTS generated by $\mathcal{N}_k$.

To this end, let buf be a function that maps elements in $\{1, \cdots, j\}$ to finite multisets of values, and l be an index ranging over $0, \cdots, k$. We define the network $\mathcal{N}_l^{\text{buf}}$ as

$$\mathcal{N}_l^{\text{buf}} = \Gamma_N \rhd n_1[\![R^l_{\text{buf}(1)}]\!] \mid \prod_{h=2}^{j} n_h[\![Q^h_{\text{buf}(h)}]\!]$$

Note that we have $\mathcal{N}_k = \mathcal{N}^k_{\text{buf}_0}$, where buf_0 is the function such that $\text{buf}_0(h) = \emptyset$ for any $h = 1, \cdots, j$. In the following we use the standard notation $\text{buf}[h \mapsto \mathscr{M}]$ to denote the function that associates the multiset $\mathscr{M}$ to the index h and the value $\text{buf}(h')$ to any other index $h' \neq i$.

We will also need some non-standard notation. The function $(\text{buf} \downarrow v)$ is defined as $\text{buf}[1 \mapsto (\text{buf}(1) \cup \{\!|v|\!\})]$; intuitively the function $(\text{buf} \downarrow v)$ described how the contents of the local buffers of nodes in the implementation $\mathcal{N}_k$, represented by buf, evolve when a value v is received at the input node n_1 and stored in its local buffer. The function $\text{buf}[h \Uparrow v] = \text{buf}[h \mapsto (\text{buf}(h) \setminus \{\!|v|\!\})]$ denotes how the local buffers of internal nodes in $\mathcal{N}_k$ evolve when value v flows from node n_h to the external node o; finally, $\text{buf}[h \overset{v}{\rightsquigarrow} h'] = (\text{buf}[h \mapsto (\text{buf}(h) \setminus \{\!|v|\!\})])[h' \mapsto (\text{buf}(h') \cup \{\!|v|\!\})]$ is used to describe a value flowing from the internal node n_h to node $n_{h'}$.

Proposition 5.2.5. Let buf be a mapping from the index set $\{1,\cdots,j\}$ to finite multisets of values, and let $l \geq 0$.

1. For any value $v \in \mathbf{Val}$ and channel $d \in \mathbf{Ch}$ (possibly equal to c, $\mathcal{N}^0_{\text{buf}} \overset{i.d?v}{\longmapsto} \mathcal{N}'$ if and only if $\mathcal{N}' \equiv \mathcal{N}^0_{\text{buf}}$
2. For any value $v \in \mathbf{Val}$ and channel $d \in \mathbf{Ch}$, $d \neq c$, $\mathcal{N}^{l+1}_{\text{buf}} \overset{i.d?v}{\longmapsto} \mathcal{N}'$ if and only if $\mathcal{N}' \equiv \mathcal{N}^l_{\text{buf}}$
3. For any value $v \in \mathbf{Val}$ $\mathcal{N}^{l+1} \overset{i.c?v}{\longmapsto} \mathcal{N}'$ if and only if $\mathcal{N}' \equiv \mathcal{N}^l_{(\text{buf}\downarrow v)}$
4. $\mathcal{N}^l_{\text{buf}} \overset{\tau}{\longmapsto} \mathcal{N}'$ if and only if there exists an index $h \neq 2$ such that $v \in f(h)$ for some value v and $\mathcal{N}' \equiv \mathcal{N}^l_{\text{buf}[h \overset{v}{\rightsquigarrow} h']}$ for some index $h' \in \text{next}(h,\Gamma_N)$
5. $\mathcal{N}^l_{\text{buf}} \overset{c!v \triangleright \{o\}}{\longmapsto} \mathcal{N}'$ if and only if $v \in f(2)$ and $\mathcal{N}' \equiv \mathcal{N}^l_{\text{buf}[2 \Uparrow v]}$
6. $\mathcal{N}^l_{\text{buf}}$ is deadlocked if and only if $\bigcup_{h=1}^{j} \text{buf}(h) = \emptyset$

Proof. We only prove Statement 4. For the if implication, let h be an index in $\{1,\cdots,j\}$ such that $h \neq 2$, and let $v \in \mathbf{Val}$ be a value such that $v \in \text{buf}(h)$. Finally, let h' be an index such that $h' \in \text{next}(h,\Gamma_N)$. Such an index exists since we are assuming that every node in the network $\mathcal{N}_k$ is connected to the output node o.

Without loss of generality, assume that $h \neq 1$ (the case $h = 1$ is similar), $h < h'$. By Definition we have that

$$\begin{aligned}
\mathcal{N}^l_{\text{buf}} &= \Gamma_N \triangleright n_1[\![R^l_{\text{buf}(1)}]\!] \,|\, n_2[\![Q^2_{\text{buf}(2)}]\!] \,|\cdots n_h[\![Q^h_{\text{buf}(h)}]\!] \,|\cdots|\, n_{h'}[\![Q^{h'}_{f(h')}]\!] \,|\cdots n_2[\![Q^2_{f(2)}]\!] \\
Q^h_{f(h)} &\Leftarrow (c_h?(x).Q^h_{(\text{buf}(h)\cup\{x\})}) + \left[\sum_{h' \in \text{next}(h,\Gamma_N)} \left(\sum_{v \in \text{buf}(h)} c_{h'}!\langle v\rangle.Q^h_{(\text{buf}(h)\setminus\{v\})} \right)\right] \\
Q^{h'}_{\text{buf}(h')} &\Leftarrow (c_{h'}?(x).Q^{h'}_{(\text{buf}(h'')\cup\{x\})}) + \left[\sum_{h'' \in \text{next}(h',\Gamma_N)} \left(\sum_{v \in \text{buf}(h')} c_{h''}!\langle v\rangle.Q^{h'}_{(\text{buf}(h')\setminus\{v\})} \right)\right]
\end{aligned}$$

from which it is not difficult to infer the following transitions in the intensional semantics:

$$\begin{aligned}
\Gamma_N \triangleright n_1[\![R^l_{\text{buf}(1)}]\!] &\overset{n_h.c_{h'}?v}{\longrightarrow} n_1[\![R^l_{\text{buf}(1)}]\!] \\
\Gamma_N \triangleright n_{h''}[\![Q^{h''}_{\text{buf}(h'')}]\!] &\overset{n_h.c_{h'}?v}{\longrightarrow} n_{h''}[\![Q^{h''}_{\text{buf}(h'')}]\!] \\
\Gamma_N \triangleright n_{h'}[\![Q^{h'}_{\text{buf}(h')}]\!] &\overset{n_h.c_{h'}?v}{\longrightarrow} n_{h'}[\![Q^{h'}_{(\text{buf}(h')\cup\{v\})}]\!] \\
\Gamma_N \triangleright n_h[\![Q^h_{\text{buf}(h)}]\!] &\overset{n_h.c_{h'}!v}{\longrightarrow} n_h[\![Q^h_{(\text{buf}(h)\setminus\{v\})}]\!]
\end{aligned}$$

here h'' is an index different from $1,h,h'$. Note that the first transition can be derived because $h' \neq 1$; in fact, recall that $h' \in \text{next}(h,\Gamma_N)$; if if were $h' = 1$, we would have $\text{length}(n_h,\Gamma_N) > \text{length}(n_1,\Gamma_N)$, contradicting the requirements that we placed over the connectivity graph Γ_N. The third transition can also be derived because we are assuming that $h' \in \text{next}(h,\Gamma_N)$, from which it follows $\Gamma_N \vdash n_h \to n_{h'}$. By a repeated application of rules $(\text{B-SYNC}-\text{L}), (\text{B-SYNC}-\text{R}), (\text{B-PROP})$ (see Figure 2.4) it is not difficult to obtain the transition

$$\mathcal{N}^l_{\text{buf}} \overset{n_h.c_{h'}!v}{\longrightarrow} \mathcal{N}'$$

where $\mathcal{N}'$ is the network

$$\Gamma_N \triangleright n_1[\![R^l_{\text{buf}(1)}]\!] \,|\, n_2[\![Q^2_{f(2)}]\!] \,|\cdots n_h[\![Q^h_{(\text{buf}(h)\setminus\{v\})}]\!] \,|\cdots|\, n_{h'}[\![Q^{h'}_{(\text{buf}(h')\cup\{v\})}]\!] \,|\cdots n_2[\![Q^2_{\text{buf}(2)}]\!]$$

which is exactly $\mathcal{N}^l_{\text{buf}[h \overset{v}{\rightsquigarrow} h']}$.

It remains to show that the transition above induces a τ-extensional transition; this is immediate, for $\mathsf{Output}(\mathcal{N}^l_{\text{buf}}) = \{o\}$ and if $\Gamma_N \vdash n_h \to o$ then $h = 2$. Since we are assuming $h \neq 2$, it follows that $\{n \in \mathsf{Output}(\mathcal{N}^l_{\text{buf}}) \mid \Gamma_N \vdash n_h \to n\} = \emptyset$. It follows from Definition 4.1.3 that $\mathcal{N}^l_f \overset{\tau}{\longmapsto} \mathcal{N}^l_{\text{buf}[h \overset{v}{\rightsquigarrow} h']}$.

Now suppose that $\mathcal{N}^l_{\text{buf}} \overset{\tau}{\longmapsto} \mathcal{N}'$ for some network $\mathcal{N}'$; it is immediate to note that such an extensional transition corresponds to a broadcast performed by one of the nodes $n_1, \cdots, n_j$, for none of such nodes can perform a (intensional) τ-transition in $\mathcal{N}^l_{\text{buf}}$. More specifically, we have that $\mathcal{N}^l_{\text{buf}} \overset{n_h.c_{h'}!v}{\longrightarrow} \mathcal{N}'$ for some indexes h, h' ranging over $1, \cdots, j$ and value v.

By Proposition 2.4.7 it holds that $\mathcal{N}^l_{\text{buf}} \equiv n_h[\![c_{h'}!\langle e\rangle.P + Q]\!] \,|\, N$ for some processes P, Q, system term N and closed expression e such that $[\![e]\!] = v$. Further, $\mathcal{N}' \equiv n_h[\![P]\!] \,|\, N'$; here N' is a system term such that $\Gamma_N \rhd N \overset{n_h.c_{h'}?v}{\longrightarrow} N'$. At this point we have two possibilities:

- $h = 1$; in this case we have that $c_h!\langle e\rangle.P + Q \equiv R^l_{f(1)}$, from which it follows that $v \in \text{buf}(h)$ and the index h' is such that $h' \in \text{next}(h, \Gamma_N)$, or
- $h > 2$[2]; in this case it follows that $c_h!\langle e\rangle.P + Q \equiv Q^h_{\text{buf}(h)}$, hence $v \in f(h)$ and $h' \in \text{next}(h, \Gamma_N)$.

Let us consider the first case; the second one is analogous. In this case we have that $N \equiv \prod_{h''=2}^{j} n_{h''}[\![Q^{h''}_{\text{buf}(h'')}]\!]$. Since $h' \in \text{next}(1, \Gamma_N)$, it follows that $\Gamma_N \vdash n_1 \to n_{h'}$; further, for any other index $h'' \neq 1, h'$, it is easy to note that $\text{rcv}(Q^{h''}_{\text{buf}(h'')}, c_{h'})$ is not true. It follows from Proposition 2.4.9 that

$$N' \equiv Q^{h'}_{\text{buf}(h')\cup\{v\}}[\![\mathbb{I}]\!] \,|\, \prod_{\substack{h''=2 \\ h''\neq h'}}^{j} n_{h''}[\![Q^{h''}_{\text{buf}(h'')}]\!]$$

Thus we have proved that the network $\mathcal{N}'$ is structurally equivalent to

$$\Gamma_N \rhd n_1[\![R^l_{\text{buf}(1)\setminus\{v\}}]\!] \,|\, n_{h'}[\![Q^{h'}_{\text{buf}(h')\cup\{v\}}]\!] \,|\, \prod_{\substack{h''=1 \\ h''\neq h'}}^{j} n_{h''}[\![Q^{h''}_{\text{buf}(h'')}]\!]$$

which is exactly the network $\mathcal{N}^l_{\text{buf}[1 \overset{v}{\rightsquigarrow} h']}$. □

It is immediate to observe from Proposition 5.2.5 that every state of the extensional LTS generated by the network $\mathcal{N}_k$ corresponds to a network of the form $\mathcal{N}^l_{\text{buf}}$, where $l \leq k$. Further, if we define the cardinality of a multiset $\mathscr{M}$ as $|\mathscr{M}| = \sum_{v\in\mathscr{M}} \text{mult}(v, \mathscr{M})$, it is also easy to note that for such networks it holds $\sum_{h=1}^{j} |\text{buf}(h)| \leq k$. In other words, in any state of the LTS generated by the network $\mathcal{N}_k$, there are no more than k values to be collected by the node n_1, and no more than k to be delivered at the output node o. At this point it is easy to note that network $\mathcal{N}_k$ is both finitary and strongly convergent.

In Proposition 5.2.5 we have shown that there is the possibility for the network $\mathcal{N}_k$ to reach a configuration which can perform τ-extensional transitions. Since our aim is to calculate the set of deadlock traces of $\mathcal{N}_k$, it is more practical to reason on weak extensional transitions.

To this end, we define the relation $\prec$ between mappings from the index set $\{1, \cdots, j\}$ and finite multisets of values by letting $\text{buf} \prec \text{buf}'$ whenever $\text{buf}' = \text{buf}[h \overset{v}{\rightsquigarrow} h']$ for two indexes h, h' such that $h' \in \text{next}(h, \Gamma_N)$ and a value $v \in \text{buf}(h)$.

We let $\preccurlyeq$ be the transitive, reflexive closure of $\prec$. Consider two networks $\mathcal{N}^l_{\text{buf}}$ and $\mathcal{N}^l_{\text{buf}'}$ such that $\text{buf} \preccurlyeq \text{buf}'$; intuitively, the last constraint says that the network $\mathcal{N}^l_{\text{buf}'}$ can be obtained from the network $\mathcal{N}^l_{\text{buf}}$ by letting the values stored at some internal node flow until they reach a location which is closer to the destination node o[3]. Further, the multisets of values stored in the overall networks $\mathcal{N}^l_{\text{buf}}$ and $\mathcal{N}^l_{\text{buf}'}$ are the same. Formally, we have the following statements:

Lemma 5.2.6. If buf, buf' are two mappings such that $\text{buf} \preccurlyeq \text{buf}'$, then $\bigcup_{h=1}^{j} \text{buf}(h) = \bigcup_{h=1}^{j} \text{buf}'(h)$.

Proof. It is easy to note that the statement holds for the relation $\prec$, and therefore it holds for $\preccurlyeq$ which is the reflexive, transitive closure of the former. □

[2]Note that the case $h = 2$ can never happen, for node n_2 can only broadcast values along channel c.

[3]This includes the location in which the value was originally stored itself.

Proposition 5.2.7. Let $l \leq k$ be a positive integer, and let buf be a mapping from the index set $\{1, \cdots, j\}$ to finite multisets of values.

Then $\mathcal{N}^l_{\text{buf}} \overset{\tau}{\Longmapsto} \mathcal{N}'$ if and only if $\mathcal{N}' \equiv \mathcal{N}^l_{\text{buf}'}$ for some mapping buf′ such that buf $\preccurlyeq$ buf′.

Proof. This statement follows directly from Proposition 5.2.5(4). Technical details are omitted. □

Propositions 5.2.5 and 5.2.7 allow us to list all the weak transitions that are defined for networks of the form $\mathcal{N}^l_{\text{buf}}$.

Corollary 5.2.8. Let $0 \leq l \leq k$, and let buf be a mapping from the index set $\{1, \cdots, j\}$ to finite multisets of values. Then

- For any channel $d \in \mathbf{Ch}$ and value $v \in \mathbf{Val}$, $\mathcal{N}^0_{\text{buf}} \overset{i.c?v}{\Longmapsto} \mathcal{N}'$ if and only if $\mathcal{N}' \equiv \mathcal{N}^0_{\text{buf}'}$ for some mapping buf′ such that buf $\preccurlyeq$ buf′
- For any channel $d \neq c$ and value $v \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\text{buf}} \overset{i.d?v}{\Longmapsto} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^{l+1}_{\text{buf}'}$ for some mapping buf′ such that buf $\preccurlyeq$ buf′
- For any value $v \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\text{buf}} \overset{i.c?v}{\Longmapsto} \mathcal{N}'$ if and only if $\mathcal{N}' \equiv \mathcal{N}^l_{\text{buf}'}$ for some mapping buf′ such that (buf $\downarrow$ v) $\preccurlyeq$ buf′
- for any value v, $\mathcal{N}^l_{\text{buf}} \overset{c!v \triangleright \{o\}}{\Longmapsto} \mathcal{N}'$ if and only if there exists an index $h = 1, \cdots, j$ such that $v \in \text{buf}(h)$ and $\mathcal{N}' \equiv \mathcal{N}^l_{\text{buf}'}$ for some mapping buf′ such that buf$[h \Uparrow v] \preccurlyeq$ buf′
- $\mathcal{N}^l_{\text{buf}}$ is deadlocked if and only if $\bigcup_{h=1}^{j} \text{buf}(h) = \emptyset$

Proof. All the statements can be proven by using propositions 5.2.5 and 5.2.7. For the last statement, it is also necessary to note that for any value v and index h such that $v \in \text{buf}(h)$ there exists a mapping buf′ such that buf $\preccurlyeq$ buf′ and $v \in \text{buf}(2)$. In other words, this means that every message which flows through the nodes of the network $\mathcal{N}_k$ eventually reaches the node n_2; this is true for n_2 is the only node in $\mathcal{N}_k$ for which the minimal length of a path between it and the external node o is exactly 1. □

At this point, we are ready to calculate the set of deadlock traces for the network $\mathcal{N}_k$; in practice, we compute the set of deadlock traces for any network of the form $\mathcal{N}^l_{\text{buf}}$.

Proposition 5.2.9. For any network of the form $\mathcal{N}^l_{\text{buf}}$, where $l \geq 0$ and buf is a mapping from the index set $\{1, \cdots, j\}$ to finite multisets of values, the set $\text{DTraces}(\mathcal{N}^l_{\text{buf}})$ is defined as the least set such that

- $\varepsilon \in \text{DTraces}(\mathcal{N}^l_{\text{buf}})$,
- If $\bigcup_{h=1}^{j} \text{buf}(i) = \emptyset$ then $\delta \in \text{DTraces}(\mathcal{N}^l_{\text{buf}})$
- If $t \in \text{DTraces}(\mathcal{N}^0_{\text{buf}'})$ and buf $\preccurlyeq$ buf′, then for any channel d and value v it holds $i.d?v::t \in \text{DTraces}(\mathcal{N}^0_{\text{buf}})$
- If $t \in \text{DTraces}(\mathcal{N}^{l+1}_{\text{buf}'})$ for some mapping buf′ such that buf $\preccurlyeq$ buf′, then for any channel $d \neq c$ and value $v \in \mathbf{Val}$ it holds $i.d?v::t \in \text{DTraces}(\mathcal{N}^l_{\text{buf}})$
- If $t \in \text{DTraces}(\mathcal{N}^l_{\text{buf}'})$ for some mapping buf′ such that (buf $\downarrow$ v) $\preccurlyeq$ buf′, then $i.c?v::t \in \text{DTraces}(\mathcal{N}^{l+1}_{\text{buf}})$
- If there exist an index h and a value v such that $v \in \text{buf}(h)$, and $t \in \text{DTraces}(\mathcal{N}^l_{\text{buf}'})$ for some mapping buf′ such that

$$(\text{buf}[h \Uparrow v]) \preccurlyeq \text{buf}'$$

  then $c!v \triangleright \{o\}::t \in \text{DTraces}(\mathcal{N}^l_{\text{buf}})$

Proof. Immediate from Corollary 5.2.8 and Definition 4.4.6 □

We are now ready to show that the network $\mathcal{N}_k$ is indeed an implementation of the routing model $\mathcal{M}_k$, defined in Section 5.2.2

Theorem 5.2.10. For any integer $k \geq 0$ we have $\mathcal{M}_k \simeq \mathcal{N}_k$.

Proof. Let $k \geq 0$; we prove that $\mathrm{DTraces}(\mathcal{M}_k) = \mathrm{DTraces}(\mathcal{N}_k)$. By Theorem 4.4.11 we have that $\mathcal{M}_k =_{\mathrm{must}} \mathcal{N}_k$, hence by Theorem 4.4.31 it follows that $\mathcal{M}_k \simeq \mathcal{N}_k$.

The proof is similar in style to that of Proposition 5.1.1. In this case we prove, by induction on a deadlock trace t, that $\mathrm{DTraces}(\mathcal{M}^l_{\mathscr{M}}) = \mathrm{DTraces}(\mathcal{M}^l_{\mathrm{buf}})$ for any multiset $\mathscr{M}$ and mapping buf such that $\mathscr{M} = \bigcup_{h=1}^{j} \mathrm{buf}(h)$. We only show the most interesting details.

- $t = \delta$; suppose that $\delta \in \mathcal{M}^l_{\mathscr{M}}$; it follows that $\mathscr{M} = \emptyset$, hence $\bigcup_{h=1}^{j} \mathrm{buf}(h) = \emptyset$. By Proposition 5.2.9 it follows that $\delta \in \mathcal{N}^l_{\mathrm{buf}}$. The opposite implication is analogous.

- $t = c!v \triangleright \{o\}::t'$; suppose that $t \in \mathcal{M}^l_{\mathscr{M}}$; then we have that $t' \in \mathcal{M}^l_{(\mathscr{M} \setminus \{\!|v|\!\})}$.

  Since $\bigcup_{h=1}^{j} \mathrm{buf}(h) = \mathscr{M}$, there exists an index h' such that $v \in \mathrm{buf}(h')$, from which it follows that $\mathcal{N}^l_{\mathrm{buf}} \stackrel{c!v \triangleright \{o\}}{\Longrightarrow} \mathcal{N}^l_{\mathrm{buf}'}$, where $\mathrm{buf}' = \mathrm{buf}[h \Uparrow v]$.

  By Hypothesis we have that $\mathrm{buf}(h') \cup \bigcup_{\substack{h=1 \\ h \neq h'}}^{j} \mathrm{buf}(h) = \mathscr{M}$, and in particular

$$\mathrm{buf}[h \Uparrow v] = (\mathrm{buf}(h') \setminus \{\!|v|\!\}) \cup \bigcup_{\substack{h=1 \\ h \neq h'}}^{j} \mathrm{buf}(h) = \mathscr{M} \setminus \{\!|v|\!\}$$

  Therefore, by Lemma 5.2.6, we find that $\bigcup_{h=1}^{j} \mathrm{buf}'(h) = \mathscr{M} \setminus \{\!|v|\!\}$. For we have already shown that $t' \in \mathcal{M}^l_{\mathscr{M} \{\!|v|\!\}}$, by inductive hypothesis we obtain that $t' \in \mathrm{DTraces}(\mathcal{N}^l_{\mathrm{buf}'})$. Finally, since we have already proved that $\mathcal{N}^l_{\mathrm{buf}} \stackrel{c!v \triangleright \{o\}}{\Longrightarrow} \mathcal{N}^l_{\mathrm{buf}'}$, we obtain that $c!v \triangleright \{o\}::t' \in \mathrm{DTraces}(\mathcal{N}^l_{\mathrm{buf}})$.

  The proof of the opposite implication is similar in style, and it is therefore omitted.

□

5.3 Connection Oriented Routing

In this Section we analyse a slightly different situation in which packets that flow through a network to reach a destination node are delivered in the same order in which they have been broadcast by the source node. In this case, the informal specification can be summarised as follows:

Specification 5.3.1 (Connection-Oriented Routing)**.** Give an input node i, an output node o and a channel c, the first k values broadcast by node i are eventually delivered, **in the same order they have been broadcast by** i, to the output node o. □ □

5.3.1 The Model

Not surprisingly, a model for Specification 5.3.1 can be defined as a slight modification of the network $\mathcal{M}_k$ as defined in Section 5.2.2. The multiset $\mathscr{M}$ which is used to represent the buffer of the unique internal node of a network is replaced with a queue q. A queue is a list of values which can be accessed and manipulated via the following functions: $\mathrm{push}(v, q, :)(\mathbf{Val} \times \mathbf{Val}^*) \to \mathbf{Val}^*, \mathrm{last}(q) : \mathbf{Val}^* \rightharpoonup \mathbf{Val}, \mathrm{remLast}(q) : \mathbf{Val}^* \to \mathbf{Val}^*$ and $\mathrm{isEmpty}(q) : \mathbf{Val}^* \to \{\mathsf{true}, \mathsf{false}\}$. Note that $\mathrm{last}(()\cdot)$ is a partial function. The definition of the functions we use to access a queue q is given below.

$$\begin{aligned}
\text{push}(v,q) &= v::q \\
\\
\text{last}(\varepsilon) &= \text{undefined} \\
\text{last}(v::\varepsilon) &= v \\
\text{last}(v::w::q) &= \text{last}(w::q) \\
\\
\text{remLast}(\varepsilon) &= \varepsilon \\
\text{remLast}(v::\varepsilon) &= \varepsilon \\
\text{remLast}(v::w::q) &= v::\text{remLast}(w::q) \\
\\
\text{isEmpty}(\varepsilon) &= \mathsf{true} \\
\text{isEmpty}(v::q) &= \mathsf{false}
\end{aligned}$$

For technical reasons we assume that the elements of the set of values **Val** are either *message values*, of the form v, or *ordered values* of the form $\langle l, v\rangle$, where l is an integer in the set $\{0,\cdots,k-1\}$. In the routing model we allow only message values to be routed from the source node to the destination, while ordered values broadcast by the input node i are ignored at the internal node of our model. Ordered values are not used in our routing model; however, they will be useful when defining one of its implementations. We assume a predicate $\text{isMessage}(()\cdot) : \mathbf{Val} \to \{\mathsf{true},\mathsf{false}\}$, defined as $\text{isMessage}(()\,v) = \mathsf{true}$, $\text{isMessage}(()\langle l, v\rangle) = \mathsf{false}$.

For any queue of (message) values q and integer $l \geq 0$, we assume the process definitions

$$P^0_q \Leftarrow \mathsf{if}\ \text{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ c!\langle\text{last}(q)\rangle.P^0_{\text{remLast}(q)}$$

$$\begin{aligned}
P^{l+1}_q \Leftarrow\ & (c?(x).\mathsf{if}\ \text{isMessage}(x)\ \mathsf{then}\ P^l_{\text{push}(x,q)}P^{l+1}_q\ \mathsf{else}\ ) + \\
+\ & (\mathsf{if}\ \text{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ c!\langle\text{last}(q)\rangle.P^l_{\text{remLast}(q)})
\end{aligned}$$

The routing model $\mathcal{M}_k$ for Specification 5.3.1 is defined as the network $\Gamma_M \triangleright M_k$, where Γ_M is the connectivity graph already used in the model of connectionless routing, depicted in Figure 5.3, and $M_k = m[\![P^k_\varepsilon]\!]$.

Let us discuss informally the behaviour of the network $\mathcal{M}_k$. At any given time, this network can perform one of the following actions:

- receive a value broadcast by the input node i along channel c; if the received value is a message value, and if the the network has not yet received k message values from the input node i, the value is stored as the top element in the queue q representing the internal buffer of node m, otherwise it is ignored
- if the internal buffer of node m is not empty, forward the last value in the queue along channel c; this value is detected by the output node o, which is in the range of transmission of node m.

Now we turn our attention to the formal behaviour of the routing model $\mathcal{M}_k$ we have defined. To this end, for any positive integer l and queue of message values v, let $\mathcal{M}^l_q$ be the network $\Gamma_M \triangleright m[\![P^l_q]\!]$.

Proposition 5.3.2. Let $l \geq 0$ and q be a queue of message values; then

1. If $\text{isEmpty}(q) = \mathsf{true}$ then $\mathcal{M}^l_q$ is deadlocked,
2. If $\text{isEmpty}(q) = \mathsf{false}$, then $\mathcal{M}^l_q \xmapsto{c!v\triangleright\{o\}} \mathcal{M}'$ if and only if $\text{last}(q) = v$ and $\mathcal{M}' \equiv \mathcal{M}^l_{\text{remLast}(q)}$,
3. For any channel $d \in \mathbf{Ch}$ and (either ordered or message) value v, $\mathcal{M}^0_q \xmapsto{i.d?v} \mathcal{M}'$ if and only if $\mathcal{M}' \equiv \mathcal{M}^0_q$,

4. For any channel $d \neq c$, and (either ordered or message) value v $\mathcal{M}_q^{l+1} \xmapsto{i.d?v} \mathcal{M}'$ if and only if $\mathcal{M}' \equiv \mathcal{M}_q^{l+1}$
5. If $\langle l' , v \rangle$ is an ordered value, then $\mathcal{M}_q^l \xmapsto{i.c?\langle l' , v \rangle} \mathcal{M}'$ if and only if $\mathcal{M}' \equiv \mathcal{M}_q^{l+1}$
6. If v is a message value then $\mathcal{M}_q^{l+1} \xmapsto{i.c?v} \mathcal{M}'$ if and only if $\mathcal{M}' \equiv \mathcal{M}^l_{\mathrm{push}(v,q)}$ □

It is also easy to check that there are no further transitions for such networks. Since there are no τ-transitions for any network of the form $\mathcal{M}_q^l$, it is easy to calculate its set of deadlock traces by exploiting proposition 5.3.2.

Corollary 5.3.3. For any integer $l \geq 0$ and queue q, the set $\mathrm{DTraces}(\mathcal{M}_q^l)$ is the least set such that

1. $\varepsilon \in \mathrm{DTraces}(\mathcal{M}_q^l)$,
2. If $\mathrm{isEmpty}(q) = \mathsf{true}$ then $\delta \in \mathrm{DTraces}(\mathcal{M}_q^l)$,
3. If $\mathrm{isEmpty}(q) = \mathsf{false}$, $\mathrm{last}(q) = v$ and $t \in \mathrm{DTraces}(\mathcal{M}^l_{\mathrm{remLast}(q)})$, then $c!v \triangleright \{o\}::t \in \mathrm{DTraces}(\mathcal{M}_q^l)$,
4. for any channel d and value v, if $t \in \mathrm{DTraces}(\mathcal{M}_q^0)$ then $i.d?v::t \in \mathrm{DTraces}(\mathcal{M}_q^0)$,
5. For any channel $d \in \mathbf{Ch}, d \neq c$ and (either message or ordered) value v, it $t \in \mathrm{DTraces}(\mathcal{M}_q^{l+1})$ then $i.d?v::t \in \mathrm{DTraces}(\mathcal{M}_q^{l+1})$,
6. For any ordered value $\langle l' , v \rangle$, if $t \in \mathrm{DTraces}(\mathcal{M}_q^{l+1})$ then $i.d?\langle l' , v \rangle::t \in \mathrm{DTraces}(\mathcal{M}_q^{l+1})$
7. If v is a message value and $t \in \mathrm{DTraces}(\mathcal{M}^l_{\mathrm{push}(v,q)})$, then $i.c?v::t \in \mathrm{DTraces}(\mathcal{M}_q^{l+1})$

Proof. Straightforward from Proposition 5.3.2 and Definition 4.4.7. □

5.3.2 Two Different Implementations

Now that we have defined a model for connection oriented routing, let us turn our attention at its possible implementations.

In the *TCP/IP standard*, protocols that ensure that the behaviour of a network is consistent with Specification 5.3.1 are implemented either at the *internet layer* or at the *transport layer*. In the first case, it is the responsibility of a routing protocol to ensure that packets are delivered to the destination in the same order they have been broadcast by the source. Here *The idea [..] is to avoid having to choose a new route for every packet sent [..]. Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup* [65].

In the second case, it is assumed that *users have no real control over the network layer [..]. The transport code runs entirely on the users' machines* [65]. In other words, this means that it is the responsibility of the source and destination nodes to ensure that packets are delivered to the latter in the same order they have been broadcast by the former; as for the other nodes in the network, their only responsibility is only that of forwarding each message they receive to the next hop in a routing path.

The fact that there exists different protocols that can be used to ensure that the behaviour of a network implements connection-oriented routing leads to the intuition that we can provide different implementations for the (connection-oriented) routing model $\mathcal{M}_k$ in our framework. In practice, we exhibit two of them. The first one reflects the behaviour of routing protocols defined at the internet layer; the latter abstracts from the behaviour of protocols defined at the transport layer of the TCP/IP standard. We remark that only this second implementation makes use of ordered values, which are instead ignored in the first one.

Implementation at the Internet Layer

As we have already mentioned, the idea behind connection-oriented routing protocols is that of choosing a route which will be used to make message values broadcast by the input node i to flow throughout a network, before reaching the destination o.

This amounts to discovering a path of minimal length in the connectivity graph we use in the implementation network; obviously, for this to be possible, we need to consider a connectivity graph in which the input node i is connected to the output node o.

In the same fashion of Section 5.2.3 we give a parametric definition of the implementation network; specifically, we consider a network $\mathcal{N}_k = \Gamma_N \triangleright N_k$ which satisfies the following properties:

- $\mathsf{Input}(\mathcal{N}_k) = \{i\}$, $\mathsf{Output}(\mathcal{N}_k) = \{o\}$; further, we assume that there is a single node n_1 such that $\Gamma_N \vdash i \to n_1$, a single node n_2 for which $\Gamma_N \vdash n_2 \to o$, and that $\mathrm{nodes}(N_k) = \{n_1, \cdots, n_j\}$ for some $h \geq 2$
- there exists a path in Γ_N from node n_1 to node n_2
- $\Gamma_N \vdash n_1 \nleftarrow n_2$
- for any index $h = 1, \cdots, j$ there exists a channel $c_h \in \mathbf{Ch}$
- for any integer $l \geq 0$, queue q and indexes $h = 3, \cdots, j$, $h' = 1, \cdots, j$ we assume the following process definitions:

$$
\begin{array}{lcl}
R^0_{q,0} & \Leftarrow & \mathsf{if}\ \mathrm{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else} \left(\displaystyle\sum_{h' \in \mathrm{next}(1,\Gamma_N)} c_{h'}!\langle \mathrm{last}(q) \rangle . R^0_{\mathrm{remLast}(q),h'} \right) \\
R^0_{q,h'} & \Leftarrow & \mathsf{if}\ \mathrm{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ c_{h'}!\langle \mathrm{last}(q) \rangle . R^0_{\mathrm{remLast}(q),h'} \\
R^{l+1}_{q,0} & \Leftarrow & (c?(x) . \mathsf{if}\ \mathrm{isMessage}(x)\ \mathsf{then}\ R^l_{\mathrm{push}(x,q),0}\ \mathsf{else}\ R^{l+1}_{q,0}) + \\
& + & \left[\mathsf{if}\ \mathrm{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else} \left(\displaystyle\sum_{h' \in \mathrm{next}(1,\Gamma_N)} c_{h'}!\langle \mathrm{last}(q) \rangle . R^{l+1}_{\mathrm{remLast}(q),h'} \right) \right] \\
R^{l+1}_{q,h'} & \Leftarrow & (c?(x) . \mathsf{if}\ \mathrm{isMessage}(x)\ \mathsf{then}\ R^l_{\mathrm{push}(x,q),h'}\ \mathsf{else}\ R^{l+1}_{q,h'}) + \\
& + & (\mathsf{if}\ \mathrm{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ c_{h'}!\langle \mathrm{last}(q) \rangle . R^{l+1}_{\mathrm{remLast}(q),h'})
\end{array}
$$

$$
\begin{array}{lcl}
Q^h_{q,0} & \Leftarrow & (c_h?(x) . Q^h_{\mathrm{push}(x,q),0}) + \\
& + & \left[\mathsf{if}\ \mathrm{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else} \left(\displaystyle\sum_{h' \in \mathrm{next}(h,\Gamma_N)} c_{h'}!\langle \mathrm{last}(q) \rangle . Q^h_{\mathrm{remLast}(q),h'} \right) \right] \\
Q^h_{q,h'} & \Leftarrow & (c_h?(x) . Q^h_{\mathrm{push}(x,v),h'}) + (\mathsf{if}\ \mathrm{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ c_{h'}!\langle \mathrm{last}(q) \rangle . Q^h_{\mathrm{remLast}(q),h'})
\end{array}
$$

$$
\begin{array}{lcl}
Q^2_{q,0} & \Leftarrow & (c_2?(x) . Q^2_{\mathrm{push}(x,q),0}) + (\mathsf{if}\ \mathrm{isEmpty}(q)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ c!\langle \mathrm{last}(q) \rangle . Q^2_{\mathrm{remLast}(q),0}) \\
Q^2_{q,h'} & \Leftarrow & Q^2_{q,0}
\end{array}
$$

Recall that, for any index $h = 1, \cdots, j$, $\mathrm{next}(h, \Gamma_N)$ contains all the nodes n' connected to it in $\mathcal{N}_k$ whose minimal length of a path from n' to o is strictly less than the minimal length of a path from n to o. The system term N_k is defined as

$$n_1[\![R^k_{\varepsilon,0}]\!] \,|\, \prod_{h=2}^{j} n_h[\![Q^h_{\varepsilon,0}]\!]$$

Let us discuss informally the behaviour of the network $\mathcal{N}_k$ we have defined. We have already seen in Section 5.2.3 that associating a channel c_h to each internal node n_h is useful to allow internal nodes to select one of its neighbours as the next hop in a routing path.

Each internal node $n \in \mathrm{nodes}(\mathcal{N}_k)$ is equipped with a queue q containing the messages it still has to forward to the next hop in a routing path. These are stored in descending order with respect to the time they have been detected by node n; in particular, the least recent received message is the last element of the queue, while the most recent is stored at the top of queue q.

Further, each node $n \in \mathrm{nodes}(\mathcal{N}_k)$ is equipped with a positive integer, which can be either 0 or be in the range $1, \cdots, j$. At least intuitively, such an integer corresponds to the next hop that the node has selected in a routing

path. If it is 0, then the next hop has not yet been selected, and it will be chosen non-deterministically among the neighbours of n; once such an index $h = 1, \cdot, j$ has been selected, node n will always select its neighbour n_h as the next hop in a routing path. This choice is made whenever the node n decides to broadcast its first value.

As in Section 5.2.3 the code running at the nodes n_1, n_2 is slightly different from that running at other nodes; we list briefly the main differences below:

- node n_1 only receives values which have been broadcast along channel c; further, it ignores ordered values, as well as message values broadcast after it has detected exactly k of them,
- node n_2 always broadcasts the value stored at the bottom of its queue along channel c.

Our aim is to prove that, for any index $k \geq 0$, it holds $\mathcal{M}_k \simeq \mathcal{N}_k$. This can be proved along the same lines of the proof of Theorem 5.2.10. However, the proof contains a greater amount of technical details, due to the more complicated structure of both the model $\mathcal{M}_k$ and the implementation $\mathcal{N}_k$.

First, let $\mathrm{buf} : \{1, \cdots, j\} \to \mathbf{Val}^*$, $\mathrm{nexthop} : \{1, \cdots, j\} \to \{0, \cdots, j\}$ be two functions. For any integer $l > 0$ we define the network $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ as the network

$$\Gamma_N \rhd n_1[\![R^l_{\mathrm{buf}(1),\mathrm{nexthop}(1)}]\!] \,|\, \prod_{h=2}^{j} n_h[\![Q^h_{\mathrm{buf}(h),\mathrm{nexthop}(h)}]\!]$$

It is easy to check that $\mathcal{N}_k = \mathcal{N}^k_{\mathrm{buf}_0,\mathrm{nexthop}_0}$, where $\mathrm{buf}_0(h) = \varepsilon$, $\mathrm{nexthop}_0(h) = 0$ for any $h = 1, \cdots, j$. We also say that a network $\mathcal{N}^k_{\mathrm{buf,nexthop}}$ is *admissible* if and only if, for any index $h = 1, \cdots, j$ if $\mathrm{buf}(h) \neq \varepsilon$ then there exists a node $n_{h'}$ such that $\mathrm{nexthop}(h') = h$, and whenever $\mathrm{buf}(h) \neq \varepsilon$ then it contains only message values. Informally speaking, a node can have values stored in its buffer only if is has been selected as a hop in the routing selected for the network $\mathcal{N}_k$. Further, only message values can flow throughout the network. It is easy enough to note that $\mathcal{N}_k = \mathcal{N}^k_{\mathrm{buf}_0,\mathrm{nexthop}_0}$ is admissible.

For any mapping $\mathrm{buf} : \{1, \cdots, k\} \to \mathbf{Val}^*$ and indexes $h, h' = 1, \cdots, j$, we use the notation $\mathrm{buf}[h \rightsquigarrow h']$ as a shortcut for $(\mathrm{buf}[h \mapsto \mathrm{remLast}(\mathrm{buf}(h))])[h' \mapsto \mathrm{push}(\mathrm{last}(\mathrm{buf}(h)), \mathrm{buf}(h'))]$, meaning that a value v has flown from the bottom of the queue of node n_h to the top of the queue of node $n_{h'}$.

We also use the notation $(\mathrm{buf} \downarrow v)$ as a shortcut for $\mathrm{buf}[1 \mapsto \mathrm{push}(v, \mathrm{buf}(1))]$, meaning that value v has been pushed on the top of the queue of node n_1. Similarly, we use the notation $(f \uparrow)$ for $f[2 \mapsto \mathrm{remLast}(\mathrm{buf}(2))]$, meaning that the last element in the queue of the node n_2 has been removed from the network $\mathcal{N}_k$.

We are now ready to list the properties of networks of the form $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ that we need to prove that $\mathcal{M}_k \simeq \mathcal{N}_k$.

Proposition 5.3.4. Let $l \geq 0$, $\mathrm{buf} : [1, \cdots, j] \to \mathbf{Val}^*$, $\mathrm{nexthop} : [1, \cdots, j] \to [0, \cdots, j]$.

1. For any channel $d \in \mathbf{Ch}$, (either message or ordered) value $v \in \mathbf{Val}$, $\mathcal{N}^0_{\mathrm{buf,nexthop}} \xmapsto{i.d?v} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^0_{\mathrm{buf,nexthop}}$,
2. For any channel $d \neq c$ and value $v \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\mathrm{buf,nexthop}} \xmapsto{i.d?v} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^{l+1}_{\mathrm{buf,nexthop}}$,
3. For any ordered value $\langle l', v \rangle \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\mathrm{buf,nexthop}} \xmapsto{i.c?\langle l', v \rangle} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^{l+1}_{\mathrm{buf,nexthop}}$,
4. for any message value $v \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\mathrm{buf,nexthop}} \xmapsto{i.c?v} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^l_{(\mathrm{buf}\downarrow v),\mathrm{nexthop}}$,
5. $\mathcal{N}^l_{\mathrm{buf,nexthop}} \xmapsto{\tau} \mathcal{N}'$ if and only if there exists an index $h \neq 2$ such that $\mathrm{buf}(h) \neq \varepsilon$ and
   (a) if $\mathrm{nexthop}(h) = 0$ then $\mathcal{N}' = \mathcal{N}^l_{(\mathrm{buf}[h \rightsquigarrow h']),(\mathrm{nexthop}[h \mapsto h'])}$ for some index h' such that $h' \in \mathrm{next}(h, \Gamma_N)$
   (b) if $\mathrm{nexthop}(h) = h'$, $h' \neq 0$, then $\mathcal{N}' = \mathcal{N}^l_{(\mathrm{buf}[h \rightsquigarrow h']),\mathrm{nexthop}}$
6. $\mathcal{N}^l_{\mathrm{buf,nexthop}} \xmapsto{c!v \rhd \{o\}} \mathcal{N}'$ if and only if $\mathrm{last}(\mathrm{buf}(2)) = v$ and $\mathcal{N}' = \mathcal{N}^l_{(\mathrm{buf}\uparrow),\mathrm{nexthop}}$

7. $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ is deadlocked if and only if $\mathrm{buf}(h) = \varepsilon$ for any index h ranging over $1, \cdots, j$,

8. if $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ is admissible, then whenever such that $\mathcal{N}^l_{\mathrm{buf,nexthop}} \overset{\lambda}{\longmapsto} \mathcal{N}'$ for some extensional action λ it holds that $\mathcal{N}'$ is admissible.

Proof. Similar in style to that of Proposition 5.2.5. □

The last statement of Proposition 5.3.4 is particularly important; it says that admissible networks are invariant by transitions. For our implementation $\mathcal{N}_k$ is itself an admissible network, it follows that any state of the extensional LTS it generates corresponds to an admissible network as well. Further, the reader can check that for any integer $k \geq 0$, the network $\mathcal{N}_k$ is finitary and strongly convergent.

The next step is that of identifying the weak extensional transitions of an admissible network of the form $\mathcal{N}^l_{\mathrm{buf,nexthop}}$. To this end, it is first necessary to give some definitions.

Given two mappings $\mathrm{buf} : \{1, \cdots, j\} \to \mathbf{Val}^*$, $\mathrm{nexthop} : \{1, \cdots, j\} \to \{0, \cdots, j\}$ and an index $h = 1, \cdots, j$, we define the the queue $\mathrm{concat}(f, g, h)$ as

$$\begin{array}{lcll} \mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}, 2) & = & \mathrm{buf}(2) & \\ \mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}, h) & = & \mathrm{buf}(h) & \text{if} \quad h \neq 2, \mathrm{nexthop}(h) = 0 \\ \mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}, h) & = & \mathrm{buf}(h) :: \mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}, \mathrm{nexthop}(h)) & \text{if} \quad h \neq 2, \mathrm{nexthop}(h) \neq 0 \end{array}$$

where $q :: q'$ represents, with an abuse of notation, the concatenation of two queues q, q'. With an abuse of notation, we use $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop})$ in lieu of $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}, 1)$; it is easy to check that, if $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ is an admissible network, then $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop})$ is well-defined, and it corresponds to the queue of messages stored in the overall network $\mathcal{N}^l_{\mathrm{buf,nexthop}}$, ordered according to the routing path described in nexthop.

We will also need to identify the index of the node in which the bottom element (if any) of $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop})$ is stored, respectively to the network $\mathcal{N}^l_{\mathrm{buf,nexthop}}$. To this end, we define a function $\mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, h)$ that maps to every mapping $\mathrm{buf}, \mathrm{nexthop}$ and index h either the value 0, in the case that $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}, h) = \varepsilon$, or the index where the last value of $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}, h)$ is stored in the network $\mathcal{N}^l_{\mathrm{buf,nexthop}}$. Its formal definition is provided below.

$$\begin{array}{lcl} \mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, 0) & = & 0 \\ \mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, 2) & = & \begin{cases} 0 & \text{if } \mathrm{buf}(2) = \varepsilon \\ 2 & \text{otherwise} \end{cases} \\ \mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, h) & = & \begin{cases} 0 & \text{if } \mathrm{buf}(h) = \varepsilon, \mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, \mathrm{nexthop}(h)) = 0 \\ h & \text{if } \mathrm{buf}(h) \neq \varepsilon, \mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, \mathrm{nexthop}(h)) = 0 \\ \mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, \mathrm{nexthop}(h)) & \text{if } \mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, \mathrm{nexthop}(h)) \neq 0 \end{cases} \end{array}$$

In the definition above h ranges over $1, \cdots, j$, and we assume that $h \neq 2$. We use the notation $\mathrm{gli}(\mathrm{buf}, \mathrm{nexthop})$ in lieu of $\mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}, 1)$; again, the latter function is well-defined if $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ is an admissible network. Given two mappings $\mathrm{buf}, \mathrm{nexthop}$ as above, we use the notation $(\mathrm{buf} \Uparrow \mathrm{nexthop})$ to denote the mapping from the index set $\{1, \cdots, j\}$ to queues of values defined as

$$(\mathrm{buf} f \Uparrow \mathrm{nexthop}) = \mathrm{buf}[\mathrm{gli}(\mathrm{buf}, \mathrm{nexthop}) \mapsto \mathrm{remLast}(\mathrm{buf}(\mathrm{gli}(\mathrm{buf}, \mathrm{nexthop})))]$$

Intuitively, the mapping $(\mathrm{buf} \Uparrow \mathrm{nexthop})$ corresponds to the mapping buf, in which the last element stored in the overall network $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ (according to the next-hop function nexthop), that is the bottom element of $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop})$, has been removed.

Finally, consider some mappings $\mathrm{buf}, \mathrm{buf}' : \{1, \cdots, j\} \mathbf{Val}^*, \mathrm{nexthop}, \mathrm{nexthop}' : \{1, \cdots, j\} \to \{0, \cdots, j\}$; we say that $\langle \mathrm{buf}, \mathrm{nexthop} \rangle \prec \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$ if and only if

- for any index h such that $\mathrm{nexthop}(h) \neq 0$ it holds $\mathrm{nexthop}(h) = \mathrm{nexthop}(h')$
- if $\mathrm{nexthop}(h) = 0$ and $\mathrm{nexthop}'(h) \neq 0$, then $\mathrm{buf}' = \mathrm{buf}[h \rightsquigarrow h']$, where $h' = \mathrm{nexthop}'(h)$,
- otherwise, $\mathrm{nexthop} = \mathrm{nexthop}'$ and there exists two indexes h, h' such that $\mathrm{nexthop}(h) = h'$ and $\mathrm{buf}' = \mathrm{buf}[h \rightsquigarrow h']$

We use $\preccurlyeq$ to denote the reflexive, transitive closure of $\prec$. Intuitively $\langle \mathrm{buf}, \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$ if and only if the routing path described by $\mathrm{nexthop}'$ is an extension of that modelled by nexthop, and messages stored at some node n_h in a network $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ have flown along the established routing path and are stored in a node n'_h in the network $\mathcal{N}^l_{\mathrm{buf}'}\mathrm{nexthop}'$.

We have provided all the definitions that we need to identify the weak extensional transitions of admissible networks. First, we note that the relation $\preccurlyeq$ we have defined is closely connected to weak τ-extensional transitions.

Proposition 5.3.5. Let $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ be an admissible network. $\mathcal{N}^l_{\mathrm{buf,nexthop}} \stackrel{\tau}{\Longrightarrow} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^l_{\mathrm{buf}',\mathrm{nexthop}'}$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\mathcal{N}^l_{\mathrm{buf}',\mathrm{nexthop}'}$ is admissible, and $\langle \mathrm{buf}, \mathrm{nexthop}, \preccurlyeq \rangle \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$. □

Proposition 5.3.5 allows us to identify the weak extensional transitions performed for both the network $\mathcal{N}_k$ and the states of the extensional LTS it generates.

Corollary 5.3.6. Let $l \geq 0$, $h \in \{1, \cdots, j\}$ and $\mathrm{buf} : \{1, \cdots, j\} \to \mathbf{Val}^*$, $\mathrm{nexthop} : \{1, \cdots, j\} \to \{0, \cdots, j\}$. Suppose $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ is an admissible network.

1. For any channel $d \in \mathbf{Ch}$ and (either ordered or message) value $v \in \mathbf{Val}$, $\mathcal{N}^0_{\mathrm{buf,nexthop}} \stackrel{i.d?v}{\Longrightarrow} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^0_{\mathrm{buf}',\mathrm{nexthop}'}$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle \mathrm{buf}, \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$
2. For any channel $d \neq c$ and (either ordered or message) value $v \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\mathrm{buf,nexthop}} \stackrel{i.d?v}{\Longrightarrow} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^{l+1}_{\mathrm{buf}',\mathrm{nexthop}'}$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle f, g \rangle \preccurlyeq \langle f', g' \rangle$,
3. For any ordered value $\langle l', v \rangle \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\mathrm{buf,nexthop}} \stackrel{i.c?\langle l', v \rangle}{\Longrightarrow} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^{l+1}_{\mathrm{buf}',\mathrm{nexthop}'}$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle \mathrm{buf}, \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$,
4. for any message value $v \in \mathbf{Val}$, $\mathcal{N}^{l+1}_{\mathrm{buf,nexthop}} \stackrel{i.c?v}{\Longrightarrow} \mathcal{N}'$ if and only if $\mathcal{N}' = \mathcal{N}^l_{(\mathrm{buf}',\mathrm{nexthop}'}$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle (\mathrm{buf} \downarrow v), \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$,
5. $\mathcal{N}^l_{\mathrm{buf,nexthop}} \stackrel{c!v \triangleright \{o\}}{\Longrightarrow} \mathcal{N}'$ if and only last(concat(buf, nexthop)) is defined and equal to some value v, and $\mathcal{N}' = \mathcal{N}^l_{\mathrm{buf}',\mathrm{nexthop}'}$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle (\mathrm{buf} \Uparrow \mathrm{nexthop}), \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$. □

We are now ready to calculate the set of deadlock traces for any admissible network of the form $\mathcal{N}^l_{\mathrm{buf,nexthop}}$.

Proposition 5.3.7. Let $l \geq 0$; for any admissible network of the form $\mathcal{N}^l_{\mathrm{buf,nexthop}}$ the set $\mathrm{DTraces}(\mathcal{N}^l_{\mathrm{buf,nexthop}})$ is the least set such that

1. $\varepsilon \in \mathrm{DTraces}(\mathcal{N}^l_{\mathrm{buf,nexthop}})$,
2. If $\mathrm{concat}((,\mathrm{buf}), \mathrm{nexthop}) = \varepsilon$ then $\delta \in \mathrm{DTraces}(\mathcal{N}^l_{\mathrm{buf,nexthop}})$,
3. For any channel $d \in \mathbf{Ch}$ and (either message or ordered) value $v \in \mathbf{Val}$, if $t \in \mathrm{DTraces}(\mathcal{N}^0_{\mathrm{buf}',\mathrm{nexthop}'})$ for some $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle \mathrm{buf}, \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$, then $i.d?v::t \in \mathrm{DTraces}(\mathcal{N}^0_{\mathrm{buf,nexthop}})$
4. For any channel $d \in \mathbf{Ch}$ such that $d \neq c$, and (either ordered or message) value v, if $t \in \mathrm{DTraces}(\mathcal{N}^{l+1}_{\mathrm{buf}',\mathrm{nexthop}'})$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle \mathrm{buf}, \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$, then $i.d?v::t \in \mathrm{DTraces}(\mathcal{N}^{l+1}_{\mathrm{buf,nexthop}})$,

5. For any channel $d \in \mathbf{Ch}$ and ordered value $\langle l', v \rangle$, if $t \in \mathrm{DTraces}(\mathcal{N}^{l+1}_{\mathrm{buf}',\mathrm{nexthop}'})$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle \mathrm{buf}, \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$, then $i.d?\langle l', v \rangle :: t \in \mathrm{DTraces}(\mathcal{N}^{l+1}_{\mathrm{buf},\mathrm{nexthop}})$,

6. Let v be a message value such that $t \in \mathrm{DTraces}(\mathcal{N}^{l}_{\mathrm{buf}',\mathrm{nexthop}'})$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ for which $\langle (\mathrm{buf} \downarrow v), \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$ holds; then $i.c?v :: t \in \mathrm{DTraces}(\mathcal{N}^{l+1}_{\mathrm{buf},\mathrm{nexthop}})$,

7. If $t \in \mathrm{DTraces}(\mathcal{N}^{l}_{\mathrm{buf}',\mathrm{nexthop}'})$ for some mappings $\mathrm{buf}', \mathrm{nexthop}'$ such that $\langle (\mathrm{buf} \Uparrow \mathrm{nexthop}), \mathrm{nexthop} \rangle \preccurlyeq \langle \mathrm{buf}', \mathrm{nexthop}' \rangle$ and $\mathrm{last}(\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop})) = v$, then $c!v \triangleright \{o\} :: t \in \mathrm{DTraces}(\mathcal{N}^{l}_{\mathrm{buf},\mathrm{nexthop}})$. □

We have stated all the results that we need to prove that, for any positive integer k, the network $\mathcal{N}_k$ is an implementation of the model $\mathcal{M}_k$.

Theorem 5.3.8. For any network of the form $\mathcal{M}^l_q$ and admissible network of the form $\mathcal{N}^l_{\mathrm{buf},\mathrm{nexthop}}$, if $\mathrm{concat}(\mathrm{buf}, \mathrm{nexthop}) = q$ then $\mathrm{DTraces}(\mathcal{M}^l_q) = \mathrm{DTraces}(\mathcal{N}^l_{\mathrm{buf},\mathrm{nexthop}})$.

In particular, since for any $k \geq 0$ it holds $\mathcal{M}_k = \mathcal{M}^k_\varepsilon$ and $\mathcal{N}_k = \mathcal{N}^k_{\mathrm{buf}_0,\mathrm{nexthop}_0}$, we have that $\mathrm{DTraces}(\mathcal{M}_k) = \mathrm{DTraces}(\mathcal{N}_k)$. □

Implementation at the Transport Layer

In this Section we provide another implementation of the connection-oriented routing model defined in Section 5.3.1; this can be seen as an abstraction of protocols defined at the transport layer of the TCP/IP standard.

Specifically, we assume that it is the responsibility of the source node i and the destination node o to ensure that values are delivered to the latter in the same order they have been broadcast by the former; instead, the role of internal nodes in a network will be only that of forwarding the values they have received to the next hop in a routing path, without performing any control on the order in which such values have been broadcast by the input node i.

In this case, one problem arise. For we are assuming that the source node i and the destination node o are external nodes in the routing model $\mathcal{M}_k$, we are not allowed to insert any code at such nodes in our implementation. In practice, this problem can be avoided by requiring that the input node i can broadcast values to a unique (internal) node n_1, while the output node can only detect values broadcast from a different node n_2; at this point we can insert some code at the internal nodes n_1, n_2 to ensure that messages broadcast by node i are delivered to node o in the same order.

Roughly speaking, one could view the nodes i and n_1 as a single (wireless) station; node n_1 contains the code that such a station run at the transport layer, while messages broadcast from the input node i as requests that the protocol running at the higher level (that is, the application layer) of the station forwards to the transport layer. The same can be thought for the nodes n_2 and o. Node n_2 provides services at the transport layer to the output node o, by processing the data it receives by other nodes in the network.

Formally, we consider a network $\mathcal{L}_k = \Gamma_L \triangleright L_k$ for which the connectivity graph Γ_L satisfies the same properties of Γ_N defined in Section 5.3.1. Before defining the system term L_k we need to provide the following process definitions:

$$
\begin{aligned}
R^{0,l'}_{\mathscr{M}} &\Leftarrow \left[\sum_{v \in \mathscr{M}} \left(\sum_{h \in \mathrm{next}(1,\Gamma_L)} c_h!\langle v \rangle . S^{0,l'}_{(\mathscr{M} \setminus \{v\})} \right)\right] \\
R^{(l+1),l'}_{\mathscr{M}} &\Leftarrow \left(c?(x).\mathsf{if}\ \mathrm{isMessage}(x)\ \mathsf{then}\ \left(S^{l,(l'+1)}_{(\mathscr{M} \cup \{\langle l', x \rangle\})}\right)\ \mathsf{else}\ S^{l+1} l'_{\mathscr{M}}\right) + \\
&+ \left[\sum_{v \in \mathscr{M}} \left(\sum_{h \in \mathrm{next}(1,\Gamma_L)} c_h!\langle v \rangle . S^{(l+1),l'}_{(\mathscr{M} \setminus \{v\})} \right)\right] \\
Q^h_{\mathscr{M}} &\Leftarrow \left(c_h?(x).Q^h_{(\mathscr{M} \cup \{x\})}\right) + \left[\sum_{v \in \mathscr{M}} \left(\sum_{h' \in \mathrm{next}(h,\Gamma_L)} c!\langle v \rangle . Q^h_{(\mathscr{M} \setminus \{x\})} \right)\right] \\
S^{l'}_{\mathscr{M}} = &\Leftarrow (c_2?(x).R^{l'}_{\mathscr{M}}) + \left(\sum_{\langle l', v \rangle \in \mathscr{M}} c!\langle v \rangle . S^{l'+1} \right)
\end{aligned}
$$

Here we assume that $\mathscr{M}$ is a multiset, l, l', h are positive integers. Then the system term L_k is defined as

$$n_1[\![R_\emptyset^{k,0}]\!] \,|\, n_2[\![S_\emptyset^0]\!] \,|\, \prod_{h=3}^{j} n_h[\![Q_\emptyset^h]\!]$$

Informally speaking, the behaviour of the network $\mathcal{L}_k$ is similar to that of the implementation we have provided for the connectionless routing model, Section 5.2.3. In fact, the only differences lie at the code defined for the nodes n_1, n_2; let us illustrate them briefly.

- Suppose node n_1 has not yet received k (message) values from the input node i. If n_1 receives a (message) value, broadcast from the input node i, it assigns to it a *sequence number* l', and it stores the (ordered) value $\langle l', v \rangle$ in its own buffer, represented by a multiset $\mathscr{M}$.

  Further, it increases the sequence number which will be assigned to the next message value it detects by 1. This behaviour ensures that sequence numbers are assigned to message values in increasing order with respect to the order in which they have been broadcast by the input node i.

- If the last value that node n_2 has forwarded to node o had assigned a sequence number l', it will wait to receive (along channel c_2) an ordered value of the form $\langle l'+1, v \rangle$. Whenever such a value is stored in the local buffer of node n_2, then such a node will remove the sequence number from it and the obtained (message) value will be delivered to node o. After the latter has been broadcast, node n_2 will update the sequence number of the next value to be forwarded to node o to $l'+2$.

The user should be convinced that message values broadcast by node i will be eventually delivered to the destination o. Further, node the code running at nodes n_1 and n_2 ensures that messages are delivered at the destination in the same order they have been broadcast by the source node i. We only state the main result, without providing any details.

Theorem 5.3.9. For any index $k > 0, \mathcal{M}_k \simeq \mathcal{L}_k$. □

5.4 Multicast Routing

In this Section we consider the problem of routing a fixed number of messages from a source node to a group of destination nodes.

The problem of multicast routing can be summarised as follows: *in some applications, hosts need to send messages to many or all other hosts. For example, a service distributing weather reports, stock market updates, or live radio programs might work best by broadcasting to all machines and letting those that are interested read the data.[..] One broadcasting method [..] is for the source to simply send a distinct packet to each destination.* [65].

This description suggests a very simple Specification of the behaviour we wish to model.

Specification 5.4.1 (Multicast Routing)**.** Let j, k be two positive integers. Given an input node i and a sequence $o_1, \cdots, o_j$ of output nodes, the first k messages broadcast by the input node i will eventually be detected by all the output nodes $o_1, \cdots, o_k$. □

Finding a suitable model for the Specification above, however, is a non-trivial task. For example, given an integer $l \geq 0$ and a multiset $\mathscr{M}$, assume that we have the process definitions

$$\begin{aligned} P^0_{\mathscr{M}} &\Leftarrow \sum_{v \in \mathscr{M}} c!\langle v \rangle . P^0_{(\mathscr{M} \setminus \{v\})} \\ P^{l+1}_{\mathscr{M}} &\Leftarrow (c?(x) . P^l_{(\mathscr{M} \cup \{x\})}) + \left(\sum_{v \in \mathscr{M}} c!\langle v \rangle . P^0_{(\mathscr{M} \setminus \{v\})} \right) \end{aligned}$$

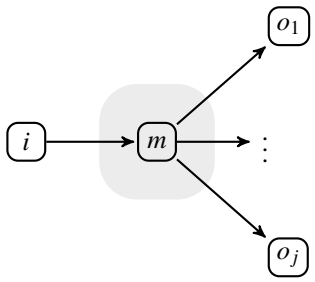


Figure 5.5: An ill-formed connectivity graph Γ'_M for modelling multicast routing.

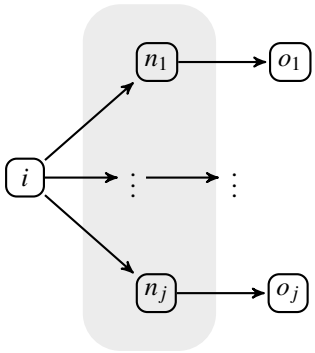


Figure 5.6: A correct connectivity graph Γ_M for modelling multicast routing.

Consider the network $\mathcal{M}'_k = \Gamma'_M \triangleright M'_k$, where $M'_k = m[\![P^k_\emptyset]\!]$.

Intuitively, node m in network $\mathcal{M}'_k$ can receive at most k values broadcast by the input node i, and each of such values is stored in the local buffer of the unique internal node. Whenever such a buffer is not empty, node m can decide to broadcast one of the values stored in it to all the output nodes $o_1, \cdots, o_j$.

The behaviour of the network $\mathcal{M}'_k$ is indeed consistent with Specification 5.4.1; that is, the first k messages broadcast by node i will eventually be detected by each of the output nodes $o_1, \cdots, o_j$. However, the behaviour of $\mathcal{M}'_k$ is too much specific. In fact, it is straightforward to note that any of the first k values broadcast by the input node i is detected by each of the output nodes $o_1, \cdots, o_k$ **at the same time**. The constraint that each values should be received simultaneously at all output nodes is not included in the specification we have provided.

Another alternative would be that of using the network $\mathcal{M}_k = \Gamma_M \triangleright M_k$, where Γ_M is the connectivity graph depicted in Figure 5.6 and

$$M_k = \prod_{i=1}^{j} n_j[\![P^k_\emptyset]\!]$$

In this network the first k values broadcast by node i are stored at j independent internal nodes $n_1, \cdots, n_j$; each of these nodes $n_h, h = 1, \cdots, j$ will eventually forward the detected values to the corresponding output node o_h. Since the locations $n_1, \cdots, n_j$ are independent each other, in this case it is straightforward to show that any value broadcast by i can be detected by each of the output nodes $o_1, \cdots, o_j$ at different times.

For the network $\mathcal{M}_k$ has a more general behaviour than that of the ill-formed model $\mathcal{M}'_k$, we expect the latter to be a *refinement* of the former. In fact, it is easy to check that the inequivalence $\mathcal{M}_k \sqsubseteq_{\text{must}} \mathcal{M}'_k$ holds for any integer $k \geq 0$. However, the inequivalence $\mathcal{M}_k \sqsubseteq_{\text{may}} \mathcal{M}'_k$ does not hold. The proof of these two statements is similar in style to those provided in examples 4.3.28 and 4.4.35.

Let us turn our attention to a possible implementation of the multicast model $\mathcal{M}_k$. For the sake of simplicity, we only consider the case in which the number of internal nodes in the model is $j = 2^{j'}$ for some index $j' \geq 0$; however, the implementation we discuss can be easily generalised to any possible value of j.

Roughly speaking, our implementation is obtained by replacing the internal nodes $n_1, \cdots, n_{2^{j'}}$ in the model $\mathcal{N}_k$ with a complete binary tree consisting of $2^{j'+1} - 1$ internal nodes. Formally, we assume that our implemen-

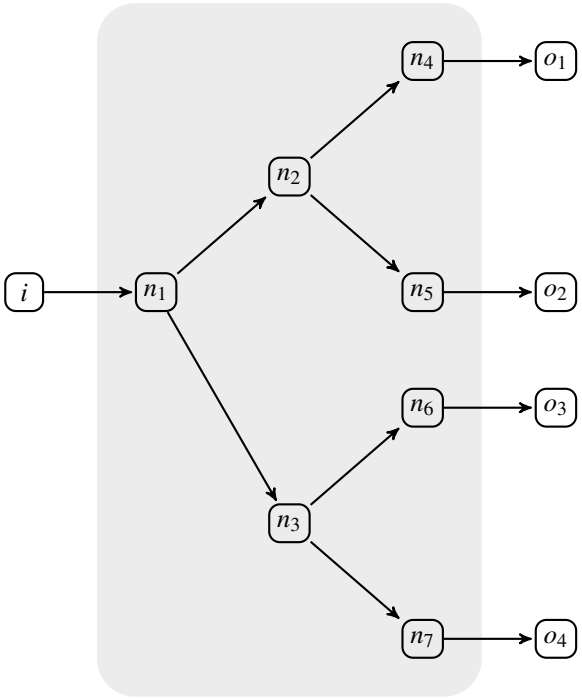


Figure 5.7: An instance of the connectivity graph Γ_N

tation is a network $\mathcal{N}_k = \Gamma_N \triangleright N_k$ which satisfies the following constraints:

- $\text{nodes}(N_k) = \{n_1, \cdots, n^{j'+1} - 1\}$,
- Γ_N is the smallest connectivity graph such that $\Gamma_N \vdash i \to n_1$, for any index $h : 0 \leq h \leq 2^{j'} - 1$ it holds $\Gamma_N \vdash n_h \to n_{2h}, \Gamma_N \vdash n_h \to n_{2h+1}$ and for any index $h : 0 \leq h \leq 2^{j'} - 1$, $\Gamma_N \vdash n_{2^{j'}+h} \to o_{h+1}$
- for any multiset of values $\mathscr{M}$, assume the process definition

$$P_{\mathscr{M}} \Leftarrow (c?(x).P_{(\mathscr{M} \cup \{x\})}) + \left(\sum_{v \in \mathscr{M}} c!\langle v \rangle .P_{(\mathscr{M} \setminus \{v\})} \right)$$

  Then the system term N_k is defined as

$$N_k = n_1[\![P_\emptyset^k]\!] \mid \prod_{h=2}^{2^{j'+1}-1} n_h[\![P_\emptyset]\!]$$

An instance of the connectivity graph Γ_N, in the case we assume $j' = 2$, is provided in Figure 5.7.

Intuitively speaking, in the network $\mathcal{N}_k$ the node n_1 is responsible to detect the first k values broadcast by the input node i and store them in its local buffer, represented as a multiset $\mathscr{M}$. Whenever the local buffer of such a node is non-empty, it can non-deterministically decide to forward one of the values stored in it along channel c; the children nodes of n_1, that is nodes n_2 and n_3, detect the value broadcast by their father node and store it in their local buffer.

This procedure is iterated until a value does not reach a leaf node $n_{2^{j'}+h}$, where $0 \leq h \leq \cdots, 2^{j'} - 1$. In this case, the latter node can non-deterministically decide to broadcast a value stored in its own local buffer (if any), which will be detected only by the output node o_{h+1}.

In order to show that the network $\mathcal{N}_k$ is an implementation of the multicast model $\mathcal{M}_k$, it is necessary to calculate both the sets $\text{DTraces}(\mathcal{M}_k), \text{DTraces}(\mathcal{N}_k)$. This can be done in a way analogous to the one already illustrated in the previous sections of this Chapter. We only state the main result.

Theorem 5.4.2. For any integer $k \geq 0$, $\text{DTraces}(\mathcal{M}_k) = \text{DTraces}(\mathcal{N}_k)$. Therefore, $\mathcal{M}_k \simeq \mathcal{N}_k$. □

5.5 Virtual Shared Memory

Our last case study focuses on implementing a *virtual shared memory* in a *distributed system*.

A distributed system is a collection of independent computers that appears to its users as a single coherent system.[..] The main goal of a distributed system is to make it easy for users to access remote resources, and to share them with other users in a controlled way.[..] An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers [66].

In a distributed system, a virtual shared memory is a service that allows different machines, called clients, to retrieve and edit data which are distributed among different computers. That is, a virtual shared memory can be seen as an implementation in a distributed system of a shared memory service running on a single computer.

Our first goal in this Section is that of exhibiting a model for virtual shared memory; this can be done by implementing a shared memory system in which all the data are stored on a single machine, to which clients can send either reading or writing requests.

Then we provide an implementation by focusing on a more complicated network, in which data are partitioned among different machines. For the sake of simplicity, we assume that the number of clients which can access the (virtual) shared memory is fixed, and equal to 2.

Specification 5.5.1 (Virtual Shared Memory)**.** Let cl_1, cl_2 be two client nodes, mVar be a finite set of memory variables, $\sigma : \text{mVar} \to \mathbb{Z}$ be a memory and k be a positive integer.

A command request issued from a client is a value either of the form READ(x) (read requests) or UPDATE(x, z) (write requests), where $x \in \text{mVar}$ and $z \in \mathbb{Z}$.

A network $\mathcal{M}_k^\sigma$ is a (virtual) shared memory system if it processes the first k request commands broadcast from client cl_1 and the first k request commands broadcast from client cl_2 as follows:

- the network eventually replies to any read request READ(x) performed by client cl_i, $i = 1, 2$; the reply consists of the value $\sigma(x)$, and can be detected only by the client that performed the request,
- the network eventually updates the memory whenever a write request UPDATE(x, z) is broadcast by a client. Upon receiving such a request, the memory of the network evolves by updating the value $\sigma(x)$ to be exactly z, and it replies to the client that has performed the request with the new value of $\sigma(x)$.
- Requests performed by the node client cl_1 are processed in the same order they have been broadcast; the same holds for the node client c_2.

□

Defining a model for Specification 5.5.1 is not an easy task. In fact, there are several details to be taken into account. First, we want the requests performed by a client to be processed in the same order they have been broadcast. We have already seen that, in order to ensure that values are processed in the same order they are detected by the network, it is necessary to equip the network with a queue. In this case, there are two different queues, one for each client node.

Second, we need to equip the network with a memory, that is a function σ that maps memory variables into integer values. Finally, we need to perform a check whether a value broadcast by a client corresponds to a command request.

In practice, the connectivity graph Γ_M we use to define a model for Specification 5.5.1, depicted in Figure 5.8, consists of 5 internal nodes. The node mem is the one in which the memory is implemented. The node i_1 processes the requests broadcast by the client node cl_1, selecting only those which are valid, while the node o_1 receives the replies from the memory node mem and forward them to the client node cl_1. In a similar way, nodes i_2 and o_2 handle the requests and the replies coming from and directed to the client node cl_2, respectively.

Before defining the system term containing the code at the 5 internal nodes of $\mathcal{M}$, we need to provide some definitions.

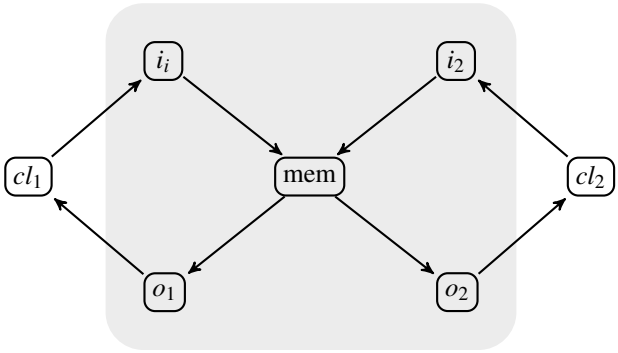


Figure 5.8: The connectivity graph of the Virtual Shared Memory model

Given a value v, we define the predicate Command$(\cdot)$: **Val** $\to$ {true, false} to be true for all values $v \in$ **Val** such that either v = READ(x) or v = UPDATE(x,z) for some memory value $x \in$ mVar and integer $z \in \mathbb{Z}$. Also, we define isReadRequest$(\cdot)$: **Val** $\to$ {true, false} as the predicate which is true for all the values v such that v = READ(x) for some memory value $x \in$ mVar. The predicate isWriteRequest$(\cdot)$: **Val** $\to$ {true, false} is defined to be true whenever v = UPDATE(x,z) for some memory value x and integer z.

We will also need the following partial functions. The function getVariable(v) : **Val** $\rightharpoonup$ mVar is defined only in the case Command(v) = true, and we let getVariable(READ(x)) = x, getVariable(UPDATE(x,z)) = x. That is, the (partial) function getVariable$(\cdot)$ returns the memory variable object of a command request. The function getValue(v) : **Val** $\rightharpoonup \mathbb{Z}$ is defined only in the case v = UPDATE(x,z), in which case it takes value z. Informally, the partial function getValue$(\cdot)$ returns the value associated with a write request.

With an abuse of notation, we will sometime use the functions getVariable(q) and getValue(q), where q is a queue; these are defined only in the case isEmpty(q) = false, and they correspond to getVariable(last(q)), getValue(last(q)), respectively. That is, getVariable(q) returns the variable associated with the last element of a queue of values, if such an element exists and corresponds to a command request. getValue(q) has a similar definition. In a similar way, we can extend the predicates isReadRequest$(\cdot)$ and isWriteRequest$(\cdot)$ to non-empty queues.

Finally, given a memory σ and a queue of values q such that isWriteRequest(last(q)) is defined and equal to true, then the notation $(q \mapsto \sigma)$ is used to denote the memory σ[getVariable(()q) $\mapsto$ getValue(() q)].

In the following we let q, q_1, q_2 be queues, $j = 1,2$, σ be a memory and l, l_1, l_2 be positive integers. For

$j = 1,2$, we assume a channel $c_j \in \mathbf{Ch}$. We also assume the process definitions below:

$$
\begin{array}{rcl}
I_q^{j,0} & \Leftarrow & \text{if isEmpty}(q) \text{ then } \mathbf{0} \text{ else } c_j!\langle \text{last}(q)\rangle.I^{j,0}_{\text{remLast}(q)} \\
I_q^{j,l+1} & \Leftarrow & (c?(x).\text{if Command}(x) \text{ then } I^{j,l}_{\text{push}(x,q)} \text{ else } I_q^{j,l+1}) + \\
& + & (\text{if isEmpty}(q) \text{ then } \mathbf{0} \text{ else } c_j!\langle \text{last}(q)\rangle.I^{j,l+1}_{\text{remLast}(q)})
\end{array}
$$

$$
O_q^j \Leftarrow (c_j?(x).O^j_{\text{push}(x,q)}) + (\text{if isEmpty}(q) \text{ then } \mathbf{0} \text{ else } c!\langle \text{last}(q)\rangle.O^j_{\text{remLast}(q)})
$$

$$
\begin{array}{rcl}
\text{RAM}^{\sigma}_{q_1,q_2} & \Leftarrow & (c_1?(x).\text{RAM}^{\sigma}_{\text{push}(x,q_1),q_2}) + \\
& + & (c_2?(x).\text{RAM}^{\sigma}_{q_1,\text{push}(x,q_2)}) + \\
& + & \text{if isEmpty}(q_1) \text{ then } \{\mathbf{0}\} \text{ else } \{ \\
& & \quad \text{if isReadRequest}(q_1) \text{ then } \{ \\
& & \quad\quad c_1!\langle\sigma(\text{getVariable}(q_1))\rangle.\text{RAM}^{\sigma}_{\text{remLast}(q_1),q_2} \\
& & \quad \} \text{ else } \{ \\
& & \quad\quad c_1!\langle\text{getValue}(q_1)\rangle.\text{RAM}^{(q_1\mapsto\sigma)}_{\text{remLast}(q_1),q_2} \\
& & \quad \} \\
& & \} + \\
& + & \text{if isEmpty}(q_2) \text{ then } \{\mathbf{0}\} \text{ else } \{ \\
& & \quad \text{if isReadRequest}(q_2) \text{ then } \{ \\
& & \quad\quad c_2!\langle\sigma(\text{getVariable}(q_2))\rangle.\text{RAM}^{\sigma}_{q_1,\text{remLast}(q_2)} \\
& & \quad \} \text{ else } \{ \\
& & \quad\quad c_2!\langle\text{getValue}(q_2)\rangle.\text{RAM}^{(q_2\mapsto\sigma)}_{q_1,\text{remLast}(q_2)} \\
& & \quad \} \\
& & \}
\end{array}
$$

Then the network

$$
\mathcal{M}_k^{\sigma} = \Gamma_M \rhd i_1[\![I_{\varepsilon}^{1,k}]\!] \,|\, i_2[\![I_{\varepsilon}^{2,k}]\!] \,|\, \text{mem}[\![\text{RAM}^{\sigma}_{\varepsilon,\varepsilon}]\!] \,|\, o_1[\![O^1_{\varepsilon}]\!] \,|\, o_2[\![O^2_{\varepsilon}]\!]
$$

where we recall that Γ_M is the connectivity graph depicted in Figure 5.8, is a model for Specification 5.5.1. Due to the complexity of network $\mathcal{M}_k^{\sigma}$, a detailed discussion concerning its informal behaviour is necessary. We only describe the behaviour of the nodes i_1, o_1 and mem. The behaviour of the nodes i_2, o_2 is in fact similar to that of i_1, o_1.

- Node i_1 acts as an input terminal for the client node cl_1. It waits for k command requests to be broadcast by the client node cl_1; whenever a command request is detected, it is stored it the internal buffer of node i_1, represented as a queue q.

- Whenever the internal queue of node i_1 is not empty, the latter can non-deterministically decide to forward the command request stored at the bottom of its queue to the internal node mem. This is the node where the memory σ is stored.

- Command requests forwarded by node i_1 are always broadcast along channel c_1; in contrast, values stored in the node i_2 are broadcast along channel c_2. In this way the memory node mem can infer the client node that fired a command request from the channel through which such a request was detected.

- Node mem is equipped with two different queues, one for the requests fired from client cl_1 and one for

those fired by client cl_2. At any given time, the memory node can either receive a value (if any) being sent either along channel c_1 or c_2, in which case the received value is stored in the respective queue $q_h, h = 1,2$ waiting to be processed, or it can process a command request stored in one of such queues.

- At any given time node mem can decide to process a command request (if any) fired from the client node cl_1; this can happen only in the case in which the queue q_1 with which node mem has been equipped is non-empty. In this case node mem can behave in two different ways, according to the content of the command request v it has to process:

  1. if v is a read request of the form READ(x), where $x \in$ mVar, then it will simply broadcast the value $\sigma(x)$ along channel c_1,

  2. if v is a write request of the form UPDATE(x, z), where $x \in$ mVar, $z \in \mathbb{Z}$, then node mem broadcasts the integer value z along channel c_1 and it updates the contents of the memory variable x have been updated to value z in the memory σ,

  3. in any case, the command request which has been processed is removed from the queue q_1.

- Node o_1 acts as an output terminal for the client node cl_1; it collects the values broadcast by the memory node mem along channel c_1, storing them in a queue. At any given time, if such a queue is non-empty, it can remove its last element and broadcast it along channel c, thus providing a reply to a former request fired from the client node cl_1.

Note that request processing in node mem is an atomic operation, meaning that such an activity requires a single strong extensional transition. For example, processing an update request requires a single intensional broadcast transition, which induces a strong extensional τ-transition; node mem broadcast a value along either channel c_1 or c_2, after which the code running at node mem in the virtual shared memory model consists of the process RAM in which the memory σ has been updated, and the command request which has been processed has been removed from the queue where it was stored.

Another important point in the design of $\mathcal{M}_k^\sigma$ is that we have used queues to store values in nodes; this ensures that command requests performed by node cl_1 are processed in the same order they have been broadcast; the same applies for the client node cl_2. However, note that network $\mathcal{M}_k^\sigma$ has no synchronisation mechanism to coordinate the requests of clients cl_1, cl_2; that is, it is possible that a command request v_1 is fired by cl_1 before a second command request v_2 issued from cl_2, but the latter is processed by node mem before the former.

Now the reader should have a clear idea of the intuitive behaviour of network $\mathcal{M}_k^\sigma$. Let us turn our attention at a possible implementation for it. The main idea is that of using different nodes to store a memory σ, in a way such that different memory variables are stored at different locations (that is, replicas of memory variables are not allowed).

The network model $\mathcal{N}_f^k$ we propose is the network $\Gamma_N \triangleright N_k^f$, where Γ_N is the connectivity graph depicted in Figure 5.9 and the system term N_k^f is presented shortly; note that Γ_N is parametric in a number of nodes $\mathrm{mem}_1, \cdots, \mathrm{mem}_j$ for some integer $j \geq 1$. In the system term N_k^f, f is a mapping from the set $\{1, \cdots, j\}$ to partial memories, that is $f(h) : \mathrm{mVar} \rightharpoonup \mathbb{Z}$ for any $h = 1, \cdots, j$.

Before defining the system term N_k^f, we need some process definitions, for which we assume two channels lock_1 and lock_2. In the following, we also let q_1, q_2 be queues of values, h be a positive integer ranging over $1, \cdots, j$, $l = 1,2$ and $\sigma : \mathrm{mVar} \rightharpoonup \mathbb{Z}$ be a partial memory. Further, for any such partial memory, we let

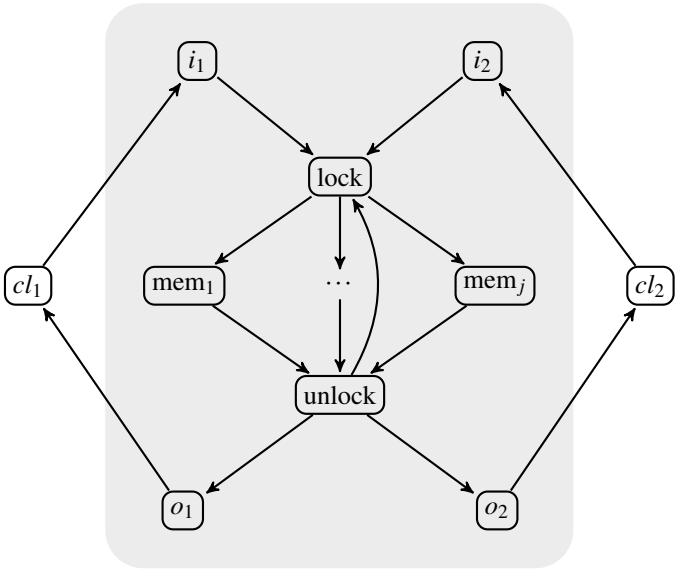


Figure 5.9: The connectivity graph for the implementation of Virtual Shared Memory

$\text{Dom}(\sigma) = \{x \in \text{mVar} \mid \sigma(x) \text{ is defined } \}$ be its domain.

$$
\begin{aligned}
\text{WAIT}^{0,0}_{q_1,q_2} \quad &\Leftarrow \quad (c_1?(x).\text{WAIT}^{0,0}_{\text{push}(x,q_1),q_2}) + \\
&+ \quad c_2?(x).\text{WAIT}^{0,0}_{q_1,\text{push}(x,q_2)} + \\
&+ \quad (\text{if isEmpty}(q_1) \text{ then } \mathbf{0} \text{ else } c_1!\langle \text{last}(q_1)\rangle.\text{WAIT}^{1,0}_{\text{remLast}(q_1),q_2}) + \\
&+ \quad (\text{if isEmpty}(q_2) \text{ then } \mathbf{0} \text{ else } c_2!\langle \text{last}(q_2)\rangle.\text{WAIT}^{0,1}_{q_1,\text{remLast}(q_2)}) \\
\text{WAIT}^{1,0}_{q_1,q_2} \quad &\Leftarrow \quad (c_1?(x).\text{WAIT}^{1,0}_{\text{push}(x,q_1),q_2}) + \\
&+ \quad c_2?(x).\text{WAIT}^{1,0}_{q_1,\text{push}(x,q_2)} + \\
&+ \quad \text{lock}_1?(x).\text{WAIT}^{0,0}_{q_1,q_2} + \\
&+ \quad (\text{if isEmpty}(q_2) \text{ then } \mathbf{0} \text{ else } c_2!\langle \text{last}(q_2)\rangle.\text{WAIT}^{1,1}_{q_1,\text{remLast}(q_2)}) \\
\text{WAIT}^{0,1}_{q_1,q_2} \quad &\Leftarrow \quad (c_1?(x).\text{WAIT}^{0,1}_{\text{push}(x,q_1),q_2}) + \\
&+ \quad c_2?(x).\text{WAIT}^{0,1}_{q_1,\text{push}(x,q_2)} + \\
&+ \quad (\text{if isEmpty}(q_1) \text{ then } \mathbf{0} \text{ else } c_1!\langle \text{last}(q_1)\rangle.\text{WAIT}^{1,1}_{\text{remLast}(q_1),q_2}) + \\
&+ \quad \text{lock}_2?(x).\text{WAIT}^{0,0}_{q_1,q_2} \\
\text{WAIT}^{1,1}_{q_1,q_2} \quad &\Leftarrow \quad (c_1?(x).\text{WAIT}^{1,1}_{\text{push}(x,q_1),q_2}) + \\
&+ \quad c_2?(x).\text{WAIT}^{1,1}_{q_1,\text{push}(x,q_2)} + \\
&+ \quad \text{lock}_1?(x).\text{WAIT}^{0,1}_{q_1,q_2} + \\
&+ \quad \text{lock}_2?(x).\text{WAIT}^{1,0}_{q_1,q_2}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{PRAM}^{\sigma}_{q_1,q_2} \Leftarrow\ & (c_1?(x).\mathrm{PRAM}^{\sigma}_{\mathrm{push}(x,q_1),q_2}) + \\
+\ & (c_2?(x).\mathrm{PRAM}^{\sigma}_{q_1,\mathrm{push}(x,q_2)}) + \\
+\ & \text{if } \mathrm{isEmpty}(q_1) \text{ then } \mathbf{0} \text{ else } \{ \\
& \quad \text{if } \mathrm{getVariable}(q_1) \in \mathrm{Dom}(\sigma) \text{ then } \{ \\
& \quad\quad \text{if } \mathrm{isReadRequest}(q_1) \text{ then } \{ \\
& \quad\quad\quad c_1!\langle \sigma(\mathrm{getVariable}(()q_1))\rangle.\mathrm{PRAM}^{\sigma}_{\mathrm{remLast}(q_1),q_2} \\
& \quad\quad \} \text{ else } \{ \\
& \quad\quad\quad c_1!\langle \mathrm{getValue}(q_1)\rangle.\mathrm{PRAM}^{(q\mapsto\sigma)}_{\mathrm{remLast}(q_1),q_2} \\
& \quad\quad \} \\
& \quad \} \text{ else } \mathbf{0} \\
& \} + \\
+\ & \text{if } \mathrm{isEmpty}(q_2) \text{ then } \mathbf{0} \text{ else } \{ \\
& \quad \text{if } \mathrm{getVariable}(q_2) \in \mathrm{Dom}(\sigma) \text{ then } \{ \\
& \quad\quad \text{if } \mathrm{isReadRequest}(q_2) \text{ then } \{ \\
& \quad\quad\quad c_1!\langle \sigma(\mathrm{getVariable}(()q_2))\rangle.\mathrm{PRAM}^{\sigma}_{q_1,\mathrm{remLast}(q_2)} \\
& \quad\quad \} \text{ else } \{ \\
& \quad\quad\quad c_1!\langle \mathrm{getValue}(q_2)\rangle.\mathrm{PRAM}^{(q\mapsto\sigma)}_{q_1,\mathrm{remLast}(q_2)} \\
& \quad\quad \} \\
& \quad \} \text{ else } \mathbf{0} \\
& \}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{SIGNAL}_{q_1,q_2} \Leftarrow\ & (c_1?(x).\mathrm{SIGNAL}_{\mathrm{push}(x,q_1),q_2}) + \\
+\ & (c_2?(x).\mathrm{SIGNAL}_{q_1,\mathrm{push}(x,q_2)}) + \\
+\ & \text{if } \mathrm{isEmpty}(q_1) \text{ then } \mathbf{0} \text{ else } \mathrm{unlock}_1!\langle \mathrm{last}(q_1)\rangle.\mathrm{SIGNAL}_{\mathrm{remLast}(q_1),q_2} + \\
+\ & \text{if } \mathrm{isEmpty}(q_2) \text{ then } \mathbf{0} \text{ else } \mathrm{unlock}_2!\langle \mathrm{last}(q_2)\rangle.\mathrm{SIGNAL}_{q_1,\mathrm{remLast}(q_2)}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{Out}^{l}_{q} \Leftarrow\ & \mathrm{unlock}_l?(x).\mathrm{Out}^{l}_{\mathrm{push}(x,q)} + \\
+\ & \text{if } \mathrm{isEmpty}(q) \text{ then } \mathbf{0} \text{ else } c!\langle \mathrm{last}(q)\rangle.\mathrm{Out}^{l}_{\mathrm{remLast}(q)}
\end{aligned}
$$

Let f be a mapping from the index set $\{1,\cdots,j\}$ to partial memories such that whenever $h \neq h'$, $h,h' = 1,\cdots,j$ then $\mathrm{Dom}(f(h)) \cap \mathrm{Dom}(f(h')) = \emptyset$ and $\bigcup_{h=1}^{j} \mathrm{Dom}(f(h)) = \mathbf{Var}$; then N^{f}_{k} is defined as the system term

$$
i_1[\![I^{1,k}_{\varepsilon}]\!] \mid i_2[\![I^{2,k}_{\varepsilon}]\!] \mid o_1[\![\mathrm{out}^{1}_{\varepsilon}]\!] \mid o_2[\![\mathrm{Out}^{2}_{\varepsilon}]\!] \mid \mathrm{lock}[\![\mathrm{WAIT}^{0,0}_{\varepsilon,\varepsilon}]\!] \mid \mathrm{unlock}[\![\mathrm{SIGNAL}_{\varepsilon,\varepsilon}]\!] \mid \prod_{h=1}^{j} \mathrm{mem}_h[\![\mathrm{PRAM}^{f(h)}_{\varepsilon,\varepsilon}]\!]
$$

As usual, we give an informal description of the network $\mathcal{N}^{f}_{k}$. We only describe how requests performed by the client node cl_1 are processed, for the behaviour of the network in the case cl_2 broadcasts a command request is similar.

- As in the model $\mathcal{M}^{\sigma}_{k}$, node i_1 acts as an input terminal for the client node cl_1; it stores the first k command requests in its own queue, and whenever such a queue is not empty it forwards the least recent request to another node, lock, along channel c_1,

- Whenever node lock receives a message along channel c_1, it stores it in its own queue used to handle the requests performed by client c_1. The way in which requests stored in this queue are forwarded to other nodes, however, is slightly different from previous cases, as we have equipped node lock with a mutual exclusion mechanism.

  In fact, node lock is equipped with a bit flag b_1, which is used to check whether the node is allowed to broadcast a command request along channel c_1; if such a bit is set to 0 then node lock is free to broadcast the last value (if any) stored in q_1 along channel c_1. In this case the node updates the flag b_1 to the value 1, meaning that the nodes $\mathrm{mem}_1, \cdots, \mathrm{mem}_j$ are already processing a request broadcast by cl_1.

  If the flag b_1 is set to 1, then node lock waits the request which is being processed to be completed before it can broadcast another request (originally broadcast from c_1 to the nodes $\mathrm{mem}_1, \cdots, \mathrm{mem}_j$. Node lock is acknowledged that such a request has been completed by receiving an input along a channel lock_1, after which it restores the flag b_1 to value 0; as we will see, it is the duty of another node, unlock, to signal a value along channel lock_1, thus making node lock restore its flag to value 0.

  In the initial configuration $\mathcal{N}_k^f$ the bit b_1 of node lock is set to 0; that is, node lock can forward values along channel c_1. Such values are detected by j different nodes, $\mathrm{mem}_1, \cdots, \mathrm{mem}_j$.

- When a command request along channel c_1 is received by the nodes $\mathrm{mem}_1, \cdots, \mathrm{mem}_j$, each of these nodes will proceed to process the received request. However, exactly one of such nodes will be able to process the request, while the others will just ignore it.

  This is because each node mem_h, $h = 1, \cdots, j$ is equipped with a partial memory σ_h, and a command request v received at node mem_h is processed only in the case the memory variable object of the request, $\mathrm{getVariable}(v)$, is in the domain of σ_h. The constraint we imposed that $\bigcup_{h=1}^{j} \mathrm{Dom}(\sigma_h) = \mathrm{mVar}$ ensures that at least one of such nodes will process the received command request, while the constraint that if $h \neq h'$ then $\mathrm{Dom}(\sigma_h) \cap \mathrm{Dom}(\sigma_{h'}) = \emptyset$ states that such a command request will never be processed by more than one of the nodes in $\{\mathrm{mem}_1, \cdots, \mathrm{mem}_j\}$.

  When the only node which can process the request has accomplished its task, it forwards the reply to such a command request along channel c_1. The broadcast value will be received by node unlock.

- The role of node unlock is that of forwarding the values received along channel c_1 from one of the nodes $\mathrm{mem}_1, \cdots, \mathrm{mem}_j$. Again, this node uses a queue to store the received values and forward them in the same order they have been received. Values received along channel c_1 will be forwarded along channel lock_1. Both nodes o_1 and lock will detect this value; we have already discussed that channel lock_1 is used at the latter node, together with a flag b_1, to implement a semaphore for the requests to be processed by nodes $\mathrm{mem}_1, \cdots \mathrm{mem}_j$. By broadcasting the value along channel lock_1, node unlock notifies to node lock that $\mathrm{mem}_1, \cdots, \mathrm{mem}_j$ are ready to process another command request.

- Node o_1 acts at an output terminal for the client node cl_1. It simply forwards values received along channel lock_1 to the client cl_1. As usual, we equipped this node with a queue in order to keep track of the values that it has received and that have not yet been forwarded. Values are broadcast by node o_1 along channel c.

A formal analysis of the behaviour of the network $\mathcal{N}_k^f$ would be extremely technical and complicated, and is therefore omitted. We just state the result in which we are interested.

Theorem 5.5.2. Let $k > 0$, $\sigma : \mathrm{mVar} \to \mathbb{Z}$ and f be a mapping from $\{1, \cdots, j\}$ to partial memories. Suppose that for any $h, h' = 1, \cdots, j$ such that $h \neq h'$ then $\mathrm{Dom}(f(h) \neq \mathrm{Dom}(f(h'))$ and let $\bigcup_{h=1}^{j} \mathrm{Dom}(f(h)) = \mathrm{mVar}$. Also, assume that for any $x \in \mathrm{mVar}$ there exists an index $h = 1, \cdots, j$ such that $f(h)(x)$ is defined and equal to $\sigma(x)$.

Then $\mathcal{M}_k^\sigma \simeq \mathcal{N}_k^f$.

Part II

Probabilistic Wireless Networks

Chapter 6

Probabilistic Wireless Networks

In this Chapter we augment the extend calculus developed in Part I by allowing nodes to behave probabilistically; intuitively, a probabilistic process in this variant corresponds to a *probability distribution* over processes as described in Figure 2.1 at page 22. More formally, a probabilistic process has the form $\bigoplus_{i\in I} p_i \cdot S_i$, where each S_i is a process and $P = \sum_{i\in I} p_i = 1$, meaning that the probabilistic process P behaves as process S_i with probability p_i. The need for probabilistic behaviour arises for it is often the case that probabilistic protocols are used in wireless systems to improve the performance of a network; see for example [59, 14, 54].

In this Chapter we address the same topics that we have already illustrated for the non-probabilistic setting in chapters 2 and 3. Specifically, we define a reduction semantics and a labelled transition semantics for networks, proving that they are consistent with each other. Further, we define a theory of composition for networks; in practice the same theory we developed in Chapter 3 applies, for our approach to compositionality is based on network topologies, rather than on the code run by processes. We also extend the testing preorders $\sqsubseteq_{\text{may}}$ and $\sqsubseteq_{\text{must}}$ to the probabilistic case, in the same way it has been done for the probabilistic version of *CSP* (pCSP) in [20, 19]; while the definition of the probabilistic testing preorders for probabilistic networks follows well-trodden paths, we decided to include them in this thesis since them main topic of Chapter 7 is that of exhibiting sound proof techniques for such preorders.

Defining a reduction semantics in our probabilistic calculus is not a difficult task; as we will see, the rules we define are an immediate extension of those defined in Chapter 2, Table 2.3 at Page (2.3). The same applies for the labelled transition semantics. Note that in Part I we used LTSs as a formal model for networks; each network generates an LTS according to the rules of the extensional semantics; since we wish to model networks which endow probabilistic behaviour, this model has to be extended to handle probability distributions. This gives rise to the notion of *probabilistic Labelled Transition System* (**pLTS**) [20, 19, 35], which is a variant of LTSs where the transition relation is defined between states and probability distribution of states.

Since in pLTSs the domain and the image of transitions do not coincide, it is not possible to define directly sequences of actions; instead, using the approach used in [20, 19], we lift the transition relations to describe how a distribution of states can perform a transition in another distribution. This enables us to reason with sequences of transitions as well as to infer the behaviour of a network in the long run.

Let us turn our attention to testing theories in our probabilistic calculus of networks. Since the code placed at nodes endows probabilistic behaviour, the qualitative property of being successful for computations is replaced with a quantitative one, in which we establish that a computation is successful with some probability. Due to the non-deterministic behaviour of networks, an experiment of the form $\mathcal{M} \between \mathcal{T}$ is then equipped with a set of success outcomes, each of which corresponds to the success probability associated with one of the computation of the network.

The testing preorders are then defined by comparing the set of probabilities with which they pass a test. If $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ and $\mathcal{M} \between \mathcal{T}$ has a computation whose success probability is p, then we require $\mathcal{N} \between \mathcal{T}$ to have a computation whose success probability is greater of p. Conversely, if whenever $\mathcal{N} \between \mathcal{T}$ has a computation whose success probability is p there exists a computation of $\mathcal{M} \between \mathcal{T}$ whose success probability is $p' \leq p$ then

we say that $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$.

The rest of this Chapter is organised as follows. Section 6.1 contains the notions that are needed to model probabilistic wireless networks.

In Section 6.2 we introduce our probabilistic calculus of networks and we provide a simple example of probabilistic networks. In the same Section we also define the reduction semantics for networks, then we turn our attention to the labelled transition semantics and we prove the Harmony Theorem for probabilistic networks. We remark that this result holds only for transitions defined from networks to distribution of networks, but it is not true in the case of the lifted transitions.

In Section 6.3 we show how the success probability of a computation in an experiment of the form $\mathcal{M} \between \mathcal{T}$ is computed, and we define the probabilistic extensions of the may and must testing preorders. We exhibit an example that shows which are related in the non-probabilistic case, with respect to the the may-testing preorder, are not related anymore in the probabilistic counterpart.

6.1 Background

In this Section we will summarise the mathematical concepts, taken from [19], that will be needed throughout the paper. First we will introduce some basic concepts from probability theory; then we will show how these can be used to model concurrent systems which exhibit both probabilistic and non-deterministic behaviour.

Let S be a set; a function $\Delta : S \to [0,1]$ is called a (probability) sub-distribution over S if $\sum_{s \in S} \Delta(s) \leq 1$. This quantity, $\sum_{s \in S} \Delta(s)$, is called the mass of the sub-distribution, denoted as $|\Delta|$. If $|\Delta| = 1$, then we say that Δ is a (full) distribution. The support of a distribution Δ, denoted $\lceil \Delta \rceil$, is the subset of S consisting of all those elements which contribute to its mass, namely $\lceil \Delta \rceil = \{s \in S \mid \Delta(s) > 0\}$.

For each $s \in S$, the point distribution $\overline{s}$ is defined to be the distribution which takes value 1 at s, and 0 elsewhere. The set of sub-distributions and distributions over a set S are denoted by $\mathcal{D}_{sub}(S)$ and $\mathcal{D}(S)$, respectively.

Given a family of sub-distributions $\{\Delta_k \mid k \in K\}$, $\sum_{k \in K} \Delta_k$ is the partial real-valued function in $S \to \mathbb{R}$ defined by $(\sum_{k \in K} \Delta_k)(s) := \sum_{k \in K} \Delta_k(s)$. This is a partial operation because for a given $s \in S$ this sum might not exist; it is also a partial operation on sub-distributions because even if the sum does exist it may be greater than 1.

Similarly, if $p \leq 1$ and Δ is a sub-distribution , then $p \cdot \Delta$ is the sub-distribution over S such that

$$(p \cdot \Delta)(s) = p \cdot \Delta(s).$$

It is not difficult to show that if $\{p_k\}_{k \in K}$ is a sequence of positive real numbers such that $\sum_{k \in K} p_k \leq 1$, and $\{\Delta_k\}_{k \in K}$ is a family of sub-distributions over a set S, then $\sum_{i=1}^{n} p_i \cdot \Delta_i$ always defines a sub-distribution over S.

Finally, if $f : X \to Y$ and Δ is a sub-distribution over X then we use $f(\Delta)$ to be the sub-distribution over Y defined by:

$$f(\Delta)(y) = \sum_{x \in X} \{\Delta(x) \mid f(x) = y\}. \tag{6.1}$$

This definition can be generalised to two arguments functions; if $f : X_1 \times X_2 \to Y$ is a function, and Δ, Θ are two sub-distributions respectively over X_1 and X_2, then $f(\Delta, \Theta)$ denotes the sub-distribution over Y defined as

$$f(\Delta, \Theta)(y) = \sum_{x_1 \in X_1, x_2 \in X_2} \{\Delta(x_1) \cdot \Theta(x_2) \mid f(x_1, x_2) = y\}. \tag{6.2}$$

Now we turn our attention to probabilistic concurrent systems. The formal model we use to represent them is a generalisation to a probabilistic setting of Labelled Transition Systems (LTSs) [50].

Definition 6.1.1. A *probabilistic labelled transition system* (pLTS) is a 4-tuple $\langle S, \mathsf{Act}_\tau, \to, \omega \rangle$, where

(i) S is a set of states,

(ii) Act_τ is a set of transition labels with a distinguished label τ,

(iii) the relation $\rightarrow$ is a subset of $S \times \mathsf{Act}_\tau \times \mathcal{D}(S)$,

(iv) $\omega : S \mapsto \{ \text{ true }, \text{ false } \}$ is a (success) predicate over the states S.

As usual, we will write $s \xrightarrow{\mu} \Delta$ in lieu of $(s, \alpha, \Delta) \in \longrightarrow$. □ □

Before discussing pLTSs, some definitions first: a pLTS whose state space is finite is said to be finite state; further, we say that a pLTS $\langle S, \mathsf{Act}_\tau, \rightarrow, \omega \rangle$ is finite branching if, for every $s \in S$, the set $\{ \Delta \mid s \xrightarrow{\mu} \Delta \text{ for some } \mu \in \mathsf{Act}_\tau \}$ is finite. Finally, a finitary pLTS is one which is both finite state and finite branching.

We have included in the definition of a pLTS a success predicate ω over states, which will be used when testing processes. Apart from this, the only difference between LTSs and pLTSs is given by the definition of the transition relation; in the latter this is defined to be a relation (parametric in some action μ) between states and distribution of states, thus capturing the concept of probabilistic behaviour.

However, this modification introduces some difficulties when sequences of transitions performed by a given pLTS have to be considered, as the domain and the image of the transition relation do not coincide. To avoid this problem, we will focus only on distributions of states by defining transitions between distributions of states. The following Definition serves to this purpose:

Definition 6.1.2 (Lifted Relations)**.** *Let $\mathcal{R} \subseteq S \times \mathcal{D}_{sub}(S)$ be a relation from states to sub-distributions. Then $\overline{R} \subseteq \mathcal{D}_{sub}(S) \times \mathcal{D}_{sub}(S)$ is the smallest relation which satisfies*

- *$s \mathcal{R} \Delta$ implies $\overline{s} \, \overline{\mathcal{R}} \, \Delta$*
- *If I is a finite index set and $\Delta_i \, \overline{\mathcal{R}} \, \Theta_i$ for each $i \in I$ then $(\sum_{i \in I} p_i \cdot \Delta_i) \; \overline{\mathcal{R}} \; (\sum_{i \in I} p_i \cdot \Theta_i)$ whenever $\sum_{i \in I} p_i \leq 1$.* □

Lifting of relations can also be defined for probability distributions, by simply requiring $\sum_{i \in I} p_i = 1$ in the last constraint of the definition above.

Example 6.1.3 (Lifted Relations)**.** Let us illustrate an example that shows how the lifting of a relation from states to distributions works. Imagine that a gambler has the chance to play at two different lotteries, but his amount of money allows him to buy only one ticket. In the first lottery, which we call A, the winning amount of the gambler is chosen probabilistically. With probability 0.6 the gambler does not win any money, with probability 0.2 he wins 2 Euros, with probability 0.1 he wins 5 Euros and with probability 0.1 he wins 10 euros. In the second lottery, which we call B, the gambler loses with probability 0.8, while he wins 10 euros with probability 0.2.

We can formalise this situation by defining an outcome relation $\mathcal{R}$ which associates to each lottery the probability distribution of the winning outcomes. That is, we have $A \mathcal{R} \Theta_A$, $B \mathcal{R} \Theta_B$, where $\Theta_A = 0.6 \cdot \overline{0} + 0.2 \cdot \overline{2} + 0.1 \cdot \overline{5} + 0.1 \cdot \overline{1}$ and $\Theta_B = 0.8 \cdot \overline{0} + 0.2 \cdot \overline{10}$. Here note that $\overline{n}, n \in \mathbb{N}$ is a probability distribution over natural numbers, which takes value n with probability 1.

Suppose now that the gambler, in order to choose the lottery that he wants to play, flips a fair coin. In the case the outcome is head, he plays lottery A, otherwise he plays lottery B. Given this information, we can represent the gambler as a probability distribution Δ between lotteries, specifically $\Delta = 0.5 \cdot \overline{A} + 0.5 \cdot \overline{B}$. Again, $\overline{A}, \overline{B}$ are two pointed probability distributions over the set of lotteries.

In order to calculate the overall probability distribution of the winning outcomes for the gambler, we can lift the outcome relation $\mathcal{R}$ to $\overline{\mathcal{R}}$, then we determine the distribution Θ for which $\Delta \overline{\mathcal{R}} \Theta$. Using definition 6.1.2 [1] we obtain that $\Delta \overline{\mathcal{R}} 0.5 \cdot \Theta_A + 0.5 \cdot \Theta_B$; by performing the required substitutions, and solving the provided calculations, it is easy to note that

$$0.5 \cdot \Theta_A + 0.5 \cdot \Theta_B = 0.7 \cdot \overline{0} + 0.1 \cdot \overline{2} + 0.05 \cdot \overline{5} + 0.15 \cdot \overline{10}$$

[1] Note that here we committed a slight abuse of notation, since we defined lifting only for relations whose domain and image were the same set. However, Definition 6.1.2 can be trivially extended to more general relations.

□

In a pLTS $\langle S, \mathsf{Act}_\tau, \rightarrow, \omega \rangle$, each transition relation $\xrightarrow{\mu} \subseteq S \times \mathcal{D}(S)$ can be lifted to $(\overline{\xrightarrow{\mu}}) \subseteq \mathcal{D}(S) \times \mathcal{D}(S)$. With an abuse of notation, the latter will still be denoted as $\xrightarrow{\mu}$.

Lifted transition relations allow us to reason about the behaviour of pLTSs in terms of sequences of transitions; here we are mainly interested in the behaviour of a pLTS in the long run; that is, given a pLTS $\langle S, \mathsf{Act}_\tau, \rightarrow, \omega \rangle$ and a distribution $\Delta \subseteq \mathcal{D}(S)$, we are interested in distributions $\Theta \subseteq \mathcal{D}(S)$ which can be reached by Δ after an indefinite number of transitions.

For the moment we will focus only on internal actions of a pLTS, in which case the behaviour of a pLTS in the long run is captured by the concept of hyper-derivation:

Definition 6.1.4. [Hyper-derivations] In a pLTS a hyper-derivation consists of a collection of sub-distributions $\Delta, \Delta_k^{\rightarrow}, \Delta_k^{\times}$, for $k \geq 0$, with the following properties:

$$\begin{array}{lcl} \Delta & = & \Delta_0^{\rightarrow} + \Delta_0^{\times} \\ \Delta_0^{\rightarrow} & \xrightarrow{\tau} & \Delta_1^{\rightarrow} + \Delta_1^{\times} \\ & \vdots & \\ \Delta_k^{\rightarrow} & \xrightarrow{\tau} & \Delta_{k+1}^{\rightarrow} + \Delta_{k+1}^{\times} \\ & \vdots & \end{array}$$

If $\omega(s) = \text{ false }$ for each $s \in \lceil \Delta_k^{\rightarrow} \rceil$ and $k \geq 0$ we call $\Delta' = \sum_{k=0}^{\infty} \Delta_k^{\times}$ a *hyper-derivative* of Δ, and write $\Delta \Longrightarrow \Delta'$. □ □

Hyper-derivations can be viewed as the probabilistic counterpart of the weak $\stackrel{\tau}{\Longrightarrow}$ action in LTSs; see [19] for a detailed discussion. Intuitively speaking, they represent fragments of computations obtained by performing only internal actions. The last constraint in Definition 6.1.4 is needed since we introduced a success predicate in our model; we require that a computation cannot proceed in the case that a state s such that $\omega(s) = \text{true}$ has been reached; this is for we are only interested in detecting if such states can be reached in a computation. States in which the predicate $\omega(\cdot)$ is true are called ω-successful.

Further, we are mainly interested in maximal computations of distributions. That is, we require a computation to proceed as long as some internal activity can be performed. To this end, we say that $\Delta \Longrightarrow\!\!\!\!\!\rightarrow \Delta'$ if

- $\Delta \Longrightarrow \Delta'$,
- for every $s \in \lceil \Delta_k^{\times} \rceil$, $s \xrightarrow{\tau}$ implies $\omega(s) = \text{true}$. □

This is a mild generalisation of the notion of *extreme derivative* from [19]. Note that the last constraint models exactly the requirement of performing some internal activity whenever it is possible; In other words extreme derivatives correspond to a probabilistic version of maximal computations.

Theorem 6.1.5. In an arbitrary pLTS

(1) $\Longrightarrow$ is reflexive and transitive

(2) if $\Delta \Longrightarrow \Delta'$ and $\Delta' \Longrightarrow\!\!\!\!\!\rightarrow \Delta''$, then $\Delta \Longrightarrow\!\!\!\!\!\rightarrow \Delta''$; this is a direct consequence of the previous statement, and the definition of extreme derivatives

(3) suppose $\Delta = \sum_{i \in I} p_i \cdot \Delta_i$, where I is an index set and $\sum_{i \in I} p_i \leq 1$. If for any $i \in I, \Delta_i \Longrightarrow \Theta_i$ for some Θ_i, then $\Delta \Longrightarrow \Theta$, where $\Theta = \sum_{i \in I} p_i \cdot \Theta_i$.

(4) for all distributions Δ, there exists a sub-distribution Θ such that $\Delta \Longrightarrow\!\!\!\!\!\rightarrow \Theta$

Proof. See [19] for detailed proofs. □

M, N	::=		**Systems**
		$n[\![S]\!]$	Nodes
		$M \mid N$	Composition
		$\mathbf{0}$	Identity
P, Q	::=		(probabilistic) Processes
		S	
		$P\ {}_p\!\oplus Q$	probabilistic choice
S, T	::=		States
		$c!\langle e\rangle.P$	broadcast
		$c?(x).P$	receive
		$\omega.\mathbf{0}$	success
		$S + T$	choice
		if b then S else T	branch
		$\tau.P$	pre-emption
		$A(\tilde{x})$	definitions
		$\mathbf{0}$	terminate

Figure 6.1: Syntax

6.2 Networks and Their Computations

In this Section we define our probabilistic calculus for networks. The main idea is that of introducing a probabilistic choice operator $P\ {}_p\!\oplus Q$; intuitively, this (probabilistic) process behaves as S with probability p, while it behaves as R with probability $1-p$.

The language for system terms, ranged over by M, N is given in Figure 6.1. Basically a system consists of a collection of named nodes at each of which there is some running code. The syntax for this code is obtained by the one illustrated in Part I, augmented by a probabilistic choice; code descriptions have the usual constructs for channel based communication, with input $c?(x).p$ being the unique binder. We assume that definitions are of the form $A(x) \Leftarrow S$, that is a process definition is allowed only for state-based processes.

Note that the non-deterministic sum and the branching construct only use state based processes, rather than probabilistic processes. This choice has been done for technical reasons, which we will explain when presenting the intensional semantics of the calculus.

A probabilistic network is a pair $\Gamma \triangleright M$, where M is a term as in Figure 6.1 and Γ is a connectivity graph; recall that connectivity graphs have already been introduced in Part I. The notation and the definitions we have introduced for non-probabilistic networks is extended to its probabilistic extension we have defined. In particular, the definition of the set nodes(M), $\mathsf{Input}(\mathcal{M})$, $\mathsf{Output}(\mathcal{M})$ and $\mathsf{Int}(\mathcal{M})$, as well as the definition of well-defined and composed networks, are still valid in the probabilistic framework.

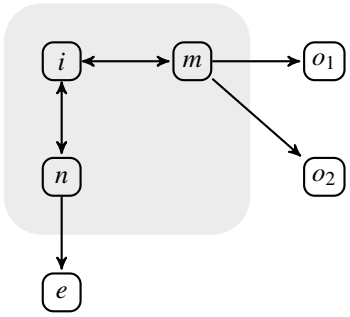
Let us illustrate the behaviour of probabilistic networks with an Example.

Example 6.2.1. Consider $\mathcal{M}$ described in Figure 6.2. There are six nodes, three occupied by code n, i and m, and three in the interface $\mathsf{Int}(\mathcal{M})$, e, o_1 and o_2. Suppose the code at nodes are given by

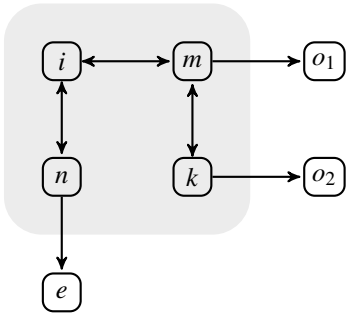
$$A_n \Leftarrow c?(x).d!\langle x\rangle.\mathbf{0} \qquad A_i \Leftarrow d?(x).d!\langle f(x)\rangle.\mathbf{0} \qquad A_m \Leftarrow d?(x).(d!\langle x\rangle.\mathbf{0}\ {}_{0.8}\!\oplus \mathbf{0})$$

Then $\mathcal{M}$ can receive input from node e at its interface along the channel c; this is passed on to the internal node i using channel d, where it is transformed in some way, described by the function f, and then forwarded to node m, where 80% of the time it is broadcast to the external nodes o_1 and o_2. The remainder of the time the message is lost.

The network $\mathcal{N}$ has the same interface as $\mathcal{M}$, but has an extra internal node k connected to o_2, and m is only



$\mathcal{M} = \Gamma_M \rhd n[\![A_n]\!] \mid i[\![A_i]\!] \mid m[\![A_m]\!]$



$\mathcal{N} = \Gamma_N \rhd n[\![A_n]\!] \mid i[\![A_i]\!] \mid m[\![A_k]\!] \mid k[\![A_m]\!]$

Figure 6.2: Example networks

connected to one interface node o_1 and the internal node k. The nodes i and n have the same code running as in $\mathcal{M}$, while nodes m and k will run the code

$$A_k \Leftarrow d?(x).(d!\langle x\rangle.\mathbf{0}_{0.9}\oplus \mathbf{0})$$

Intuitively, the behaviour of $\mathcal{N}$ is more complex than that of $\mathcal{M}$; indeed, there is the possibility for a computation of $\mathcal{N}$ to deliver a value only to one between the external nodes o_1 and o_2, while this is not possible in $\mathcal{M}$. However, 81% of the times this message will be delivered to both these nodes, and thus it is more reliable than $\mathcal{M}$. Suppose now that we change the code at the intermediate code m in $\mathcal{M}$,

$$\mathcal{M}_1 = \Gamma_M \rhd \ldots \mid m[\![B_m]\!] \qquad \text{where } B_m \Leftarrow d?(x).(\tau.(d!\langle x\rangle.\mathbf{0}_{0.5}\oplus \mathbf{0}) + \tau.d!\langle x\rangle.\mathbf{0})$$

In $\mathcal{M}_1$ the behaviour at the node m is non-deterministic; it may act like a perfect forwarder, or one which is only 50% reliable. Optimistically it could be more reliable than $\mathcal{M}$, or pessimistically it could be less reliable than the latter. Further, there is no possibility for the network $\mathcal{M}_1$ to forward the message to only one of the external nodes o_1, o_2, so that its behaviour is somewhat less complex than that of $\mathcal{N}$.

As a further variation let $\mathcal{M}_2$ be the result of replacing the code at m with

$$\begin{aligned} C_m &\Leftarrow d?(x).D \\ D &\Leftarrow \tau.(d!\langle x\rangle.\mathbf{0}_{0.5}\oplus \tau.D) \end{aligned}$$

Here the behaviour is once more deterministic, with the probability that the message will be eventually transmitted successfully through node k approaching 1 in the limit. Thus, this network is as reliable as $\mathcal{M}_1$, when the latter is viewed optimistically. □

We now turn our attention to the reduction semantics for networks; following [20, 19], processes are interpreted as probability distributions of states; such an interpretation is encoded by the function $\mathbb{P}(\cdot)$ defined below:

$$\begin{aligned} \mathbb{P}(S) &= \overline{S} \\ \mathbb{P}(P_1 {}_p\!\oplus P_2) &= p\cdot\mathbb{P}(P_1) + (1 - p\cdot\mathbb{P}(P_2)). \end{aligned}$$

We can lift the interpretation function to system terms; in fact, it suffices to define $\mathbb{P}(n[\![P]\!]) = n[\![\mathbb{P}(P)]\!]$ and $\mathbb{P}(M_1 \mid M_2) = \mathbb{P}(M_1) \mid \mathbb{P}(M_2)$. Here the operators $n[\![\cdot]\!]$ for distribution of processes and the operator $(\cdot \mid \cdot)$ for pairs of distributions of system terms are instances of Equation 6.1.

The Rules of the reduction semantics are defined in Figure 6.3; these are the probabilistic variant of those defined in Section 2.3 at Page 26; we used the notation $\Gamma \rhd M \twoheadrightarrow \Delta$ as a shortcut for $\Gamma \rhd M \twoheadrightarrow \Gamma \rhd \Delta$, where the

(R-BCAST)
$$\frac{[\![e]\!] = v \quad \forall i \in I.\Gamma \vdash m \to n_i \quad \neg \mathrm{rcv}(M,c) \quad \forall n \in \mathrm{nodes}(N).\Gamma \vdash m \nrightarrow n}{m[\![c!\langle e\rangle.P + Q]\!] \mid \prod_{i\in I} n_i[\![(c?(x).P_i) + Q_i]\!] \mid M \mid N \rightarrow \mathbb{P}(m[\![P]\!] \mid \prod_{i\in I} n_i[\![\{v/x\}P_i]\!] \mid M \mid N)}$$

(R-TAU)
$$\frac{}{\Gamma \rhd m[\![\tau.P + Q]\!] \mid M \rightarrow \mathbb{P}(m[\![P]\!] \mid M)}$$

(R-STRUCT)
$$\frac{M \equiv M' \quad \Gamma \rhd M' \rightarrow \Theta \quad \Theta \equiv \Delta}{\Gamma \rhd M \rightarrow \Delta}$$

Figure 6.3: Reduction Semantics for (high level) networks

(B-BROAD)
$$\frac{s \xrightarrow{c!v} p}{\Gamma \rhd n[\![s]\!] \xrightarrow{c.n!v} n[\![\Delta]\!]}\ \mathbb{P}(p) = \Delta$$

(B-REC)
$$\frac{s \xrightarrow{c?v} p}{\Gamma \rhd n[\![s]\!] \xrightarrow{c.m?v} n[\![\Delta]\!]}\ \mathbb{P}(p) = \Delta, \Gamma \vdash n \leftrightarrow m$$

(B-DEAF)
$$\frac{s \not\xrightarrow{c?v}}{\Gamma \rhd n[\![s]\!] \xrightarrow{c.m?v} \overline{n[\![s]\!]}}\ \Gamma \vdash m \leftrightarrow n$$

(B-DISC)
$$\frac{}{\Gamma \rhd n[\![s]\!] \xrightarrow{c.m?v} \overline{n[\![s]\!]}}\ \Gamma \vdash n \nleftrightarrow m$$

(B-$\mathbf{0}$)
$$\frac{}{\mathbf{0} \xrightarrow{c.m?v} \overline{\mathbf{0}}}$$

(B-τ)
$$\frac{s \xrightarrow{\tau} p}{\Gamma \rhd n[\![s]\!] \xrightarrow{n.\tau} n[\![\Delta]\!]}\ \mathbb{P}(p) = \Delta$$

(B-τ.PROP)
$$\frac{\Gamma \rhd M \xrightarrow{n.\tau} \Delta}{\Gamma \rhd M \mid N \xrightarrow{n.\tau} \Delta \mid \overline{N}}$$

(B-PROP)
$$\frac{\Gamma \rhd M \xrightarrow{c.m?v} \Delta,\ \Gamma \rhd N \xrightarrow{c.m?v} \Theta}{\Gamma \rhd M \mid N \xrightarrow{c.m?v} \Delta \mid \Theta}$$

(B-SYNC)
$$\frac{\Gamma \rhd M \xrightarrow{c.m!v} \Delta,\ \Gamma \rhd N \xrightarrow{c.m?v} \Theta}{\Gamma \rhd M \mid N \xrightarrow{c.m!v} \Delta \mid \Theta}$$

Figure 6.4: Intensional semantics of networks

latter is a probability distribution over networks obtained as an instantiation of Equation 6.1.

Note that Rule (R-STRUCT) uses a notion of structural congruence over distributions of system terms which has not yet been defined. We say that $\Delta \equiv \Theta$ if $\Delta = \sum_{i\in I} p_i \cdot S_i$ for some index set $i \in I$ such that $\sum_{i\in I} p_i = 1$ and $\Delta = \sum_{i\in I} p_i \cdot R_i$ for some collection of processes $\{R_i\}_{i\in I}$ such that $R_i \equiv S_i$.

The second kind of semantics we want to present is the Labelled Transition Semantics. A pre semantics for states is depicted in Figure 6.5; here a state evolves in a probability distribution. Note that in Figure 6.5 transitions are defined only for state based processes; this motivates our choice of not allowing probabilistic processes in the non-deterministic and matching operators. If we had allowed a system of the form $P + Q$ in our framework, where P, Q are probabilistic processes, we could not have used the rules of Figure 6.5. The reader could argue that we could have defined our presemantics directly for probabilistic processes, rather than restricting our attention to state based processes. We remark that this approach would have led to a very complicated semantics for processes.

The rules that define the semantics of networks are virtually identical to those defined in part I, though we redefine them in Figure 6.4 for the sake of consistency of the notation.

Let us perform some sanity checks for the operational semantics of networks:

Proposition 6.2.2 (Sanity Check)**.** (1) A distribution Δ over sSys is called *(node)-stable* if $\Delta(N) > 0$ and $\Delta(M) > 0$ implies $\mathrm{nodes}(N) = \mathrm{nodes}(M)$. If $\Gamma \rhd M \xrightarrow{\mu} \Delta$ then Δ is a stable distribution. Intuitively this

$$\text{(S-SND)}\quad \frac{}{c!\langle e\rangle.P \xrightarrow{c![\![e]\!]} \mathbb{P}(P)} \qquad \text{(S-}\omega\text{)}\quad \frac{}{\omega.\mathbf{0} \xrightarrow{\omega} \overline{\mathbf{0}}}$$

$$\text{(S-RCV)}\quad \frac{}{c?(x).P \xrightarrow{c?v} \mathbb{P}(P\{v/x\})} \qquad \text{(S-}\tau\text{)}\quad \frac{}{\tau.P \xrightarrow{\tau} \mathbb{P}(P)}$$

$$\text{(S-SUML)}\quad \frac{S \xrightarrow{\alpha} \Delta}{S + T \xrightarrow{\alpha} \Delta} \qquad \text{(S-SUMR)}\quad \frac{T \xrightarrow{\alpha} \Delta}{S + T \xrightarrow{\alpha} \Delta}$$

$$\text{(S-THEN)}\quad \frac{S \xrightarrow{\alpha} \Delta}{\text{if } b \text{ then } S \text{ else } T \xrightarrow{\alpha} \Delta}\,[\![b]\!] = \text{true} \qquad \text{(S-ELSE)}\quad \frac{T \xrightarrow{\alpha} \Delta}{\text{if } b \text{ then } S \text{ else } T \xrightarrow{\alpha} \Delta}\,[\![b]\!] = \text{false}$$

$$\text{(S-DEF)}\quad \frac{A(\tilde{x}) \Leftarrow S \quad \{\tilde{e}/\tilde{x}\}S \xrightarrow{\alpha} \Delta}{A\langle\tilde{e}\rangle \xrightarrow{\alpha} \Delta}$$

Figure 6.5: Pre-semantics of states

means that probabilistic choices are only ever made at the level of processes, not systems.

(2) If $\Gamma \rhd M \xrightarrow{\mu} \Delta$ then Δ is a stable distribution

(3) $\Gamma \rhd M \xrightarrow{c.m?v} \Delta$ implies

(a) $m \notin \text{nodes}(M)$,

(b) for every value w there exists some Δ' such that $\Gamma \rhd M \xrightarrow{c.m?w} \Delta'$,

(c) if $\Gamma \nvdash m$, then Δ is $\overline{M}$.

(4) $\Gamma \rhd M \xrightarrow{\mu} \Delta$, with μ being equal either to $c.m!v$ or $m.\tau$, implies $m \in \text{nodes}(M)$ and therefore $\Gamma \vdash m$.

□

We conclude this Section by showing that the reduction semantics of Figure 6.3 and the labelled transition semantics of Figure 6.4 coincide up-to structural congruence.

Theorem 6.2.3 (Harmony Theorem)**.** 1. if $\Gamma \rhd M \twoheadrightarrow \Delta$ then either $\Gamma \rhd M \xrightarrow{m.c!v} \Theta$ or $\Gamma \rhd M \xrightarrow{m.\tau} \Theta$ for some node m, channel c, value v and distribution $\Theta \equiv \Delta$,

2. if $\Gamma \rhd M \xrightarrow{m.c!v} \Delta$ then $\Gamma \rhd M \twoheadrightarrow \Delta$

3. if $\Gamma \rhd M \xrightarrow{m.\tau} \Delta$ then $\Gamma \rhd M \twoheadrightarrow \Delta$.

Proof. The proof is virtually identical to the one of Theorem 2.4.10 at Page 34. The proof requires to prove the probabilistic variant of all the structural properties proved in Section 2.4, which have been omitted for the sake of simplicity. □

Remark 6.2.4. Theorem 6.2.3 does not hold for distributions of networks. Specifically, it is possible to exhibit a network distribution Δ such that $\Delta \twoheadrightarrow \Theta$, but there exists no distribution $\Theta' \equiv \Theta$ such that either $\Delta \xrightarrow{m.c!v} \Theta'$ or $\Delta \xrightarrow{m.\tau} \Theta'$.

To this end, consider the least connectivity graph Γ such that $\Gamma \vdash m$ for some node m (that is, Γ consists of the sole node m) and let $\Delta = \mathbb{P}((\Gamma \rhd m[\![m.c!\langle v\rangle_{0.5} \oplus \tau]\!]))$. It is straightforward to note that $\Delta \twoheadrightarrow \overline{m[\![\mathbf{0}]\!]}$; however,

we do not have $\Delta \xrightarrow{m.c!v} \Theta$, for some distribution Θ, for the state $\Gamma \triangleright m[\![\tau]\!]$ cannot perform such a transition. Similarly, we can show that there is no distribution such that $\Delta \xrightarrow{m.\tau} \Theta$.

6.3 Testing Networks

The aim of this Section is that of introducing a behavioural theory based on a probabilistic generalisation of the *Hennessy* and *De Nicola* testing preorders [17]. In order to develop our testing framework, we will exploit the mathematical tools introduced in Section 6.1.

Let us recall the intuitive idea behind testing; a network is composed with another one, which takes the name of *testing network*. The composition of these two networks is then isolated from the external environment, in the sense that no external agent (in our case nodes in the interface of the composed network) can interfere with its behaviour; we will shortly present how such a task can be accomplished. The composition of the two networks, isolated from the external environment, takes the name of *experiment*.

Once these two operations (composition with a test and isolation from the external environment) have been performed, the behaviour of the resulting experiment is analysed to check whether there exists a computation that yields to a state which is successful. This task can be accomplished by relying on testing structures, which will be presented shortly.

At an informal level, successful states in our languages coincide with those associated with networks where at least the code running in one node can perform the action ω. For network have probabilistic behaviour, each computation will be associated with the probability of reaching a successful state; thus, every experiment will be associated with a set of success probabilities, one for each of its computation.

Let us now look at how the procedure explained above can be formalised; the topic of composing networks has already been addressed in detail in Chapter 3; since the composition operators we have introduced depend only on the connectivity graph of networks, all the results established for them in chapters 3 and 4 still hold in the probabilistic setting. We recall that composition operators are defined only for *composable networks* (see Definition 3.1.3 at Page 41); the set of (probabilistic) composable networks is denoted by CNets.

To model experiments and their behaviour, we rely on the following mathematical structure.

Definition 6.3.1. A *Testing structure* (TS) is a triple $\langle S, \rightarrow, \omega \rangle$ where

(i) S is a set of states,

(ii) the relation $\rightarrow$ is a subset of $S \times \mathcal{D}(S)$,

(iii) ω is a success predicate over S, that is $\omega : S \rightarrow \{\mathsf{true}, \mathsf{false}\}$. □

□

Testing structures can be seen as (degenerate) pLTSs where the only possible action corresponds to the internal activity τ, and the transition $\xrightarrow{\tau}$ with $\rightarrow$.

Conversely every pLTS automatically determines a testing structure, by concentrating on the relation $\xrightarrow{\tau}$.

Every network can be converted to a testing structure by defining the transition relation $\rightarrow$ to coincide with the reduction transition $\twoheadrightarrow$. Also, the success predicate ω is defined to be true for all, and only all, those networks in which a node can perform the success action $\xrightarrow{\omega}$. Formally, we have the following:

Example 6.3.2. The main example of a TS is given by

$$\langle \mathsf{Nets}, \longmapsto, \Omega \rangle$$

where

(i) $(\Gamma \triangleright M) \longmapsto (\Gamma \triangleright \Delta)$, with $\Gamma \triangleright M$ being a state based network, whenever

(a) $\Gamma \rhd M \xrightarrow{m.\tau} \Delta$ for some $m \in \text{nodes}(M)$

(b) or, $\Gamma \rhd M \xrightarrow{c.m!v} \Delta$ for some value v, node name m and channel c

(ii)

$$\omega(M) = \begin{cases} \mathsf{true}, & M = M' \,|\, n[\![s]\!],\ s \xrightarrow{\omega} \text{ for some n} \\ \mathsf{false}, & \text{otherwise} \end{cases}$$

If $\omega(M)$ = true for some system term M, we say that a network $\Gamma \rhd M$, where Γ is an arbitrary connectivity graph, is ω-successful. Note that when recording an ω-success we do not take into account the node involved. □

As TSs can be seen as pLTSs, we can use in an arbitrary TS the various constructions introduced in Section 6.1. Thus the reduction relation $\longmapsto$ can be lifted to $\mathcal{D}(\mathsf{sSys}) \times \mathcal{D}(\mathsf{sSys})$ and we can make use of the concepts of hyper-derivatives and extreme-derivatives, introduced in Section 6.1, to model fragments of executions and maximal executions of a testing structure, respectively. Below we provide two simple examples that show how to reason about the behaviour of the testing structures presented in Example 6.3.2.

Example 6.3.3. Consider the testing structure associated with the network $\mathcal{N} = \Gamma_N \rhd k[\![q]\!]$, where Γ_N consists of the two nodes k, o and the only connection $\Gamma_N \vdash k \to o$. Assume the code q is given by the definition $q \Leftarrow q\tau.(_{0.5}\oplus c!\langle v\rangle.\mathbf{0})$. We can show that, in the long run, this network will broadcast message v to the external location o by exhibiting a hyper derivation for it which terminates in the pointed distribution $\overline{\Gamma_N \rhd k[\![c!\langle v\rangle.\mathbf{0}]\!]}$. If we let $\mathcal{N}_1$ denote the configuration $\Gamma_N \rhd \Gamma_N \rhd k[\![c!\langle v\rangle.\mathbf{0}]\!]$, we have the following hyper-derivation:

$$\begin{array}{lcll} \overline{\mathcal{N}} & = & \frac{1}{2}\cdot\overline{\mathcal{N}} & + \ \frac{1}{2}\cdot\overline{\mathcal{N}_1} \\ \frac{1}{2}\cdot\overline{\mathcal{N}} & \longmapsto & \frac{1}{2^2}\cdot\overline{\mathcal{N}} & + \ \frac{1}{2^2}\cdot\overline{\mathcal{N}_1} \\ & \vdots & & \\ \frac{1}{2^n}\cdot\overline{\mathcal{N}} & \longmapsto & \frac{1}{2^{n+1}}\cdot\overline{\mathcal{N}} & + \ \frac{1}{2^{n+1}}\cdot\overline{\mathcal{N}_1} \\ & \vdots & & \\ & & & \Delta' = \sum_{n=1}^{\infty} \frac{1}{2^n}\cdot\overline{\mathcal{N}_1} \end{array}$$

Now it is straightforward to check that $\Delta' = \overline{\mathcal{N}_1}$ and therefore we have the hyper-derivation $\mathcal{N} \Longrightarrow \overline{\mathcal{N}_1}$. □

An arbitrary network $\mathcal{N}$ can be tested by another (testing) network $\mathcal{T}$ provided $\mathcal{N} \between \mathcal{T}$ is well-defined. Executions of the resulting testing structure will then be checked to establish whether the network $\mathcal{M}$ satisfies a property the test was designed for; in such a case, the testing component of an experiment will reach a ω-successful state.

Executions, or maximal computations, correspond to extreme derivatives of $\mathcal{N} \between \mathcal{T}$, as defined in Section 6.1. Since the framework is probabilistic, each execution (that is extreme derivative) will be associated with a probability value, representing the probability that it will lead to an ω-successful state. Since the framework is also non deterministic the possible results of this test application is given by a non-empty set of probability values.

Definition 6.3.4. [Tabulating results] The *value* of a sub-distribution in a TS is given by the function $\mathcal{V} : \mathcal{D}_{sub}(S) \to [0,1]$, defined by $\mathcal{V}(\Delta) = \sum\{\Delta(s) \mid \omega(s) = \mathsf{true}\}$. Then the set of possible results from a sub-distribution Δ is defined by $\mathcal{R}(\Delta) = \{\mathcal{V}(\Delta') \mid \Delta \Longrightarrow \Delta'\}$. □ □

Example 6.3.5. Consider the testing network $\mathcal{T} = \Gamma_T \rhd o[\![t]\!]$, where Γ_T is the least connectivity graph such that $\Gamma_T \vdash o \leftrightarrow l$ and the code t is determined by $t \Leftarrow c?(x).\omega.\mathbf{0}$; further, let $\mathcal{N}$ as defined in Example 6.3.3. It is easy to check that $\mathcal{N} \between \mathcal{T}$ is well-defined, and is equal $\Gamma \rhd k[\![q]\!] \,|\, o[\![t]\!]$. So consider the testing structure associated with it; recall that we have the definition $q \Leftarrow q_{\,0.5}\oplus c!\langle v\rangle.\mathbf{0}$. For convenience let $\mathcal{N}_1 = \Gamma_N \rhd k[\![c!\langle v\rangle.\mathbf{0}]\!]$ as in

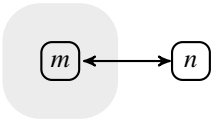


Figure 6.6: A connectivity graph for two recursive networks

the previous example, $\mathcal{N}_2 = \Gamma_N \rhd k[\![\mathbf{0}]\!]$ and $\mathcal{T}_\omega = \Gamma_T \rhd o[\![\omega.\mathbf{0}]\!]$. Then we have the following hyper-derivation for $\mathcal{N} \between \mathcal{T}$:

$$\begin{array}{lcll}
\overline{\mathcal{N} \between \mathcal{T}} & \longmapsto & (\frac{1}{2}\cdot\overline{\mathcal{N} \between \mathcal{T}} + \frac{1}{2}\cdot\overline{\mathcal{N}_1 \between \mathcal{T}}) & + \quad \varepsilon \\
\frac{1}{2}\cdot\overline{\mathcal{N} \between \mathcal{T}} + \frac{1}{2}\cdot\overline{\mathcal{N}_1 \between \mathcal{T}} & \longmapsto & (\frac{1}{2^2}\cdot\overline{\mathcal{N} \between \mathcal{T}} + \frac{1}{2^2}\cdot\overline{\mathcal{N}_1 \between \mathcal{T}}) & + \quad \frac{1}{2}\cdot\overline{\mathcal{N}_2 \between \mathcal{T}_\omega} \\
\vdots & & \vdots & \vdots \\
\frac{1}{2^n}\cdot\overline{\mathcal{N} \between \mathcal{T}} + \frac{1}{2^n}\cdot\overline{\mathcal{N}_1 \between \mathcal{T}} & \longmapsto & (\frac{1}{2^{n+1}}\cdot\overline{\mathcal{N} \between \mathcal{T}} + \frac{1}{2^{n+1}}\cdot\overline{\mathcal{N}_1 \between \mathcal{T}}) & + \quad \frac{1}{2^n}\cdot\overline{\mathcal{N}_2 \between \mathcal{T}_\omega} \\
\vdots & & \vdots & \vdots
\end{array}$$

were ε denotes the empty sub-distribution, that is the one with $\lceil\varepsilon\rceil = \emptyset$. We have therefore the hyper-derivative

$$\overline{\mathcal{N} \between \mathcal{T}} \Longrightarrow \varepsilon + \sum_{n=1}^{\infty} \frac{1}{2^n}\overline{\mathcal{N}_2 \between \mathcal{T}_\omega} = \overline{\mathcal{N}_2 \between \mathcal{T}_\omega}.$$

Further, the above hyper-derivation satisfies the constraints required by $\Longrightarrow$, defined in Section 6.1, and therefore we have the extreme derivative $\overline{\mathcal{N} \between \mathcal{T}} \Longrightarrow \overline{\mathcal{N}_2 \between \mathcal{T}_\omega}$. Since $\mathcal{V}(\overline{\mathcal{N}_2 \between \mathcal{T}_\omega}) = 1$ we can therefore deduce that $1 \in \mathcal{R}(\mathcal{N} \between \mathcal{T})$. □

We are now ready to define the probabilistic variant of the testing preorders. We have seen how to associate a non-empty set of probabilities, tabulating the possible outcomes from applying the test $\mathcal{T}$ to the network $\mathcal{N}$. As explained in [20] there are two natural ways to compare such sets, optimistically or pessimistically.

Definition 6.3.6. [Testing networks] For $\mathcal{M}_1, \mathcal{M}_2 \in \mathsf{CNets}$ we write $\mathcal{M}_1 \sqsubseteq_{\text{may}} \mathcal{M}_2$ if

(i) $\mathsf{Input}(\mathcal{M}_1) = \mathsf{Input}(\mathcal{M}_2)$ and $\mathsf{Output}(\mathcal{M}_1) = \mathsf{Output}(\mathcal{M}_2)$,

(ii) for every network $\mathcal{T} \in \mathsf{Nets}$ such that both $\mathcal{M}_1 \between \mathcal{T}$ and $\mathcal{M}_2 \between \mathcal{T}$ are defined, then for any outcome $p \in \mathcal{R}(\mathcal{M}_1 \between \mathcal{T})$ there exists $p' \geq p$ such that $p' \in \mathcal{R}(\mathcal{M}_2 \between \mathcal{T})$.

For $\mathcal{M}_1, \mathcal{M}_2 \in \mathsf{CNets}$ we write $\mathcal{M}_1 \sqsubseteq_{\text{must}} \mathcal{M}_2$ if

(i) $\mathsf{Input}(\mathcal{M}_1) = \mathsf{Input}(\mathcal{M}_2)$ and $\mathsf{Output}(\mathcal{M}_1) = \mathsf{Output}(\mathcal{M}_2)$,

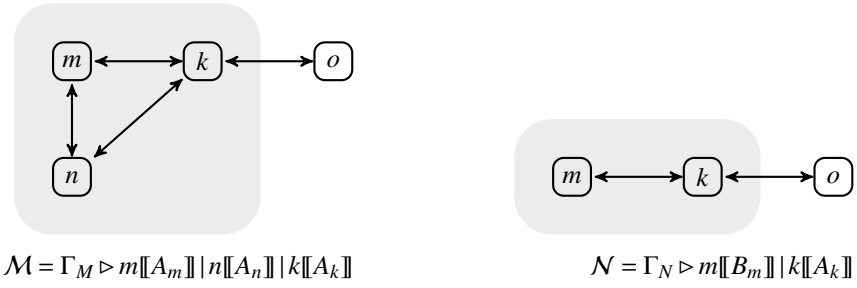
(ii) for every network $\mathcal{T} \in \mathsf{Nets}$ such that both $\mathcal{M}_1 \between \mathcal{T}$ and $\mathcal{M}_2 \between \mathcal{T}$ are defined and for any outcome $p' \in \mathcal{R}(\mathcal{M}_2 \between \mathcal{T})$ there exists $p' \leq p$ such that $p' \in \mathcal{R}(\mathcal{M}_1 \between \mathcal{T})$.

We use $\mathcal{M}_1 =_{\text{may}} \mathcal{M}_2$ as an abbreviation for $\mathcal{M}_1 \sqsubseteq_{\text{may}} \mathcal{M}_2$ and $\mathcal{M}_2 \sqsubseteq_{\text{may}} \mathcal{M}_1$. The equivalence $\mathcal{M}_1 =_{\text{must}} \mathcal{M}_2$ is defined similarly. □ □

Let us exhibit an example that show how the probabilistic testing preorders can be used.

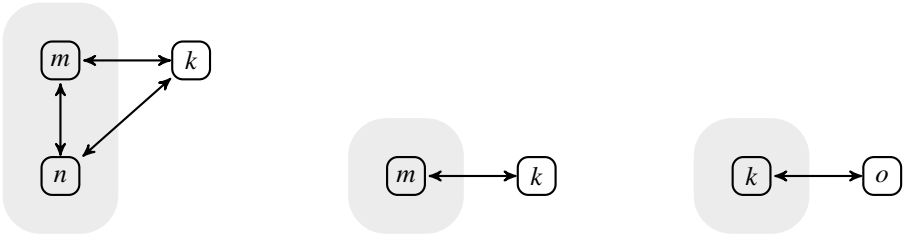
Example 6.3.7 (Distinguishing Probabilistic Networks)**.** Consider the connectivity graph depicted in Figure 6.6, and consider the networks $\mathcal{M} = \Gamma \rhd m[\![c!\langle v\rangle.(c!\langle w\rangle + d!\langle w'\rangle)]\!]$ and $\mathcal{N} = \Gamma \rhd m[\![(c!\langle v\rangle.c!\langle w'\rangle) + c!\langle v\rangle.d!\langle w'\rangle]\!]$.

Let $\mathcal{T} = \Gamma_T \rhd o[\![c?(x).(c?(x).\omega_{0.5} \oplus d?(x).\omega)]\!]$, where Γ_T is the connectivity graph consisting of the sole node n. It is straightforward to note that $1 \in \mathcal{R}(\mathcal{M} \between \mathcal{T})$; intuitively, after node m has broadcast value v along channel c, node n decides with probability 0.5 to detect a second value broadcast along channel c before reaching a successful configuration, and with probability 0.5 to detect the value along another channel d. However, after broadcasting a value along channel c, node m can non-deterministically decide to broadcast a second value



$\mathcal{M} = \Gamma_M \rhd m[\![A_m]\!] \mid n[\![A_n]\!] \mid k[\![A_k]\!]$ $\quad$ $\mathcal{N} = \Gamma_N \rhd m[\![B_m]\!] \mid k[\![A_k]\!]$

Figure 6.7: Two networks with a common sub-network



$\mathcal{M}_1 = \Gamma_1 \rhd m[\![A_m]\!] \mid n[\![A_n]\!]$ $\quad$ $\mathcal{N}_1 = \Gamma_2 \rhd m[\![B_m]\!]$ $\quad$ $\mathcal{K} = \Gamma_K \rhd k[\![B_k]\!]$

Figure 6.8: Decomposition of the networks $\mathcal{M}$ and $\mathcal{N}$

along channel c or d, so that it is always able to perform a broadcast which allows the testing component to reach a successful configuration.

This is not true for $\mathcal{N} \fatslash \mathcal{T}$; in fact, after node m has broadcast the first value, there is only one possible channel (either c or d) along which the second value can be broadcast. Thus, half of the time the network $\mathcal{N}$ is not able to make the test $\mathcal{T}$ evolve in a successful configuration. Formally, we have that whenever $p \in \mathcal{R}(\mathcal{N} \fatslash \mathcal{T})$ then $p \leq 0.5$.

We have found a test which is able to distinguish $\mathcal{N}$ from $\mathcal{M}$, that is $\mathcal{N} \sqsubseteq_{\text{may}} \mathcal{M}$. □

Example 6.3.7 shows that the probabilistic choice operator augmented the observational power of tests. In fact, it is easy to show that in the non-probabilistic framework the equivalence $\mathcal{M} =_{\text{may}} \mathcal{N}$, where $\mathcal{M}$ and $\mathcal{N}$ are the networks as in Example 6.3.7, holds.

In contrast with the (non-probabilistic) framework, where a computation could either be unsuccessful or successful, the probabilistic preorders $\sqsubseteq_{\text{may}}$ and $\sqsubseteq_{\text{must}}$ have been defined by comparing the probabilities of reaching a successful computation in an experiment. As we could expect, the two introduced preorders are compositional.

Proposition 6.3.8 (Compositionality)**.** *Let $\mathcal{M}_1, \mathcal{M}_2$ be two networks such that $\mathcal{M}_1 \sqsubseteq_{may} \mathcal{M}_2$, and let $\mathcal{N}$ be another network such that both $\mathcal{M}_1 \fatslash \mathcal{N}$ and $\mathcal{M}_2 \fatslash \mathcal{N}$ are defined. Then $\mathcal{M}_1 \fatslash \mathcal{N} \sqsubseteq_{may} \mathcal{M}_2 \fatslash \mathcal{N}$.*

A similar result holds for the preorder $\sqsubseteq_{must}$.

Proof. A direct consequence of $\fatslash$ being both associative and interface preserving. □

An application of this Compositionality result is given by the following Example:

Example 6.3.9. Consider the networks $\mathcal{M}$ and $\mathcal{N}$ in Figure 6.7, where the codes at the various nodes are given

by

$$\begin{aligned} A_m &\Leftarrow c!\langle v\rangle.\mathbf{0} \\ A_n &\Leftarrow c?(x).d!\langle w\rangle.\mathbf{0} \\ A_k &\Leftarrow c?(x).d?(y).e!\langle u\rangle.\mathbf{0} \\ B_n &\Leftarrow c!\langle v\rangle.d!\langle w\rangle.\mathbf{0} \end{aligned}$$

It is possible to write both of them respectively as $\mathcal{M}_1 \between \mathcal{K}$ and $\mathcal{N}_1 \between \mathcal{K}$, where the networks $\mathcal{M}_1, \mathcal{N}_1$ and $\mathcal{K}$ are depicted in Figure 6.8. In order to prove that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$, it is therefore sufficient to focus on their respective sub-networks $\mathcal{M}_1$ and $\mathcal{N}_1$, and prove $\mathcal{M}_1 \sqsubseteq_{\text{may}} \mathcal{N}_1$. The equivalence of the two original networks will then follow from a direct application of Proposition 6.3.8. □

Chapter 7

Proof Methods for Probabilistic Networks

In this chapter we exhibit sound proof methods to check whether two networks can be related via either the may-testing or the must-testing preorder. In [20] it was shown that the may-testing preorder over the process calculus pCSP can be characterised in terms of certain kinds of simulations over a probabilistic labelled transition system, while the must-testing preorder is characterised by failure simulations.

Here we consider the simulation preorder which is induced by a non-standard definition of weak extensional actions, which are defined in order to deal with the local broadcast features which characterize our calculus. We show that our simulation preorder provides a sound proof methodology for checking whether two networks can be compared with respect to the $\sqsubseteq_{\text{may}}$ preorder; however our simulations fail to be complete.

The notion of failure simulation, in which two distributions can be distinguished by looking at the transitions they cannot perform, is not appropriate as a proof methodology for the must-testing preorder. We have already shown in Chapter 4 that, due to the non-blocking nature of broadcast transitions, it is necessary to focus on *deadlock configurations*. We propose a novel kind of simulations, called *deadlock simulations*, and we prove that they are sound with respect to the must-testing preorder.

This Chapter is organised as follows; in Section 7.1 we define the activities of a network which can be observed by nodes in their interface, thus defining both strong and weak extensional actions for a network. As it is not surprising, these are the probabilistic variant of the transitions defined in Chapter 4. The extensional semantics leads to the notion of (weak probabilistic) simulation between networks.

In Section 7.2 we show that simulations are indeed sound with respect to the may-testing preorder. In Section 7.3 we explain why the proof methodology is not complete.

Finally, in Section 7.4 we introduce deadlock simulations and we show that they are sound with respect to the must-testing preorder.

7.1 Extensional Semantics

Here the idea is to design a pLTS over the collection of networks Nets such that whenever two networks are simulation related as defined in [20, 61], then they will also be testing related, $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$. The intensional semantics in Section 6.2 already provides a pLTS and it is instructive to see why this is not appropriate.

Consider $\mathcal{M} = \Gamma_M \rhd l[\![p]\!]$ and $\mathcal{N} = \Gamma_N \rhd k[\![p]\!]$, where Γ_M, Γ_N are the least connectivity graphs such that $\Gamma_M \vdash l \leftrightarrow o$, $\Gamma_N \vdash k \leftrightarrow o$. Suppose that the code p and running at l and k in $\mathcal{M}, \mathcal{N}$, respectively is $c!\langle v\rangle.\mathbf{0}$. Then we would expect $\mathcal{M}$ and $\mathcal{N}$ to be behaviourally indistinguishable. However $\mathcal{M}$ will have an output action, labelled $c.l!v$, which is not possible for $\mathcal{N}$. So output actions cannot record their source node. What turns out to be important is the target nodes. For example if in $\mathcal{M}$ we added a new node m to the interface, with a connection to l then we would be able to distinguish $\mathcal{M}$ from $\mathcal{N}$; the required test would simply place some

appropriate testing code at the new node m.

We now present an extensional semantics for networks; here the visible actions consist of activities which can be detected (hence tested) by placing code at the interface of a network. In this semantics we have internal, input and output actions.

Definition 7.1.1. [Extensional actions] The actions of the extensional semantics are defined as follows:

(1) **internal**, $(\Gamma \rhd M) \xrightarrow{\tau} (\Gamma \rhd \Delta)$; some internal activity reduces the system M, relative to the connectivity Γ, to some system N, where $N \in \lceil \Delta \rceil$. Here the internal activity of a network coincides either with some node performing a silent move $m.\tau$ or broadcasting a value which cannot be detected by any node in the interface of the network itself.

Formally, $(\Gamma \rhd M) \xrightarrow{\tau} (\Gamma \rhd \Delta)$ whenever

(a) $\Gamma \rhd M \xrightarrow{m.\tau} \Delta$

(b) or $\Gamma \rhd M \xrightarrow{c.n!v} \Delta$ for some value v, channel c and node name n satisfying $\Gamma \vdash n \leftrightarrow m$ implies $m \in \mathrm{nodes}(M)$

Note that we are using the notation given in Section 6.2 for defining distributions. Here Δ is a distribution over sSys and so $(\Gamma \rhd \Delta)$ is a distribution over networks; however all networks in its support use the same network connectivity Γ.

(2) **input**, $(\Gamma \rhd M) \xrightarrow{c.n?v} (\Gamma \rhd \Delta)$; an observer placed at node n can send the value v along the channel c to the network $(\Gamma \rhd M)$. For the observer to be able to place the code at node n we must have $n \in \mathsf{Int}(\Gamma \rhd M)$.

Formally $(\Gamma \rhd M) \xrightarrow{c.n?v} (\Gamma \rhd \Delta)$ whenever

(a) $\Gamma \rhd M \xrightarrow{c.n?v} \Delta$

(b) $n \in \mathsf{Int}(\Gamma \rhd M)$

(3) **output**, $(\Gamma \rhd M) \xrightarrow{c!v \rhd \eta} (\Gamma \rhd \Delta)$, where η is a non-empty set of nodes; an observer placed at any node $n \in \eta$ can receive the value v along the channel c. For this to happen each node $n \in \eta$ must be in $\mathsf{Int}(\Gamma \rhd M)$, and there must be some code running at some node in M which can broadcast along channel c to each such n.

Formally, $(\Gamma \rhd M) \xrightarrow{c!v \rhd \eta} (\Gamma \rhd \Delta)$ whenever

(i) $(\Gamma \rhd M) \xrightarrow{c.m!v} \Delta$ for some node m

(ii) $\eta = \{n \in \mathsf{Int}(\Gamma \rhd M) \mid \Gamma \vdash m \to n\} \neq \emptyset$. □

□

These extensional actions endow the set of networks with the structure of a pLTS. Thus the terminology used for pLTSs is extended to networks, so that in the following we will use terms such as finitary networks or finite branching networks. Also, there is a close relationship between extensional τ-actions and the reduction relation of testing structures.

In the following we will need *weak* versions of extensional actions, which abstract from internal activity, provided by the relation $\xrightarrow{\tau}$. Internal activity can be modelled by the hyper-derivation relation $\Longrightarrow$, which is a probabilistic generalisation of the more standard weak internal relation $\xrightarrow{\tau}{}^{*}$.

Definition 7.1.2. [Weak extensional actions]

(1) Let $\mathcal{M} \overset{\tau}{\Longrightarrow} \Delta$ whenever we have the hyper-derivation $\mathcal{M} \Longrightarrow \Delta$

(2) $\mathcal{M} \overset{c.n?v}{\Longrightarrow} \Delta$ whenever $\mathcal{M} \Longrightarrow \xrightarrow{c.n?v} \Longrightarrow \Delta$

(3) Let $\mathcal{M} \overset{c!v \rhd \eta}{\Longrightarrow} \mathcal{N}$ be the least relation satisfying:

(a) $\mathcal{M} \Longrightarrow \xrightarrow{c!v \rhd \eta} \Longrightarrow \Delta$ implies $\mathcal{M} \overset{c!v \rhd \eta}{\Longrightarrow} \mathcal{N}$

Figure 7.1: Ensuring soundness

(b) $\mathcal{M} \overset{c!v \rhd \eta_1}{\Longrightarrow} \Delta'$, $\Delta' \overset{c!v \rhd \eta_2}{\Longrightarrow} \Delta$, where $\eta_1 \cap \eta_2 = \emptyset$, implies $\mathcal{M} \overset{c!v \rhd (\eta_1 \cup \eta_2)}{\Longrightarrow} \Delta$ □

□

These weak actions endow the set of networks Nets with the structure of another pLTS, called the *extensional pLTS* and denoted by $\mathsf{pLTS}_{\mathsf{Nets}}$. We can therefore use such weak actions to define a simulation preorder between networks, as in [20].

Definition 7.1.3. [Simulation preorder] In $\mathsf{pLTS}_{\mathsf{Nets}}$ we let $\lhd_{sim}$ denote the largest relation in $\mathsf{Nets} \times \mathcal{D}(\mathsf{Nets})$ such that if $s \lhd_{sim} \Theta$ then:

- if $\omega(s) =$ true, then $\Theta \overset{\tau}{\Longrightarrow} \Theta'$ such that for every $t \in \lceil \Theta' \rceil, \omega(t) =$ true
- otherwise, whenever $\overline{s} \overset{\mu}{\Longrightarrow} \Delta'$, for $\mu \in \mathsf{Act}_\tau$, then there is a $\Theta' \in \mathcal{D}(S)$ with $\Theta \overset{\mu}{\Longrightarrow} \Theta'$ and $\Delta' \overline{\lhd_{sim}} \Theta'$.

We often use $s_1 \lhd_{sim} s_2$ in place of $s_1 \lhd_{sim} \overline{s_2}$. □ □

This is a mild generalisation of the corresponding definition in [20] where we factor in the presence of the success predicate $\omega()$.

Let us recall why the definition of output actions in Definition 7.1.2(3) is non-standard. Informally speaking, the definition of weak extensional output actions expresses the capability of simulating broadcast through multicast; that is, a single broadcast action detected by a set of nodes η can be matched by a sequence of broadcast actions (possibly interrupted by internal actions), detected respectively by $\eta_1, \cdots, \eta_i \subseteq \eta$, provided that the collection $\{\eta_1, \cdots, \eta_i\}$ is a partition of η. This constraint is needed to ensure that

(i) every node in η will detect the transmitted value, and

(ii) no node in η will detect the value more than once.

As we will see, simulations can be used as a proof methodology for the may-testing preorder. To ensure that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ for two networks $\mathcal{M}$, $\mathcal{N}$, it is sufficient to exhibit a simulation between them $\mathcal{M} \lhd_{sim} \mathcal{N}$. The same result holds for deadlock simulations with respect to the must testing preorder.

We end this section with an example which reinforce the delicacy of the issues involved in achieving soundness.

Example 7.1.4. Soundness requires that the extensional output actions records the set of target nodes, rather than single nodes. Consider $\mathcal{M} = \Gamma \rhd k_1[\![\mathbf{0}]\!] \,|\, k_2[\![c!\langle 1\rangle]\!]$ and $\mathcal{N} = \Gamma \rhd k_1[\![c!\langle 1\rangle]\!] \,|\, k_2[\![\mathbf{0}]\!]$, where the connectivity is given in Figure 7.1. $\mathcal{N} \not\sqsubseteq_{\text{may}} \mathcal{M}$ because of the test $\mathcal{T} = \Gamma_T \rhd T$, where T is the code $l_1[\![c?.c!]\!] \,|\, l_2[\![c?.c!]\!] \,|\, o[\![c?.c?.\omega]\!]$. Moreover $\mathcal{N} \not\lhd_{sim} \mathcal{M}$ because $\mathcal{N}$ can perform the output action labelled $c!\langle 1\rangle \rhd \{l_1, l_2\}$, which can not be matched by $\mathcal{M}$.

However suppose we were to restrict η in the definition of extensional output actions, part (3) of Definition 7.1.1, to be singleton sets of node names. Then in the resulting pLTS it is easy to check that $\mathcal{M}$ can simulate $\mathcal{N}$. In other words with this simplification the resulting simulations would not be sound; that is, Theorem 7.2.1 would no longer hold. □

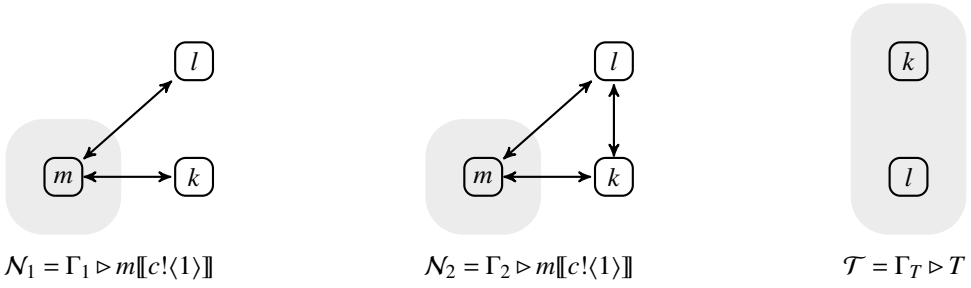


Figure 7.2: Ensuring soundness

7.2 Soundness for May-testing

In this Section we prove the following result:

Theorem 7.2.1. [Soundness for may-testing] Suppose $\mathcal{N}_1, \mathcal{N}_2$ are finitary networks such that $\mathsf{Input}(\mathcal{N}_1) = \mathsf{Input}(\mathcal{N}_2), \mathsf{Output}(\mathcal{N}_1) = \mathsf{Output}(\mathcal{N}_2)$. Then $\mathcal{N}_1 \lhd_{sim} \mathcal{N}_2$ in $\mathsf{pLTS}_{\mathsf{Nets}}$ implies $\mathcal{N}_1 \sqsubseteq_{\mathrm{may}} \mathcal{N}_2$.

The proof of Soundness relies on an alternative characterisations of the simulation preorder in $\mathsf{pLTS}_{\mathsf{Nets}}$, which is essentially a simplification of what it means to be a simulation. With simulations in $\mathsf{pLTS}_{\mathsf{Nets}}$ weak actions are matched against weak actions; an alternative would be simply to require that strong actions are matched by weak actions.

Definition 7.2.2. [Simple simulations] In $\mathsf{pLTS}_{\mathsf{Nets}}$ we let $\lhd^{\mathrm{s}}$ denote the largest relation in $\mathsf{Nets} \times \mathcal{D}(\mathsf{Nets})$ such that if $\mathcal{M} \lhd^{\mathrm{s}} \Theta$ then:

- if $\omega(\mathcal{M}) = \mathsf{true}$ for any $\omega \in \Omega$ then $\Theta \stackrel{\tau}{\Longrightarrow} \Theta'$ such that $\omega(\Theta') = \mathsf{true}$
- otherwise,
  (i) whenever $\overline{\mathcal{M}} \stackrel{\mu}{\longrightarrow} \Delta'$ there is a $\Theta' \in \mathcal{D}(\mathsf{Nets})$ with $\Theta \stackrel{\mu}{\Longrightarrow} \Theta'$ and $\Delta' \; \overline{\lhd^{\mathrm{s}}} \; \Theta'$.

□

Theorem 7.2.3. [Alternative characterisation] In $\mathsf{pLTS}_{\mathsf{Nets}}$, $\mathcal{M} \lhd_{sim} \Theta$ if and only if $\mathcal{M} \lhd^{\mathrm{s}} \Theta$, provided that $\mathcal{M}$ is a finitary network, and Θ is a finitary distribution of networks (that is, every network in its support is finitary),

Proof. (Outline) Practically identical to the corresponding proof in [19]. The difficulty is to check that if $s \lhd_{sim} \Theta$ and $s \Longrightarrow \Delta'$ then $\Theta \Longrightarrow \Theta'$ such that $\Delta' \; \overline{\lhd^{\mathrm{s}}} \; \Theta'$; see Theorem 7.20 of [19]. □

Theorem 7.2.4. Suppose $\mathsf{Input}(\mathcal{M}) = \mathsf{Input}(\mathcal{N})$, $\mathsf{Output}(\mathcal{M}) = \mathsf{Output}(\mathcal{N})$ and both $\mathcal{M} \between (\Gamma \rhd n[\![p]\!])$ and $\mathcal{N} \between (\Gamma \rhd n[\![p]\!])$ are defined. Then $\mathcal{M} \lhd^{\mathrm{s}} \mathcal{N}$ implies $\mathcal{M} \between (\Gamma \rhd n[\![p]\!]) \lhd^{\mathrm{s}} \mathcal{N} \between (\Gamma \rhd n[\![p]\!])$.

The proof of this result is quite long and technical, and is therefore relegated to an independent subsection, Section 7.2.1 below.

Corollary 7.2.5. [Compositionality] Suppose $\mathsf{Input}(\mathcal{M}_1) = \mathsf{Input}(\mathcal{M}_2)$, $\mathsf{Output}(\mathcal{M}_1) = \mathsf{Output}(\mathcal{N}_2)$. Then $\mathcal{M}_1 \lhd^{\mathrm{s}} \mathcal{M}_2$ implies $(\mathcal{M}_1 \between \mathcal{N}) \lhd_{sim} (\mathcal{M}_1 \between \mathcal{N})$ whenever both these networks are defined.

Proof. By induction on the number of nodes in $\mathcal{N}$, using the previous theorem, Theorem 7.2.3. □

We also need to relate the simulation preorder $\lhd_{sim}$ to the valuation of distributions, as given in Definition 6.3.4. First an auxiliary result.

Lemma 7.2.6. Let Δ, Θ be distributions in $\mathsf{pLTS}_{\mathsf{Nets}}$ such that $\Delta \; \overline{\lhd_{sim}} \; \Theta$; then $\Theta \Longrightarrow \Theta'$ such that $\mathcal{V}(\Delta) \leq \mathcal{V}(\Theta')$.

Proof. There are two cases.

(i) First suppose Δ is a point distribution $\overline{s}$. If the predicate $\omega(s)$ is equal to false, $\mathcal{V}(\overline{s}) = 0$. In this case, we recall that Theorem 6.1.5 (4) ensures that there exists at least one extreme derivative Θ' of Θ, for which $0 \leq \mathcal{V}(\Theta')$ trivially holds.

Otherwise the predicate $\omega(s)$ is satisfied and $\mathcal{V}(\overline{s})$ has to be 1. Since $s \lhd_{sim} \Theta$ we know $\Theta \Longrightarrow \Theta'$ such that for all $s' \in \Theta', \omega(s') = \text{ true }$. This means that $\mathcal{V}(\Theta') = 1$; moreover, as every state in $\lceil \Theta' \rceil$ is a successful state, we also have that $\Theta \Longrightarrow\!\!\!\!\!\rightarrow \Theta'$

(ii) Otherwise Θ can be written as $\sum_{s \in \lceil \Delta \rceil} \Delta(s) \cdot \Theta_s$ where $s \lhd_{sim} \Theta_s$ for each s in the support of Δ. By part (i) each $\Theta_s \Longrightarrow\!\!\!\!\!\rightarrow \Theta'_s$ such that $\mathcal{V}(\overline{s}) \leq \mathcal{V}(\Theta'_s)$. As an extreme derivative is also a hyper-derivative, we can combine these to obtain a hyper derivation for Θ, using Theorem 6.1.5 (3). This leads to

$$\Theta = \sum_{s \in \lceil \Delta \rceil} \Delta(s) \cdot \Theta_s \Longrightarrow \sum_{s \in \lceil \Delta \rceil} \Theta'_s = \Theta'$$

As for every $s \in \lceil \Delta \rceil$, $t \in \lceil \Theta'_s \rceil$ we have that $t \xrightarrow{\tau}$ implies $\omega t = \text{ true }$, this condition is respected also by all states in $\lceil \Theta' \rceil$. Thus, the hyper derivation $\Theta \Longrightarrow \Theta'$ is also an extreme derivation. Finally, the quantity $\mathcal{V}(\Delta) = \sum\{\Delta(s) \mid \omega(s) = \text{ true }\}$ can be rewritten as $\sum_{s \in \lceil \Delta \rceil} \mathcal{V}(\overline{s})$, leading to

$$\mathcal{V}(\Delta) = \sum_{s \in \lceil \Delta \rceil} \mathcal{V}(\overline{s}) \leq \sum_{s \in \lceil \Delta \rceil} \mathcal{V}(\Theta'_s) = \mathcal{V}(\Theta').$$

□

Theorem 7.2.7. In $\mathsf{pLTS}_{\mathsf{Nets}}$, $\Delta \overline{\lhd_{sim}} \Theta$ implies $\mathcal{R}(\Delta) \sqsubseteq_{\text{may}} \mathcal{R}(\Theta)$.

Proof. Suppose $\Delta \Longrightarrow\!\!\!\!\!\rightarrow \Delta'$. We have to find a derivation $\Theta \Longrightarrow\!\!\!\!\!\rightarrow \Theta'$ such that $\mathcal{V}(\Delta') \leq \mathcal{V}(\Theta')$. We can use the definition of $\lhd_{sim}$ to find a derivation $\Theta \Longrightarrow \Theta''$ such that $\Delta' \overline{\lhd_{sim}} \Theta''$. Applying the previous lemma we obtain $\Theta'' \Longrightarrow\!\!\!\!\!\rightarrow \Theta'$ such that $\mathcal{V}(\Delta') \leq \mathcal{V}(\Theta')$. The result follows since Theorem 6.1.5 gives $\Theta \Longrightarrow\!\!\!\!\!\rightarrow \Theta'$.

□

Proof of Theorem 7.2.1:

This is now a straightforward application of Compositionality and Theorem 7.2.7.

Let us assume that $\mathcal{N}_1 \lhd_{sim} \mathcal{N}_2$. To prove the conclusion, $\mathcal{N}_1 \sqsubseteq_{\text{may}} \mathcal{N}_2$, we must show that $\mathcal{R}(\mathcal{N}_1 \between \mathcal{T}) \sqsubseteq_{\text{may}} \mathcal{R}(\mathcal{N}_2 \between \mathcal{T})$ for an arbitrary testing network $\mathcal{T}$ such that both $\mathcal{N}_1 \between \mathcal{T}$ and $\mathcal{N}_2 \between \mathcal{T}$ are defined. For such a $\mathcal{T}$ Compositionality entails $(\mathcal{N}_1 \between \mathcal{T}) \lhd_{sim} (\mathcal{N}_2 \between \mathcal{T})$, and now we can apply Theorem 7.2.7. □

7.2.1 Single Node Compositionality

The aim of this section is to outline the proof of Theorem 7.2.4; it may be safely skipped by the reader uninterested in the detail. The standard approach to compositionality for a behavioural preorder involves proving decomposition and composition results for the actions on which the pre-order depends. As an example decomposition would involve showing that an action $P_1 \mid\mid\mid P_2 \xrightarrow{\mu} \Delta_1 \mid\mid\mid \Delta_2$ can be decomposed into two components

$$P_1 \xrightarrow{\mu_1} \Delta_1 \qquad\qquad P_2 \xrightarrow{\mu_1} \Delta_2$$

where μ_i are such that any other pairs of actions $Q_1 \xrightarrow{\mu_1} \Theta_1$, $Q_2 \xrightarrow{\mu_1} \Theta_2$ can be recomposed into $Q_1 \mid\mid\mid Q_2 \xrightarrow{\mu} \Theta_1 \mid\mid\mid \Theta_2$.

Unfortunately such decomposition results do not hold in $\mathsf{pLTS}_{\mathsf{Nets}}$ for our operator $\between$.

Example 7.2.8. Let $\mathcal{M}, \mathcal{N}$ be defined by $(\Gamma_M \rhd m[\![c!\langle 0\rangle]\!])$, $(\Gamma_N \rhd n[\![c?.\omega]\!])$, where Γ_N is the trivial graph containing only one node, and Γ_M is determined by $\Gamma_M \vdash m \leftrightarrow n$.

Then in $\mathsf{pLTS}_{\mathsf{Nets}}$ $\mathcal{M} \between \mathcal{N} \xrightarrow{\tau} (\Gamma_M \rhd m[\![\mathbf{0}]\!]) \between (\Gamma_N \rhd n[\![\omega]\!])$. But this move can not be decomposed into individual actions in $\mathsf{pLTS}_{\mathsf{Nets}}$ from $\mathcal{M}$ and $\mathcal{N}$ respectively, as $\mathcal{N}$ cannot perform the transition $\mathcal{N} \xrightarrow{c?v} \overline{\Gamma_N \rhd n[\![\omega]\!]}$. □

Luckily, because we are only considering composition on the right-hand side by single node networks, we can work with the symmetric operator ||; as we will see this operator will support the decomposition and composition of actions in $\mathsf{pLTS}_{\mathsf{Nets}}$.

Proposition 7.2.9. Suppose $\mathcal{M} \between (\Gamma_n \rhd n[\![p]\!])$ is well-defined. Then there exists a Γ such that $(\Gamma \rhd n[\![p]\!])$ is well-defined, and $\mathcal{M} \between (\Gamma_n \rhd n[\![p]\!])$ coincides with $\mathcal{M} \,||\, (\Gamma \rhd n[\![p]\!])$.

Proof. Γ can be constructed by adding to Γ_n all nodes in $\mathcal{M}$ which are connected to n; also, we require all those nodes to be connected to n in Γ. □

Proposition 7.2.10. [Strong decomposition in $\mathsf{pLTS}_{\mathsf{Nets}}$]

(1) If $(\Gamma_M \rhd M) \,||\, (\Gamma_n \rhd n[\![s]\!]) \xrightarrow{\tau} \Delta$

then

- either $(\Gamma_M \rhd M) \xrightarrow{\tau} (\Gamma_M \rhd \Delta_M)$ and $\Delta = (\Gamma_M \rhd \Delta_M) \,||\, \overline{\Gamma_n \rhd n[\![s]\!]}$
- or $(\Gamma_n \rhd n[\![s]\!]) \xrightarrow{\tau} (\Gamma_n \rhd n[\![\Delta_n]\!])$ and $\Delta = \overline{(\Gamma_M \rhd M)} \,||\, (\Gamma_n \rhd \Delta_n)$
- or $(\Gamma_M \rhd M) \xrightarrow{c!v \rhd \{n\}} (\Gamma_M \rhd \Delta_M)$, $(\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c.m?v} (\Gamma_n \rhd n[\![\Delta_n]\!])$, with $\Gamma_n \vdash m \leftrightarrow n$ and $\Delta = (\Gamma_M \rhd \Delta_M) \,||\, \Gamma_n \rhd n[\![\Delta_n]\!]$,
- or $(\Gamma_M \rhd M) \xrightarrow{c.n?v} (\Gamma_M \rhd \Delta_M)$, $(\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c!v \rhd \eta} (\Gamma_n \rhd n[\![\Delta_n]\!])$, with $\eta \subseteq \mathrm{nodes}(M)$ and $\Delta = (\Gamma_M \rhd \Delta_M) \,||\, \Gamma_n \rhd n[\![\Delta_n]\!]$,

(2) If $(\Gamma_M \rhd M) \,||\, (\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c.m?v} \Delta$ then

- either $(\Gamma_M \rhd M) \xrightarrow{c.m?v} (\Gamma_M \rhd \Delta_M)$, and $\Delta = (\Gamma_M \rhd \Delta_M) \,||\, \overline{\Gamma_n \rhd n[\![s]\!]}$,
- or $(\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c.m?v} (\Gamma_n \rhd n[\![\Delta_n]\!])$ and $\Delta = \overline{\Gamma_M \rhd M} \,||\, (\Gamma_n \rhd n[\![\Delta_n]\!])$,
- or $(\Gamma_M \rhd M) \xrightarrow{c.m?v} (\Gamma_M \rhd \Delta_M)$, $(\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c.m?v} (\Gamma_n \rhd n[\![\Delta_M]\!])$ and $\Delta = (\Gamma_M \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Delta_n]\!])$,

(3) If $(\Gamma_M \rhd M) \,||\, (\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c!v \rhd \eta} \Delta$ then

- either $\Delta = (\Gamma \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Delta_n]\!])$ where $(\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c.m?v} (\Gamma_n \rhd n[\![\Delta_n]\!])$ for some $m \in \mathsf{Int}(\Gamma_n \rhd n[\![p]\!])$, and $(\Gamma_M \rhd M) \xrightarrow{c!v \rhd \eta \cup \{n\}} (\Gamma_M \rhd \Delta_M)$
- or $\Delta = (\Gamma_M \rhd \Delta_M) \,||\, \overline{(\Gamma_n \rhd n[\![s]\!])}$ where $(\Gamma_M \rhd M) \xrightarrow{c!v \rhd \eta} (\Gamma_M \rhd \Delta_M)$ and $n \notin \eta$.
- or $\Delta = (\Gamma_M \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Delta_n]\!])$, where $(\Gamma_M \rhd M) \xrightarrow{c.n?v} (\Gamma_M \rhd \Delta_M)$ and $(\Gamma_n \rhd n[\![s]\!]) \xrightarrow{c!v \rhd \eta'} (\Gamma \rhd n[\![\Delta_n]\!])$, and $\eta = \eta' \cap \mathrm{nodes}(M)$.

Proof. See the appendix. □

Next we consider how the weak actions performed by a node-stable distribution of the form $\Gamma_M \rhd \Delta \,||\, \Gamma_n \rhd n[\![\Theta]\!]$ can be inferred from those performed by $\Gamma_M \rhd \Delta$ and $\Gamma_n \rhd n[\![\Theta]\!]$ respectively.

Proposition 7.2.11. [Weak composition in $\mathsf{pLTS}_{\mathsf{Nets}}$] Suppose $(\Gamma_M \rhd \Delta) \,||\, (\Gamma_n \rhd n[\![\Theta]\!])$ is well-defined.

(i) $(\Gamma_M \rhd \Delta) \overset{\tau}{\Longrightarrow} (\Gamma_M \rhd \Delta_M), \Gamma_n \rhd n[\![\Theta]\!] \overset{\tau}{\Longrightarrow} \Gamma_n \rhd n[\![\Theta_n]\!]$ implies $(\Gamma_M \rhd \Delta) \,||\, (\Gamma_n \rhd n[\![\Theta]\!]) \overset{\tau}{\Longrightarrow} (\Gamma_M \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Theta_n]\!])$,

(ii) $(\Gamma_M \rhd \Delta) \overset{c!v \rhd \eta}{\Longrightarrow} \Gamma_M \rhd \Delta_M$, $n \notin \eta$ implies $(\Gamma_M \rhd \Delta) \,||\, (\Gamma_n \rhd n[\![\Theta]\!]) \overset{c!v \rhd \eta}{\Longrightarrow} (\Gamma_M \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Theta]\!])$,

(iii) $(\Gamma_M \rhd \Delta) \overset{c!v \rhd \eta}{\Longrightarrow} (\Gamma_M \rhd \Delta_M)$, $n \in \eta$ and $(\Gamma_n \rhd n[\![\Theta]\!]) \overset{c.m?v}{\Longrightarrow} (\Gamma_n \rhd n[\![\Theta_n]\!])$ for some $m \in \mathsf{Int}(\Gamma_n \rhd n[\![s]\!])$ implies $(\Gamma_M \rhd \Delta) \,||\, (\Gamma_n \rhd n[\![\Theta]\!]) \overset{c!v \rhd \eta \setminus \{n\}}{\Longrightarrow} (\Gamma_M \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Theta_n]\!])$,

(iv) $(\Gamma_M \rhd \Delta) \overset{c.n?v}{\Longrightarrow} \Gamma_M \rhd \Delta_M$ and $(\Gamma_n \rhd n[\![\Theta]\!]) \overset{c!v \rhd \eta}{\Longrightarrow}$ implies $(\Gamma_M \rhd \Delta) \,||\, (\Gamma_n \rhd n[\![\Theta]\!]) \overset{c!v \rhd \eta'}{\Longrightarrow} (\Gamma_M \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Theta_n]\!])$, where $\eta' = \eta \setminus \text{nodes}(\Delta)$.

(v) $(\Gamma_M \rhd \Delta) \overset{c.m?v}{\Longrightarrow} (\Gamma_M \rhd \Delta_M)$ and $(\Gamma_N \rhd n[\![s]\!]) \overset{c.m?v}{\Longrightarrow} (\Gamma_n \rhd p[\![\Delta_n]\!])$ implies $(\Gamma_M \rhd M) \,||\, (\Gamma_n \rhd n[\![s]\!]) \overset{c.m?v}{\Longrightarrow} (\Gamma_M \rhd \Delta_M) \,||\, (\Gamma_n \rhd n[\![\Delta_n]\!])$.

Proof. See the appendix. □

Before proving Theorem 7.2.4 we need another result that allows us to relate the actions performed by two single node networks with different connectivities; this is because the fact that the operator $\between$ being asymmetric is reflected in the fact that the application of Proposition 7.2.9 to a network of the form $(\Gamma_M \rhd M) \between (\Gamma_n \rhd p[\![n]\!])$ leads to a change in the connectivity graph of the network appearing in the right hand side of the composition.

Formally, let $(\Gamma_M \rhd M), (\Gamma_N \rhd N)$ and $(\Gamma_n \rhd n[\![s]\!])$ be three state based networks, and suppose both $(\Gamma_M \rhd M) \between (\Gamma_n \rhd n[\![s]\!])$ and $(\Gamma_N \rhd N) \between (\Gamma_n \rhd n[\![s]\!])$ are defined. Then

$$\begin{aligned} (\Gamma_M \rhd M) \between (\Gamma_n \rhd n[\![s]\!]) &= (\Gamma_M \rhd M) \,||\, (\Gamma_1 \rhd n[\![s]\!]) \\ (\Gamma_N \rhd N) \between (\Gamma_n \rhd n[\![s]\!]) &= (\Gamma_N \rhd N) \,||\, (\Gamma_2 \rhd n[\![s]\!]) \end{aligned}$$

Since the definitions of the connectivity graphs Γ_1 and Γ_2 depend on those of Γ_M and Γ_N, respectively, it is possible to obtain $\Gamma_1 \neq \Gamma_2$. Thus, when proving Theorem 7.2.4, we will need to deal with situations in which the two networks $\Gamma_M \rhd M$ and $\Gamma_N \rhd N$ are composed (via the $||$ operator) with networks having different connectivities. The following result, however, allows us to relate such networks:

Proposition 7.2.12 (Single node Inputs)**.** *Let $n[\![s]\!] \in \mathsf{sSys}$, and let Γ_1, Γ_2 be two connectivity graphs such that both $\Gamma_1 \rhd n[\![s]\!]$ and $\Gamma_2 \rhd n[\![s]\!]$ are well formed. If $(\Gamma_1 \rhd n[\![s]\!]) \overset{c.m?v}{\longrightarrow} (\Gamma_1 \rhd n[\![\Delta_n]\!])$ with $m \in \mathsf{Input}(\Gamma_1 \rhd n[\![s]\!])$ then $(\Gamma_2 \rhd n[\![s]\!]) \overset{c.l?v}{\longrightarrow} \Gamma_2 \rhd n[\![\Delta_n]\!]$ for every $l \in \mathsf{Input}(\Gamma_2 \rhd n[\![s]\!])$.*

Proof. Straightforward from the definitions of both extensional and intensional input actions. □

Corollary 7.2.13 (Single node simulations)**.** *Let $\Gamma_1 \rhd n[\![s]\!]$ be a single node network with $\mathsf{Input}(\Gamma_1 \rhd n[\![s]\!]) \neq \emptyset$, $\mathsf{Output}(\Gamma_1 \rhd n[\![s]\!]) \neq \emptyset$, and suppose $\Gamma_1 \rhd n[\![s]\!] \lhd_{sim} \Gamma_1 \rhd n[\![\Theta]\!]$; then, for any Γ_2 such that $\Gamma_2 \rhd n[\![s]\!]$ is well formed it holds $\Gamma_2 \rhd n[\![s]\!] \lhd_{sim} \Gamma_2 \rhd n[\![\Theta]\!]$.*

Proof. Follows directly from the definition of extensional and intensional actions and from Proposition 7.2.12. The constraint that $\mathsf{Int}(\Gamma_1 \rhd n[\![s]\!])$ be non-empty is needed when considering the case $(\Gamma_2 \rhd n[\![s]\!]) \overset{c.m?v}{\longrightarrow} \Gamma_2 \rhd n[\![\Theta]\!]$. □

With an abuse of notation, we write $s \lhd_{sim} \Theta$ whenever $(\Gamma \rhd n[\![s]\!]) \lhd_{sim} (\Gamma \rhd \Theta[\![s]\!])$ for any Γ such that $\mathsf{Int}(\Gamma \rhd n[\![s]\!]) \neq \emptyset$.

We are now ready to prove the main result of this section.

Proof of Theorem 7.2.4: We actually prove a more general result. Recall that a distribution Δ over sSys is called node-stable if $\text{nodes}(N) = \text{nodes}(M)$ whenever $\Delta(N) > 0$ and $\Delta(M) > 0$. For such a distribution it makes sense to define $\mathsf{Int}(\Gamma \rhd \Delta)$ to be $\mathsf{Int}(\Gamma \rhd M)$ for any M such that $\Delta(M) > 0$. Now let

Let $\mathcal{R} \subseteq \mathsf{Nets} \times \mathcal{D}(\mathsf{Nets})$ be given by

$$(\Gamma_M \rhd M) \between (\Gamma_n \rhd n[\![s]\!]) \;\; \mathcal{R} \;\; (\Gamma \rhd \Delta_1) \between (\Gamma_n \rhd n[\![\Theta_1]\!])$$

whenever

(a) Θ_1, a distribution over sSys, is node stable

(b) $\mathsf{Int}(\Gamma_M \rhd M) = \mathsf{Int}(\Gamma \rhd \Delta_1)$

(c) $(\Gamma_M \rhd M) \lhd_{sim} (\Gamma \rhd \Delta_1)$

(d) $s \lhd_{sim} \Theta_1$; here Θ_1 is a distribution over states, from the syntax in Figure 6.1.

Theorem 7.2.4 will follow if we can show that $\mathcal{R}$ is a simple simulation, in the sense of Definition 7.2.2.

The proof proceeds by considering a strong extensional action

$$(\Gamma_M \rhd M) \between (\Gamma_n \rhd n[\![s]\!]) \xrightarrow{\mu} \Delta \tag{7.1}$$

We must find a corresponding weak extensional action

$$(\Gamma \rhd \Delta_1) \between (\Gamma_n \rhd n[\![\Theta_1]\!]) \stackrel{\mu}{\Longrightarrow} \Theta$$

such that $\Delta \,\overline{\mathcal{R}}\, \Theta$.

The first step is to employ Proposition 7.2.9 so as to write $(\Gamma_M \rhd M) \between (\Gamma_n \rhd n[\![s]\!])$ in the form $(\Gamma_M \rhd M) \parallel (\Gamma_1 \rhd n[\![s]\!])$ for some network connectivity Γ_n. After this translation has been carried out, we may apply Decomposition , Proposition 7.2.10, to the action (7.1) above. There are three cases, depending on μ. We only consider the case $\mu = c!v \rhd \eta$.

Suppose then $(\Gamma_M \rhd M) \parallel (\Gamma_1 \rhd n[\![s]\!]) \xrightarrow{c!v \rhd \eta} (\Gamma_M \rhd \Delta_M) \parallel (\Gamma_1 \rhd n[\![\Theta_n]\!])$. According to Proposition 7.2.10 we have three different sub-cases to consider; again, we will consider only the most interesting one, namely

$$\begin{array}{rcl} (\Gamma_M \rhd M) & \xrightarrow{c!v \rhd \eta'} & (\Gamma_M \rhd \Delta_M) \\ (\Gamma_1 \rhd n[\![s]\!]) & \xrightarrow{c.m?v} & \Gamma_1 \rhd n[\![\Theta_n]\!] \\ \eta' = \eta \cup \{n\} & & m \in \mathsf{Int}(\Gamma_1 \rhd n[\![\Theta_n]\!]) \end{array}$$

We have that $(\Gamma_M \rhd M) \lhd_{sim} (\Gamma \rhd \Delta_1)$ by hypothesis, so that $(\Gamma \rhd \Delta_1) \stackrel{c!v \rhd \eta'}{\Longrightarrow} (\Gamma \rhd \Delta_2)$ with $(\Gamma_M \rhd \Delta_M) \,\overline{\lhd_{sim}}\, (\Gamma \rhd \Delta_2)$. We can now rewrite $(\Gamma \rhd \Delta_1) \between (\Gamma_n \rhd n[\![\Theta_1]\!])$ as $(\Gamma \rhd \Delta_1) \parallel (\Gamma_2 \rhd n[\![\Theta_1]\!])$.

Notice also that, since $n \in \eta'$, there exists a node m in nodes(M) such that $\Gamma_M \vdash n \leftrightarrow m$. By the definition of $\parallel$, we obtain therefore that $m \in \mathsf{Int}(\Gamma_1 \rhd n[\![s]\!])$. Thus, we can apply both Proposition 7.2.12 and Corollary 7.2.13 to infer $(\Gamma_2 \rhd n[\![\Theta_1]\!]) \stackrel{c.l?v}{\Longrightarrow} (\Gamma_2 \rhd n[\![\Theta_2]\!])$ for some $l \in \mathsf{Int}(\Gamma_2 \rhd n[\![\Theta_n]\!])$, and $\Theta_n \,\overline{\lhd_{sim}}\, \Theta_2$.

Thus we have proved

$$\begin{array}{rcl} \Gamma \rhd \Delta_1 & \stackrel{c!v \rhd \eta'}{\Longrightarrow} & \Gamma \rhd \Delta_2, \\ \Gamma_2 \rhd n[\![\Theta_1]\!] & \stackrel{c.l?v}{\Longrightarrow} & \Gamma_2 \rhd n[\![\Theta_2]\!], l \in \mathsf{Int}(\Gamma_2 \rhd n[\![\Theta_2]\!]), \\ \Gamma_M \rhd \Delta_M & \overline{\lhd_{sim}} & \Gamma \rhd \Delta_2, \\ \Theta_n & \overline{\lhd_{sim}} & \Theta_2. \end{array}$$

The first two results can be used together with Decomposition to prove $\Gamma \rhd \Delta_1 \parallel \Gamma_2 \rhd n[\![\Theta_n]\!] \stackrel{c!v \rhd \eta}{\Longrightarrow} \Gamma \rhd \Delta_2 \parallel \Gamma_2 \rhd n[\![\Theta_2]\!]$, while the last two allow us to infer $\Gamma_M \rhd \Delta_M \parallel \Gamma_1 \rhd n[\![\Theta_n]\!] \,\overline{\mathcal{R}}\, \Gamma_M \rhd \Delta_2 \parallel \Gamma_2 \rhd n[\![\Theta_2]\!]$. □

7.3 Simulation Preorder Fails to be Complete

Although the simulation preorder $\lhd_{sim}$ provides a proof methodology for establishing that two networks are related via the testing preorder $\sqsubseteq_{\text{may}}$, it is not complete. That is, it is possible to find two networks $\mathcal{M}, \mathcal{N}$ such that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ holds, but $\mathcal{M}$ cannot be simulated by $\mathcal{N}$. This is quite surprising, as simulation preorder has been already proved to provide a characterisation of the may-testing preorder for more standard process calculi, such as pCSP [19].

However, in our setting a problem arises; the mathematical basis of simulation preorders rely on (full) probability distributions, which are a suitable tool in a framework where a weak action from a process term has

Figure 7.3: Two testing related networks

to be matched with the same action performed by a distribution of processes.

This is not true in our calculus; we have already shown that, due to the presence of local broadcast communication, it is possible to match a weak broadcast action with a sequence of outputs whose sets of target nodes are pairwise disjoint. This behaviour has been formalised by giving a non-standard definition of weak extensional actions in Definition 7.1.2.

Such a definition captures the possibility of simulating a broadcast through a multicast only when the former action is performed with probability 1.

However, when comparing distributions of networks we have to also match actions which are performed with probabilities less than 1, at least informally; here the simulation of broadcast using multicast runs into problems, as the following example shows.

Example 7.3.1. Consider the two network distributions $\Gamma_M \rhd \Delta$, $\Gamma_N \rhd \Theta$ depicted in Figure 7.3; let

$$\begin{aligned} \Delta &= 0.81 \cdot m[\![c!\langle v\rangle.\mathbf{0}]\!] + 0.19 \cdot m[\![\mathbf{0}]\!] \\ \Theta &= 0.9 \cdot m[\![c!\langle v\rangle.\mathbf{0}]\!] \,|\, n[\![P]\!] + 0.1 \cdot m[\![\mathbf{0}]\!] \,|\, n[\![P]\!] \end{aligned}$$

where P is the process $c?(x).(c!\langle x\rangle.\mathbf{0}_{0.9}\oplus \mathbf{0}) + c?(x).P$. In $\Gamma_M \rhd \Delta$ a message is broadcast to nodes o_1, o_2 with probability 0.81, while in $\Gamma_N \rhd \Theta$ two different broadcasts happen in sequence (first to node o_1, then to o_2). Each of these broadcasts happens with probability 0.9, so that the overall probability of message v to be detected by both nodes o_1, o_2 is again 0.81.

We first show that $\Gamma_M \rhd \Delta \sqsubseteq_{\text{may}} \Gamma_N \rhd \Theta$, then we prove that $\Gamma_M \rhd \Delta \not\lhd_{sim} \Gamma_N \rhd \Theta$.
For the first statement, we only supply informal details, as a complete proof would be rather long and technical. Consider a test distribution $\Gamma_T \rhd \Lambda$, such that both $\Gamma_M \rhd \Delta \between \Gamma_T \rhd \Lambda$ and $\Gamma_N \rhd \Theta \between \Gamma_T \rhd \Lambda$ are defined. Without loss of generality, suppose that both $o_1, o_2 \in \text{nodes}(\Gamma_T \rhd \Lambda)$, thus every $T \in \lceil \Lambda \rceil$ can be written in $o_1[\![t_1]\!] \,|\, o_2[\![t_2]\!] \,|\, T$. Also, we provide details only for the most interesting case, that is when the testing component reaches (with some probability p) an ω-successful configuration after network $\mathcal{M}$ broadcasts the message v. In this case, a computation fragment of $\Gamma_M \rhd \Delta \between \Gamma_T \rhd \Lambda$ can be summarised as follows:

1. The testing component $\Gamma_T \rhd \Lambda$ performs some internal activity, thus leading to $\Gamma_T \rhd \Lambda \stackrel{\tau}{\Longrightarrow} \Gamma_T \rhd o_1[\![\Lambda_1]\!] \,|\, o_2[\![\Lambda_2]\!] \,|\, \Lambda_T$

2. At this point, the distribution Δ will broadcast the message with probability 0.81, causing the testing component to evolve in $\Gamma_T \rhd o_1[\![\Lambda_1']\!] \,|\, o_2[\![\Lambda_2']\!] \,|\, \Lambda_T$[1]. The tested component, at this point, will be in a deadlocked configuration, that is it cannot perform any action.

Consider now the distribution $\Gamma_N \rhd \Theta \between \Gamma_T \rhd \Lambda$. For such a network, a matching computation will proceed as follows:

[1] Note that only nodes o_1 and o_2 are affected by the broadcast performed by node m

1. The testing component $\Gamma_T \rhd \Lambda$ performs the same sequence of internal activities as before, thus it will end up in the distribution $\Gamma_T \rhd o_1[\![\Lambda_1]\!] \mid o_2[\![\Lambda_2]\!] \mid \Lambda_T$.

2. At this point, message v will be broadcast by Θ to node o_1. This happens with probability 0.9, and it causes the testing network to evolve in $\Gamma_T \rhd o_1[\![\Lambda_1']\!] \mid o_2[\![\Lambda_2]\!] \mid \Theta_T$. Here note that, since the broadcast can not be heard by node o_2, the probability distribution for such a node in the testing component has not been affected.

3. Before allowing the testing component to perform any activity, we require the tested network to perform the second broadcast, which will be heard by node o_2; again, this will happen with probability 0.9 and it will not affect the probability distribution of processes running at node o_1. Thus, after the second message has been broadcast by the tested network, the testing component will have the form $\Gamma_T \rhd o_1[\![\Lambda_1']\!] \mid o_2[\![\Lambda_2']\!] \mid \Lambda_T$, which is exactly the same configuration obtained in the first experiment, after $\Gamma_M \rhd \Delta$ has broadcast the message to both locations. Further, note that the overall probability of Θ delivering both messages is again 0.81, and that the tested network is now in a deadlocked configuration.

Thus we have shown that, whenever the broadcast of message v by $\Gamma_M \rhd \Delta$ affects the testing network $\Gamma_T \rhd \Lambda$ in some way, then $\Gamma_N \rhd \Theta$ is able to multicast the message to both o_1 and o_2, causing $\Gamma_T \rhd \Lambda$ to behave in the same way. Note also that in $\Gamma_N \rhd \Theta$ we introduced a non-deterministic choice in process P; this choice has been introduced because it is possible for node n to receive messages from the external node o_2. Since in $\Gamma_M \rhd \Delta$ messages received from external nodes do not affect the behaviour of the network, we require $\Gamma_N \rhd \Theta$ to have at least an extreme derivative in which this policy is respected. In fact, when $\Gamma_N \rhd \Theta$ receives a message from o_2, code P running at node n can decide to ignore the message by evolving to the process P itself. At this point, the reader should be convinced that $\Gamma_M \rhd \Delta \sqsubseteq_{\text{may}} \Gamma_N \rhd \Theta$.

Now we show that it is the case that $\Gamma_M \rhd \Delta$ cannot be simulated by $\Gamma_N \rhd \Theta$. The proof is obtained by contradiction. Suppose then that $\Gamma_M \rhd \Delta \,\overline{\lhd_{sim}}\, \Gamma_N \rhd \Theta$. Let M_1 be the system term $m[\![c!\langle v\rangle]\!]$. As $\Gamma_M \rhd M_1 \overset{c!v\rhd\{o_1,o_2\}}{\longrightarrow}$, and $\Delta = 0.81 \cdot \overline{\Gamma_M \rhd M_1} + 0.19 \cdot \overline{\Gamma_M \rhd m[\![\mathbf{0}]\!]}$, we can rewrite Θ as

$$\Theta = 0.81 \cdot \Theta_1 + 0.19 \cdot \Theta_2$$

such that $\Theta_1 \overset{c!v\rhd\{o_1,o_2\}}{\Longrightarrow}$. Let now N_1 and N_2 be the state based terms $m[\![c!\langle v\rangle]\!] \mid n[\![P]\!]$, and $m[\![\mathbf{0}]\!] \mid n[\![P]\!]$, respectively. These terms have been defined so that $\lceil \Gamma_N \rhd \Theta \rceil = \{\Gamma_N \rhd N_1, \Gamma_N \rhd N_2\}$. Since $\Gamma_N \rhd N_2$ is a deadlocked network (hence it cannot perform any output action), the only network in the support of Θ_1 has to be $\Gamma_N \rhd N_1$, for a distribution can perform an action only if all the networks in its support can perform the same action.

It is easy to show that the only possible action for $\Gamma_N \rhd N_1$ is $\Gamma_N \rhd N_1 \overset{c!v\rhd\{o_1\}}{\Longrightarrow} \Gamma_N \rhd \Theta''$, where $\Theta'' = \overline{m[\![\mathbf{0}]\!]} \mid \Theta_N$ and $\Theta_N = 0.9 \cdot \overline{n[\![c!\langle v\rangle]\!]} + 0.1 \cdot \overline{n[\![\mathbf{0}]\!]}$. Since the latter is a deadlock state, we can conclude that the action $\Gamma_N \rhd \Theta'' \overset{c!v\rhd\{o_2\}}{\Longrightarrow}$ is not possible, so neither is $\Gamma_N \rhd N_1 \overset{c!v\rhd\{o_1,o_2\}}{\Longrightarrow}$. It follows that the broadcast action performed by $\Gamma_M \rhd M_1$ cannot be matched by $\Gamma_N \rhd N_1$, and hence $\Gamma_M \rhd \Delta \,\overline{\not\lhd_{sim}}\, \Gamma_N \rhd \Theta$. □

Remark 7.3.2 (About incompleteness)**.** We end this section by discussing, at least informally, why our simulation preorder fails to be complete.

The main problem lies in the non-standard definition of weak extensional actions. Intuitively, a test can be defined to detect whether a message has been multicast to a set of output nodes η with a given probability p. However, our current definition of weak extensional outputs does not capture multicasts performed with an overall probability p. In fact, whenever $\Delta \overset{c!v\rhd(\eta_1\cup\eta_2)}{\models\!\Longrightarrow} \Theta$ because $\Delta \overset{c!v\rhd\eta_1}{\models\!\Longrightarrow} \Theta', \Theta' \overset{c!v\rhd\eta_2}{\models\!\Longrightarrow} \Theta$ and $\eta_1 \cap \eta_2 = \emptyset$, note that we require the whole distribution Θ' to perform an extensional output action. In other words, we only capture the broadcast which can be detected to the notes in η_2 if it happens with probability 1. Clearly, this is not the only kind of multicast that we can test by placing code at the interface nodes of a network; as we have pointed out above, we can define a test to check whether nodes in the set η_2 detect the broadcast of value v along channel c with probability less than one. As a consequence, our proof technique turns out not to be complete. □

7.4 Soundness for Must-testing

In this Section we turn our attention to achieving soundness for the must-testing preorder. We only state the main results, without providing any proof. However, the proof of the following Theorem can be performed by the interested reader by following the one contained in [19], Section 8.

We have already shown in Chapter 4 that, in a non-probabilistic setting, acceptance sets cannot be used to characterise the (non-probabilistic) must testing preorder; this is because of the non-blocking nature of output extensional transitions, and because since we are only comparing networks with the same input interface[2] we cannot infer any information from input transitions. What is really important in the must-testing framework is the impossibility for a network to evolve autonomously.

This leads to the definition of deadlock configuration, which can be extended to probability (sub)distributions. We say that a network $\mathcal{M}$ is *deadlocked*, denoted $\mathcal{M} \not\rightarrow$ whenever $\omega(\mathcal{M}) = \mathsf{false}$ and $\mathcal{M} \stackrel{\tau}{\longrightarrow}, \mathcal{M} \stackrel{c!v \triangleright \eta}{\longrightarrow}$ for any c, v, η. A sub-distribution Δ over $\mathcal{D}_{sub}(\mathsf{Nets})$ is *deadlocked* if any network in its support is deadlocked.

Then we can define another kind of simulation in which the possibility for a sub-distribution being deadlocked is taken into account. For reasons explained in [19] it is more straightforward to express this form of simulation as a relation from sub-distributions to sub-distributions.

Definition 7.4.1 (Deadlock Simulations)**.** In $\mathsf{pLTS}_{\mathsf{Nets}}$ we let $\sqsupseteq_{DS}$ denote the largest relation in $\mathcal{D}_{sub}(\mathsf{Nets}) \times \mathcal{D}_{sub}(\mathsf{Nets})$ such that if $\Delta \sqsupseteq_{DS} \Theta$ then:

- $\Delta \stackrel{\mu}{\Longrightarrow} \sum_{i \in I}(p_i \cdot \Delta'_i)$, with I being an index set such that $\sum_{i \in I} p_i \leq 1$, whenever there are $\Theta'_i \in \mathcal{D}_{sub}(\mathsf{Nets})$ such that $\Theta \stackrel{\mu}{\longrightarrow} \sum_{i \in I}(p_i \cdot \Theta'_i)$ and, for any $i \in I$, $\Delta'_i \sqsupseteq_{DS} \Theta'_i$
- $\Delta \Longrightarrow \not\rightarrow$ whenever $\Theta \Longrightarrow \not\rightarrow$. □

As it is not surprising, deadlock simulations are sound with respect to the must-testing preorder.

Theorem 7.4.2 (Soundness for Must-testing)**.** Let $\mathcal{M}, \mathcal{N}$ be finitary networks. If $\overline{\mathcal{M}} \sqsupseteq_{DS} \overline{\mathcal{N}}$ then $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$. □

Remark 7.4.3. Note that the characterisation result in the non-probabilistic setting, Theorem 4.4.11 is valid only for strongly convergent networks. This constraint is not necessary in the probabilistic setting.

The reason why we needed to focus on strongly convergent networks in Chapter 4 was because the non-blocking nature of output transitions leads to different kinds of divergence. Specifically, a network which immediately diverges (that is, divergence is obtained via an infinite sequence of τ-actions) can be distinguished by one in which divergence can be obtained only after a broadcast transition.

For any networks $\mathcal{M}, \mathcal{N}$, in the non-probabilistic setting, we have that if $\mathcal{M}$ diverges immediately (that is, it has an infinite sequence of τ-transitions) and $\mathcal{M} \sqsubseteq_{\text{must}} \mathcal{N}$ then $\mathcal{N}$ diverges immediately. The same applies in the probabilistic framework, where a sub-distribution Δ diverges immediately if $\Delta \Longrightarrow \varepsilon$; we recall that ε is the empty sub-distribution.

In Definition 7.4.1 we have that if $\mathcal{M} \sqsupseteq_{DS} \overline{\mathcal{N}}$ and $\mathcal{M} \Longrightarrow \varepsilon$, then $\mathcal{N} \Longrightarrow \varepsilon$. That is, deadlock simulations capture the notion of immediate divergence; this is not true for the predicate $\mathcal{M} \Uparrow$ defined in Section 4.4, which captured all possible kinds of divergence.

[2] Recall that any network $\mathcal{M}$ can perform an extensional input transition of the form $n.c?v$, provided $n \in \mathsf{Input}(\mathcal{M})$.

Chapter 8

Applications for Probabilistic Wireless Networks

In this Chapter we provide some applications of our theory; first, we focus on a simple routing model to show how networks can be related via the may testing preorder by providing a simulation between them. In this model we require routing of messages to be sequential; that is, at any given time the network is routing at most one message. We decided to focus on a simple model of routing in order to show how simulations can be used to prove whether two networks are may-testing related. We first define a specification (or model) for routing in terms of a network $\mathcal{M}$; then we consider a more complicated network $\mathcal{N}$ and we show that it is may testing related to our model by exhibiting a simple simulation between $\mathcal{M}$ and $\mathcal{N}$; by Theorem 7.2.1, it follows that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$. Finally, we generalize our result by focusing on a network $\mathcal{L}$ which is only partially defined; again, we prove that $\mathcal{M}$ and $\mathcal{L}$ are testing related by showing that $\mathcal{M} \lhd^{s} \mathcal{L}$.

Then we focus on more practical applications; we consider the connectionless routing model and defined in Chapter 5, and we exhibit a probabilistic implementation for it. We only consider the may-testing preorder, though the reader can check that in both cases the proposed implementation is must-equivalent to the model.

8.1 Probabilistic Sequential Routing

8.1.1 The Specification

Routing is the central task that has to be accomplished in the network layer of (wireless) network protocols [65]. The goal of a routing protocol is that of guaranteeing that a message, generated by a node of the network and intended for a second, flows through the nodes of the network to eventually reach the desired destination. The design of routing protocols relies on the assumption that the communication between two nodes is perfect; in practice, this task is accomplished by the Datalink and MAC layers, while in our calculus this is guaranteed by the intensional semantics that define network transitions.

Here we propose a basic network to model routing; as we will see, the way it is defined ensures it enjoys the following features:

- Two external nodes o_1, o_2, are used as the endpoints of a communication; a message generated by node o_1 has to be forwarded to o_2, and vice-versa. The constraint that o_1, o_2 are external nodes guarantees that messages are generated non-deterministically,
- The network detects the messages generated by the external nodes only along a single channel; all the messages generated via other channels are ignored,
- Routing is sequential; that is, only one packet at time can be routed.

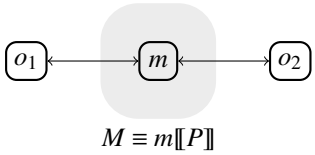


Figure 8.1: A model for routing

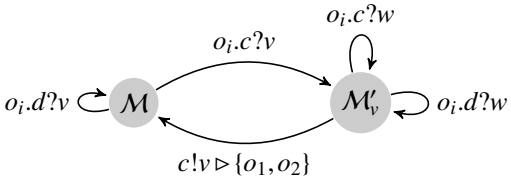


Figure 8.2: the pLTS induced by our routing model. Here d is an arbitrary channel different from c, v and w are arbitrary messages and i ranges over $1, 2$.

Our model consists in the network $\mathcal{M} = \Gamma \triangleright M$ depicted in Figure 8.1. We define P to be the process $c?(x).c!\langle x\rangle.P$. The role of the internal node is that of repeatedly listening for incoming messages (either from o_1 or o_2); once a message has been received, it will forward it to the destination by performing a broadcast. In this case the message is heard by both the external nodes o_1 and o_2, thus we ensure that it will reach the destination node. This is needed because it is impossible for node m, upon receiving a message, to detect if it was originally sent by o_1 or o_2. Further, it is easy to check that the network $\mathcal{M}$ satisfies the constraints above. Finally, in order to ensure that the pLTS generated by network $\mathcal{M}$ is finitary, we assume that the sets of both channels and values are finite. This constraint is needed only because simulations are sound with respect to the $\sqsubseteq_{\text{may}}$ preorder in a finitary setting. As we will see, only messages received along channel c from some external node will be able to affect the behaviour of a network; therefore, the results that we prove are valid also in the case the set of channels and values are assumed to be infinite.

The pLTS induced by network $\mathcal{M}$ is depicted in Figure 8.2. Below we summarise the actions that network $\mathcal{M}$ can perform:

1. $\mathcal{M} \xrightarrow{d.o_i?v} \overline{\mathcal{M}}$, provided channel d is different from c, $i = 1, 2$,
2. $\mathcal{M} \xrightarrow{c.o_i?v} \overline{\mathcal{M}'_v}$, $i = 1, 2$, where $\mathcal{M}'_v = \Gamma \triangleright m[\![c!\langle v\rangle.P]\!]$.

We also list the possible actions that can be performed by the derivative of $\mathcal{M}$, $\mathcal{M}'_v$.

1. $\mathcal{M}'_v \xrightarrow{d.o_i?v} \overline{\mathcal{M}'}$, where d is an arbitrary channel (including c),
2. $\mathcal{M}'_v \xrightarrow{c!v \triangleright \eta} \overline{\mathcal{M}}$, where $\eta = \{o_1, o_2\}$.

8.1.2 A Simple Implementation

Now that we have provided a specification for routing, let us look at a possible implementation. In our framework, for implementation of $\mathcal{M}$ we mean a network $\mathcal{N} = \Gamma_N \triangleright N$ such that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$. The main idea here is to build $\mathcal{N}$ by replacing node m in $\mathcal{M}$ with a rather simple network, consisting of different nodes. The main goal we want to achieve for $\mathcal{N}$ consists in routing a message generated from o_1 to o_2, and vice versa. To this end, we design $\mathcal{N}$ to have at least one computation in which the constraints imposed for $\mathcal{M}$ are satisfied, thereby ensuring that $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$ will be true.

The network $\mathcal{N}$ we consider is depicted in Figure 8.3. Here N is defined to be

$$m_1[\![P_m]\!] \,|\, m_2[\![P_m]\!] \,|\, \prod_{i=1}^{4} n_i[\![P_i]\!]$$

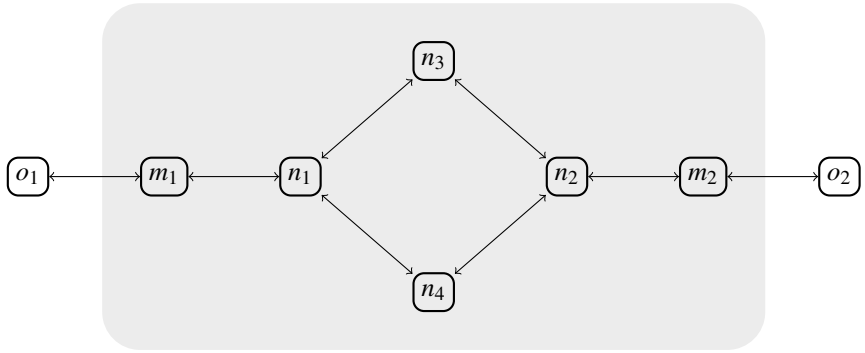


Figure 8.3: A simple implementation of $\mathcal{M}$

where

$$\begin{array}{rcl} P_m & = & c?(x).c!\langle x\rangle.P_m + c?(x).P_m \\ P_i & = & c?(x).[(c_3!\langle x\rangle.P_i \,{}_{\frac{1}{2}}\!\oplus c_4!\langle x\rangle.P_i)] + c?(x).P_i \\ & + & c_i?(x).[(c_3!\langle x\rangle.P_i \,{}_{\frac{1}{2}}\!\oplus c_4!\langle x\rangle.P_i)] + c_i?(x).c!\langle x\rangle.P_i,\ \ i=1,2 \\ P_i & = & c_i?(x).(c_1!\langle x\rangle \,{}_{\frac{1}{2}}\!\oplus c_2!\langle x\rangle)\ \ i=3,4 \end{array} \tag{8.1}$$

Let us discuss the intuitive behaviour of each node in network $\mathcal{N}$. The idea is that of implementing probabilistic routing; once a message is received by a node in the network, it will perform a probabilistic choice to select a node, among its neighbours, to which the message will be forwarded. For this purpose, each internal node $n_i, i = 1,\cdots,4$, has a channel c_i associated to it; the code of network $\mathcal{N}$ is designed so that each of these internal nodes n_i waits for a message to be received along its associated channel c_i. Further, it is the only node in the network which can receive messages along this channel; this ensures that whenever a message is broadcast along channel c_i only node n_i is able to actually receive it.

The behaviour of an internal nodes n_3 or n_4 is straightforward; if it receives a message, which may come from either of its neighbours, it selects according to a fair probabilistic choice one of these neighbours to whom the message is forwarded.

The behaviour of n_1, n_2 is more complicated. We describe that of n_1; the behaviour of n_2 is symmetric. If it receives a message along channel c_1, its associated channel, we know that it must come from one of its internal neighbours n_3 or n_4. Non-deterministically, the message is either

- forwarded to the externally connected node m_1 using the channel c
- or using a fair probabilistic choice it is rebroadcast back to one of its internal neighbours, along their associated channels c_j.

But n_1 can also listen to messages broadcast along channel c; this allows it to receive messages from the node m_1, which in turn is connected to the interface. When such a message is received non-deterministically it is either

- ignored
- or forwarded to one if the internal nodes n_3, n_4, using their associated channels c_j; the destination node is selected randomly, using a fair probabilistic choice.

The nodes m_1, m_2, being connected to the interface, are responsible for routing messages between the external nodes o_1, o_2 and the internal ones n_1, n_2 respectively. This is achieved by m_i forwarding every message it receives along channel c; we have already seen that its neighbour n_i is both broadcasting and listening on c. However, node m_1 can also decide non-deterministically to discard any of these messages.

Note that the behaviour of network $\mathcal{N}$ is decidedly more complicated of the routing model $\mathcal{M}$. For example, it is possible in $\mathcal{N}$ that the task of routing a message fails before being completed; as nodes m_i, n_i, $i = 1,2$ can non-deterministically ignore messages received along channel c, a message can be lost at one of these nodes before the routing activity is completed.

Despite having a more complicated behaviour, network $\mathcal{N}$ simulates the routing model $\mathcal{M}$. However this simulation is far from being trivial. For example in $\mathcal{M}$ when a message is received from o_1 it is broadcast simultaneously to both nodes in the interface o_1 and o_2. But in $\mathcal{N}$ there is no node directly connected to both o_1 and o_2 and so this behaviour can not be replicated. Moreover all communication between the individual nodes in $\mathcal{N}$ is probabilistic, so that the multi-cast which simulates this broadcast is only achieved in the *probabilistic limit*.

We show that $\mathcal{M} \lhd^{s} \mathcal{N}$. From Theorem 7.2.3 it will follow that that $\mathcal{M} \lhd_{sim} \mathcal{N}$, and therefore by Soundness, Theorem 7.2.1 we will have $\mathcal{M} \sqsubseteq_{\text{may}} \mathcal{N}$.

In order to show that $\mathcal{M} \lhd^{s} \mathcal{N}$, we have to exhibit a simulation between them; this is facilitated by introducing some suitable notation. We define the system term

$N_{m_1} = m_2[\![P_m]\!] \mid \prod_{i=1}^{4} n_i[\![P_i]\!]$. That is, N_{m_1} is the term obtained by removing from N the code from node m_1. Similar definitions apply for each node in nodes(N). For any message v, let $\mathcal{N}_v^1 = \Gamma_N \rhd m_1[\![c!\langle v\rangle.P_m]\!] \mid N_{m_1}$, $\mathcal{N}_v^2 = \Gamma_N \rhd m_2[\![c!\langle v\rangle.P_m]\!] \mid N_{m_2}$.

Now we show that the relation

$$\mathcal{S} = \{(\mathcal{M}, \overline{\mathcal{N}})\} \cup \{(\mathcal{M}'_v, \overline{\mathcal{N}_v^1}), (\mathcal{M}'_v, \overline{\mathcal{N}_v^2}) \mid v \in \mathcal{V}\}$$

satisfies the requirements of Definition 7.2.2.

Let us first look at the pair $(\mathcal{M}, \overline{\mathcal{N}})$. Recall that network $\mathcal{M}$ has only four possible actions (up to the choice of a message v):

- $\mathcal{M} \xrightarrow{d.o_1?v} \overline{\mathcal{M}}$, where $d \neq c$. We need to match this action with a derivation of the form $\overline{\mathcal{N}} \stackrel{d.o_1?v}{\Longrightarrow} \Theta$ for some Θ such that $(\overline{\mathcal{M}}, \Theta) \in \overline{\mathcal{S}}$. It is not difficult to note that $\overline{\mathcal{N}} \xrightarrow{d.o_1?v} \overline{\mathcal{N}}$, as none of the nodes m_1 and m_2 (which are the only one which can detect messages broadcast from external nodes) is waiting to receive a value on channel d. By Definition 6.1.2 we have $(\overline{\mathcal{M}}, \overline{\mathcal{N}}) \in \overline{\mathcal{S}}$.
- $\mathcal{M} \xrightarrow{d.o_2?v} \overline{\mathcal{M}}$, where $d \neq c$. This case is analogous to the one above
- $\mathcal{M} \xrightarrow{c.o_1?v} \overline{\mathcal{M}'_v}$. In this case it is easy to show that $\overline{\mathcal{N}} \xrightarrow{c.o_1?v} \overline{\mathcal{N}_v^1}$, and $(\overline{\mathcal{M}'_v}, \overline{\mathcal{N}_v^1}) \in \overline{\mathcal{S}}$.
- $\mathcal{M} \xrightarrow{c.o_2?v} \overline{\mathcal{M}'_v}$. As above, one can check that $\overline{\mathcal{N}} \xrightarrow{c.o_1?v} \overline{\mathcal{N}_v^2}$, and $(\overline{\mathcal{M}'_v}, \mathcal{N}_v^2) \in \overline{\mathcal{S}}$.

It remains to check the pairs of the form $(\mathcal{M}'_v, \overline{\mathcal{N}_v^1})$ and $(\mathcal{M}'_v, \overline{\mathcal{N}_v^2})$. We only supply the details for the former case, as the latter one is analogous.

- $\mathcal{M}'_v \xrightarrow{d.o_1?v} \overline{\mathcal{M}'_v}$, where d is an arbitrary channel, including c. Note that in $\mathcal{N}_v^1$ the node m_1 is not waiting to receive a message along any channel. That is, we have $\mathcal{N}_v^1 \xrightarrow{d.o_1?v} \overline{\mathcal{N}_v^1}$, and $(\overline{\mathcal{M}_v^1}, \overline{\mathcal{N}_v^1}) \in \overline{\mathcal{S}}$
- $\mathcal{M}'_v \xrightarrow{d.o_2?v} \overline{\mathcal{M}'_v}$, where d is an arbitrary channel, included c. If $d \neq c$, then this case is similar to the above one. If $d = c$, note that node m_2 is waiting to receive a message along channel c. However, we already remarked that node m_2 can non-deterministically choose to ignore messages broadcast along channel c, so that it is easy to derive $\mathcal{N}_v^1 \xrightarrow{c.o_2?v} \overline{\mathcal{N}_v^1}$. Now it suffices to note that $(\overline{\mathcal{M}_v^1}, \overline{\mathcal{N}_v^1}) \in \mathcal{S}$.
- $\mathcal{M}'_v \xrightarrow{c!v \rhd \{o_1,o_2\}} \overline{\mathcal{M}}$. This is the most interesting case. In fact, it is not possible to match the strong extensional output performed by $\mathcal{M}'_v$ directly. Rather, we exhibit a weak derivation of the form $\mathcal{N}_v^1 \stackrel{c!v \rhd \{o_1,o_2\}}{\Longrightarrow} \overline{\mathcal{N}}$. This is obtained by exploiting the non-standard definition of weak extensional outputs, given in Definition 7.1.2(3). Specifically, we show that that there exist Δ_1 and Δ_2 such that

$$\mathcal{N}_v^1 \xrightarrow{c!v \rhd \{o_1\}} \Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_2 \xrightarrow{\tau} \overline{\mathcal{N}_v^2} \xrightarrow{c!v \rhd \{o_2\}} \overline{\mathcal{N}}. \tag{8.2}$$

At this point, since $(\overline{\mathcal{M}}, \overline{\mathcal{N}}) \in \overline{\mathcal{S}}$ the proof is finished.

In order to provide the sequence of derivations (8.2) above, we use

$$\begin{aligned}\Delta_1 &= \frac{1}{2}\cdot\overline{\Gamma_N \rhd n_1[\![c_3!\langle v\rangle.P_1]\!]\,|\,N_{n_1}} + \frac{1}{2}\cdot\overline{\Gamma_N \rhd n_1[\![c_4!\langle v\rangle.P_1]\!]\,|\,N_{n_1}},\\ \Delta_2 &= \overline{\Gamma_N \rhd n_2[\![c!\langle v\rangle.P_2]\!]\,|\,N_{n_2}}\end{aligned}$$

The transitions $\mathcal{N}_v^1 \xrightarrow{c!v\rhd\{o_1\}} \Delta_1$, $\Delta_2 \xrightarrow{\tau} \overline{\mathcal{N}_v^2}$ and $\mathcal{N}_v^2 \xrightarrow{c!v\rhd\{o_2\}} \overline{\mathcal{N}}$ are easy to derive. In the latter, to obtain the pointed distribution $\overline{\mathcal{N}}$ as the result of the transition, we exploited the ability of node n_2 to ignore messages received along channel c. The only difficulty lies in exhibiting the hyper-derivation $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_2$.

First, note that each network in the support of Δ_1 can perform a τ-action. Specifically, we have $\Gamma_N \rhd n_1[\![c_3!\langle v\rangle.P_1]\!]\,|\,N_{n_1} \xrightarrow{\tau} \Gamma_N \rhd \Delta_3$, where

$$\Delta_3 = \frac{1}{2}\cdot\overline{\Gamma_N \rhd n_3[\![c_1!\langle v\rangle.P_3]\!]\,|\,N_{n_3}} + \frac{1}{2}\cdot\overline{\Gamma_N \rhd n_1[\![c_2!\langle v\rangle.P_3]\!]\,|\,N_{n_3}}$$

and $\Gamma_N \rhd n_1[\![c_4!\langle v\rangle.P_1]\!]\,|\,N_{n_1} \xrightarrow{\tau} \Gamma_N \rhd \Delta_4$, where

$$\Delta_4 = \frac{1}{2}\overline{\Gamma_N \rhd n_4[\![c_1!\langle v\rangle.P_4]\!]\,|\,N_{n_4}} + \frac{1}{2}\overline{\Gamma_N \rhd n_4[\![c_2!\langle v\rangle.P_4]\!]\,|\,N_{n_4}}.$$

The last two derivations ensure that $\Delta_1 \xrightarrow{\tau} \frac{1}{2}\cdot\Delta_3 + \frac{1}{2}\cdot\Delta_4$. In a similar way, we can derive the following τ transitions for Δ_3 and Δ_4:

1. $\Delta_3 \xrightarrow{\tau} \frac{1}{2}\cdot\Delta_1 + \frac{1}{2}\cdot\Delta_2$,
2. $\Delta_4 \xrightarrow{\tau} \frac{1}{2}\cdot\Delta_1 + \frac{1}{2}\cdot\Delta_2$.

Putting together these derivations, we obtain the hyper-derivation

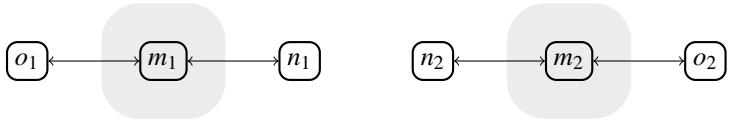
$$\begin{array}{lcll}
\Delta_1 & \xrightarrow{\tau} & \frac{1}{2}\cdot\Delta_3 + \frac{1}{2}\cdot\Delta_4 & + \quad \varepsilon \\
\frac{1}{2}\cdot\Delta_3 + \frac{1}{2}\cdot\Delta_4 & \xrightarrow{\tau} & \frac{1}{2}\cdot\Delta_1 & + \quad \frac{1}{2}\cdot\Delta_2 \\
\frac{1}{2}\cdot\Delta_1 & \xrightarrow{\tau} & \frac{1}{4}\cdot\Delta_3 + \frac{1}{4}\cdot\Delta_4 & + \quad \varepsilon \\
\frac{1}{4}\cdot\Delta_3 + \frac{1}{4}\cdot\Delta_4 & \xrightarrow{\tau} & \frac{1}{4}\cdot\Delta_1 & + \quad \frac{1}{4}\cdot\Delta_2 \\
\vdots & & \vdots & \vdots \\
\frac{1}{2^n}\cdot\Delta_1 & \xrightarrow{\tau} & \frac{1}{2^{n+1}}\cdot\Delta_3 + \frac{1}{2^{n+1}}\cdot\Delta_4 & + \quad \varepsilon \\
\frac{1}{2^{n+1}}\cdot\Delta_3 + \frac{1}{2^{n+1}}\cdot\Delta_4 & \xrightarrow{\tau} & \frac{1}{2^{n+1}}\cdot\Delta_1 & + \quad \frac{1}{2^{n+1}}\cdot\Delta_2
\end{array}$$

where we recall that ε is the empty sub-distribution. Thus we have $\Delta_1 \stackrel{\tau}{\Longrightarrow} \sum_{i=1}^{\infty}\frac{1}{2^i}\cdot\Delta_2$, which is exactly Δ_2. This concludes the proof that $\mathcal{N}_v^1 \stackrel{c!v\rhd\{o_1,o_2\}}{\Longrightarrow} \overline{\mathcal{N}}$.

8.1.3 Implementation Using Paramaterised Networks

In this Section we provide another example of network $\mathcal{L}$ which implements the routing model $\mathcal{M}$. In this case rather than a single instance of an implementation, we outline a set of properties of networks, and show that any network satisfying these properties implements the routing model $\mathcal{M}$. The code for the various nodes will be fixed and so the properties all concern the connectivity allowed between them.

Formally, we split $\mathcal{L}$ in two sub-networks, $\mathcal{L}'$ and C, such that $\mathcal{L} = \mathcal{L}' \between C$. Network $\mathcal{L}'$ is completely defined, and its representation is given in Figure 8.4. Here the process P_m is the same used for nodes m_1, m_2 in network $\mathcal{N}$, defined in Section 8.1.2. In contrast, network C is specified only in terms of a list properties which we assume it satisfies. These are as follows.



$$L' \equiv m_1[\![P_m]\!] \,|\, m_2[\![P_m]\!]$$

Figure 8.4: The network $\mathcal{L}' = \Gamma'_L \rhd L'$

1. $n_1, n_2 \in \text{nodes}(\Gamma_C \rhd C)$. For the sake of simplicity, we also assume that $\text{nodes}(\Gamma_C \rhd C) = (\Gamma_C)_V = \{n_1, \cdots, n_k\}$ for some $k > 2$. Note that if we assume $k = 2$, then the next constraint will force the connectivity graph Γ_C to have a connection between n_1, n_2. However, the same constraints specifically allow such a connection not to be included in Γ_C.

2. The connectivity graph Γ_C contains a single connected component. Further, we assume that $\Gamma_C \vdash n_1 \nleftrightarrow n_2$.

3. Every node n_i, $i = 1, \cdots, k$ is associated with a channel c_i and a probability distribution $\Lambda_i : \{1, \cdots, k\} \to [0, 1]$. The latter are defined so that $\lceil \Lambda_i \rceil = \{j \mid \Gamma_C \vdash n_i \leftrightarrow n_j\}$, for any $i = 1, \cdots, k$.

4. $C = \prod_{i \in I} n_i[\![P_i]\!]$, where

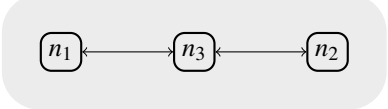
$$\begin{aligned}
P_i &= c?(x).\left[\bigoplus_{j=1}^{k} \Lambda_i(j) \cdot c_j!\langle x\rangle.P_i\right] + c?(x).P_i \\
&+ c_i?(x).\left[\bigoplus_{j=1}^{k} \Lambda_i(j) \cdot c_j!\langle x\rangle.P_i\right] + c_i?(x).c!\langle x\rangle.P_n, \quad i = 1, 2 \\
P_i &= c_i?(x).(\bigoplus_{j=1}^{k} \Lambda_i(j) \cdot c_j!\langle x\rangle.P_i), \quad i > 2
\end{aligned}$$

   Here the construct $\bigoplus_{i \in I} p_i \cdot P_i$ is interpreted as the probability distribution $\mathbb{P}(\bigoplus_{i \in I} p_i \cdot P_i) = \sum_{i \in I} p_i \cdot \mathbb{P}(P_i)$, and is defined only whenever $\sum_{i \in I} p_i = 1$.

Let us comment on these requirements. Requirement (2) is needed to ensure that, in $\mathcal{L}' \nparallel C$, inputs received by node m_1 from node o_1 can be routed to the external node o_2, and vice-versa. The constraint that $\Gamma_C \vdash n_1 \nleftrightarrow n_2$ is needed to ensure that whenever node n_1 receives a message along channel c, then it has been originally broadcast by the external node o_1.

For requirement (3), note that only node n_i can listen to a message broadcast along channel c_i. As we already explained in Section 8.1.2, this allows a node n_i to select one of its neighbour n_j, $j = 1, \cdots, k$, as the next hop in a routing path by simply forwarding a message along channel c_i. This choice, by node n_i uses the probability distribution Δ_i. Intuitively, the value $\Lambda_i(j)$ corresponds to the probability for node n_i to select n_j as the next hop in a routing path.

Finally, requirement (4) simply defines the structure of the system term C. Note that, with these requirements, the network C is determined completely by the connectivity graph Γ_C and by the set of probability distributions $\{\Lambda_i : 1 \leq i \leq k\}$.



$$\Gamma'_C \rhd C'$$

Figure 8.5: A network C'

Remark 8.1.1. *We require that* $\lceil \Lambda_i \rceil = \{j : \Gamma_C \vdash n_i \leftrightarrow n_j\}$*, that is every neighbour of node* n_i *has some non-zero probability of being selected as the next hop. This is needed to ensure that inputs received from node* o_1 *in* $\mathcal{L}' \between C$ *can be routed until they eventually reach node the external node* o_2*, and vice-versa.*

In fact, suppose we drop the requirement above from those defined for network C*; consider the network* C' *of Figure 8.5. Here we assume that the system term* C' *is defined according to Requirement (4) above and by letting* $\Lambda_1 = \overline{3}$*,* $\Lambda_3 = \overline{1}$ *and* $\Lambda_2 = \overline{3}$*. It is easy to note that network* C' *satisfies the constraints listed above. However, notice that in network* $\mathcal{L}' \between C'$*, when an input is fired from node* o_1*, it cannot flow to the external node* o_2*. This is because the message will never reach node* n_2*, as node* n_3 *always selects* n_1 *as the next hop in a routing path.*

Thus $\mathcal{L}' \between C'$ *can not be an implementation of the routing model* $\mathcal{M}$*.* □

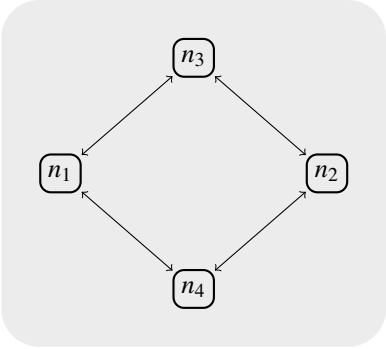


Figure 8.6: A network $\mathcal{N}'$

Remark 8.1.2. *The network* $\mathcal{N}$*, defined in Section 8.1.2, can be obtained as the network* $\mathcal{L}' \between \mathcal{N}'$*, where* $\mathcal{N}'$ *is defined in Figure 8.6, by letting*

$$\begin{aligned} \Lambda_i &= \frac{1}{2}\cdot\overline{3} + \frac{1}{2}\cdot\overline{4},\ \ i = 1,2 \\ \Lambda_i &= \frac{1}{2}\cdot\overline{1} + \frac{1}{2}\cdot\overline{2},\ \ i = 3,4. \end{aligned}$$

It is easy to note that $\mathcal{N}'$ *satisfies the constraints required by the network* C*, so that it is actually an instantiation of the paramaterised network* $\mathcal{L}$ *we are considering. Indeed many similar instances can be generated by changing the probabilities used in the code in (8.1) of Section 8.1.2 to arbitrary non-zero values.* □

Now we show that the specification for routing $\mathcal{M}$ and any network $\mathcal{L}$ satisfying the above constraints are simulation related. For the sake of clarity let $L_{m_1} = m_2[\![P_m]\!] | \prod_{i=1}^{k} n_i[\![P_i]\!]$. That is, L_{m_1} is the system term obtained by deleting node m_1 from L. Similar definitions apply for every node in nodes($\mathcal{L}$).

Let $\mathcal{L}_v^1 = \Gamma_L \rhd m_2[\![c!\langle v\rangle.P_m]\!] | L_{m_1}$, $\mathcal{L}_v^2 = \Gamma_L \rhd m_2[\![c!\langle v\rangle.P_m]\!] | L_{m_2}$. We show that the relation

$$\mathcal{S} = \{(\mathcal{M}, \overline{\mathcal{L}})\} \cup \{(\mathcal{M}'_v, \overline{\mathcal{L}_v^1}) \mid v \in \mathrm{Val}\} \cup \{(\mathcal{M}'_v, \overline{\mathcal{L}_v^2}) \mid v \in \mathrm{Val}\}$$

is a simple simulation.

Let us first look at the pair $(\mathcal{M}, \overline{\mathcal{L}})$; recall that network $\mathcal{M}$ has only four possible actions, for a given message v.

- $\mathcal{M} \xrightarrow{d.o_1?v} \overline{\mathcal{M}}$, where $d \neq c$. We need to match this action with a derivation of the form $\overline{\mathcal{L}} \stackrel{d.o_1?v}{\Longrightarrow} \Theta$ for some Θ such that $(\mathcal{M}, \Theta) \in \overline{\mathcal{S}}$. It is not difficult to note that $\overline{\mathcal{L}} \xrightarrow{d.o_1?v} \overline{\mathcal{L}}$, as none of the nodes m_1 and m_2 (which are the only one which can detect messages broadcast from external nodes) is waiting to receive a value on channel d. Thus we have $(\overline{\mathcal{M}}, \overline{\mathcal{L}}) \in \overline{\mathcal{S}}$

- $\mathcal{M} \stackrel{d.o_2?v}{\longrightarrow} \overline{\mathcal{M}}$, where $d \neq c$. This case is analogous to the one above
- $\mathcal{M} \stackrel{c.o_1?v}{\longrightarrow} \overline{\mathcal{M}'_v}$. In this case it is easy to show that $\overline{\mathcal{L}} \stackrel{c.o_1?v}{\longrightarrow} \overline{\mathcal{L}^1_v}$, and $(\overline{\mathcal{M}'_v}, \overline{\mathcal{L}^1_v}) \in \overline{\mathcal{S}}$
- $\mathcal{M} \stackrel{c.o_1?v}{\longrightarrow} \overline{\mathcal{M}'_v}$. Again this is straightforward.

It remains to check the pairs of the form $\{(\mathcal{M}'_v, \mathcal{L}^1_v)\}$ and $\{(\mathcal{M}'_v, \mathcal{L}^2_v)\}$. We only supply the details for the former case, as the latter is analogous.

- $\mathcal{M}'_v \stackrel{d.o_1?v}{\longrightarrow} \overline{\mathcal{M}'_v}$, where d is an arbitrary channel, including c. Note that, in $\mathcal{L}^1_v$, node m_1 is not waiting to receive a message along any channel. That is, we have $\mathcal{L}^1_v \stackrel{d.o_1?v}{\longrightarrow} \overline{\mathcal{L}^1_v}$, and $(\overline{\mathcal{M}^1_v}, \overline{\mathcal{L}^1_v}) \in \overline{S}$
- $\mathcal{M}'_v \stackrel{d.o_2?v}{\longrightarrow} \overline{\mathcal{M}'_v}$, where d is an arbitrary channel, including c. If $d \neq c$, then this case is similar to the above one. However, if $d = c$, note that node m_2 is waiting to receive a message along channel c. We already remarked that node m_2 can non-deterministically choose to ignore messages broadcast along channel c, so that it is easy to derive $\mathcal{L}^1_v \stackrel{c.o_2?v}{\longrightarrow} \overline{\mathcal{L}^1_v}$. Now it suffices to note that $(\overline{\mathcal{M}^1_v}, \overline{\mathcal{L}^1_v}) \in \mathcal{S}$
- $\mathcal{M}'_v \stackrel{c!v \triangleright \{o_1,o_2\}}{\longrightarrow} \overline{\mathcal{M}}$. This is the most interesting case. Here we exhibit a weak derivation of the form $\mathcal{L}^1_v \stackrel{c!v \triangleright \{o_1,o_2\}}{\Longrightarrow} \overline{\mathcal{L}}$. This is obtained by exploiting the non-standard definition of weak extensional outputs, given in Definition 7.1.2(3). Specifically, we show that that there exists a distribution Δ_1 such that

$$\mathcal{L}^1_v \stackrel{c!v \triangleright \{o_1\}}{\longrightarrow} \Delta_1 \stackrel{\tau}{\Longrightarrow} \overline{\mathcal{L}^2_v} \stackrel{c!v \triangleright \{o_2\}}{\longrightarrow} \overline{\mathcal{L}}. \tag{8.3}$$

here we have that

$$\Delta_1 = \sum_{j=1}^{k} \Lambda_i(j) \cdot \overline{\Gamma_L \triangleright n_i[\![c_j!\langle v\rangle.P_i]\!] \,|\, L_{n_i}}$$

The result will follow because $(\overline{\mathcal{M}}, \overline{\mathcal{L}}) \in \overline{\mathcal{S}}$.

The only difficult derivation to prove in Equation 8.3 is the hyper-derivation $\Delta_1 \stackrel{\tau}{\Longrightarrow} \overline{\mathcal{L}^2_v}$. All the other cases, in fact, are analogous to those analysed in Section 8.1.2.

Here the main idea is to reduce the network, in which the code at nodes n_i, m_i, where $i = 1,2$, is non-deterministic, to a deterministic one, in which the computation stops when the value v being routed is delivered at node m_2. For deterministic systems, in fact, we can rely on the useful result stated below.

Lemma 8.1.3. Let $\langle S, \mathrm{Act}, \tau, \omega\rangle$ be a deterministic pLTS, that is whenever $\Delta \stackrel{\tau}{\longrightarrow} \Delta'$ and $\Delta \stackrel{\tau}{\longrightarrow} \Delta''$ it holds that $\Delta' = \Delta''$. Then, whenever $\Delta \stackrel{\tau}{\longrightarrow} \Delta'$ and $\Delta \Longrightarrow \Theta$ it follows that $\Delta' \Longrightarrow \Theta$.

The resolution of our network to a deterministic one is very easy to define. Let $P'_2 = c_2?(x).c!\langle v\rangle.\mathbf{0}$, $P'_m = c?(x).\mathbf{0}$; the network $(\mathcal{L}_v{}^2)'$ is defined by replacing, in any of the states of the pLTS generated by $\mathcal{L}$, the code at nodes n_i, $i = 1,2$ with P'_2 and the code at nodes m_i, $i = 1,2$ with P'_m. That is, we establish that once that a message is received at node n_2, it will broadcast to node m_2, after which the computation of the network stops. This transformation can be applied to the network distribution Δ_1 defined above, leading to another network distribution Δ'_1.

It is trivial to note that if we prove that $\Delta'_1 \Longrightarrow \overline{(\mathcal{L}^2_v)'}$, then we also have that $\Delta_1 \Longrightarrow \mathcal{L}^2_v$. This is because $\Delta'_1, (\mathcal{L}^2_v)'$ have been defined by removing the non-deterministic choices from $\Delta_1, \mathcal{L}^2_v$ respectively, and by imposing that the computation stops once a message is received at the node m_2.

In practice, we show that $\Delta'_1 \Longrightarrow \overline{(\mathcal{L}^2_v)'}$. In the following, given a node $m \in \mathrm{nodes}(\mathcal{L})$ we use L'_m for the system term obtained from L_m by resolving the non-deterministic choices as described above, and by requiring that nodes n_2 and m_2 do not perform any activity after n_2 has broadcast a value along channel c. Further, we let

$$\begin{aligned} \Delta'_i &= \sum_{j=1}^{k} \Lambda_i(j) \cdot \overline{\Gamma_L \triangleright n_i[\![c_j!\langle v\rangle.P_i]\!] \,|\, L'_{n_i}}, \quad \text{if } i > 2 \\ \Delta'_2 &= \overline{\Gamma_L \triangleright n_2[\![c!\langle v\rangle.\mathbf{0}]\!] \,|\, L'_{n_2}} \end{aligned}$$

be probability distributions. We prove that, for any $i = 1, \cdots, k$, we have the extreme derivative $\Delta'_i \Longrightarrow \overline{(\mathcal{L}^2_v)'}$; to this end, we show that

- $\Delta'_2 \Longrightarrow \overline{(\mathcal{L}^2_v)'}$ and
- For any two indexes i, j are two nodes such that $\Gamma_L \vdash n_i \leftrightarrow n_j$, if $\Delta'_i \Longrightarrow \overline{(\mathcal{L}^2_v)'}$ then $\Delta'_j \Longrightarrow \overline{(\mathcal{L}^2_v)'}$.

Then it remains to note that the connectivity graph Γ_L has a single connected component to infer that $\Delta'_i \Longrightarrow \overline{(\mathcal{L}^2_v)'}$ for any i ranging over $1, \cdots, k$.

The proof that $\Delta'_2 \Longrightarrow \overline{(\mathcal{L}^2_v)'}$ is straightforward. In fact, we have that $\Delta'_2 = \overline{\Gamma_L \rhd n_2[\![c!\langle v\rangle.\mathbf{0}]\!] \mid L'_{n_2}} \xrightarrow{\tau} \overline{(\mathcal{L}^2_v)'}$, since no node can perform a transition in $(\mathcal{L}^2_v)'$ (recall that we changed the code at nodes m_2 so that, once it has received the value broadcast from node n_2 the network deadlocks), this transition can be easily transformed in an extreme derivation.

For the second statement, consider now a distribution Δ'_i, where $i = 1, \cdots, k$. This distribution in deterministic, and therefore it has a unique τ-transition. It is not difficult to show that, for $i \neq 2$, then $\Delta'_i \xrightarrow{\tau} \sum_{i=1}^{k} \Lambda_i(j) \cdot \Delta'_j$, where $\Lambda_i(j) > 0$ if and only if $\Gamma_L \vdash n_i \leftrightarrow n_j$. This is because, any state based network $(\mathcal{L}^j_i)' = \Gamma_L \rhd n_i[\![c_j!\langle v\rangle.P_i]\!] \mid L'_{n_i}$ has the unique transition $(\mathcal{L}^j_i)' \xrightarrow{\tau} \Delta'_j$.

Let then $j \neq 2$, and suppose that $\Gamma_L \vdash n_i \leftrightarrow n_j$ for some index i (possibly equal to 2). Also, suppose that $\Delta'_i \Longrightarrow \overline{(\mathcal{L}^2_v)'}$. We have already proved that $\Delta'_i \xrightarrow{\tau} p \cdot \Delta'_j + (1-p) \cdot \Theta$ for some distribution Θ and $p \in [0,1]$ such that $p > 0$. By Lemma 8.1.3 it follows that $(p \cdot \Delta'_j + (1-p) \cdot \Theta) \Longrightarrow \overline{(\mathcal{L}^2_v)'}$; since $p > 0$ this leads to $\Delta'_j \Longrightarrow \overline{(\mathcal{L}^2_v)'}$, which is exactly what we wanted to prove.

Then, as we already observed, the assumption that the graph Γ_L has a single connected component allows us to infer that $\Delta'_i \Longrightarrow \overline{(\mathcal{L}^2_v)'}$ for any $i = 1, \cdots, k$. In particular, $\Delta'_1 \Longrightarrow \overline{(\mathcal{L}^2_v)'}$. We have also remarked that this extreme derivative corresponds to $\Delta_1 \Longrightarrow \overline{\mathcal{L}^2_v}$ (recall that these distributions refer to the network in which the nodes n_2 and m_2 have non-deterministic behaviour), which is exactly what we wanted to prove.

8.2 Probabilistic Connection-less Routing

In this Section we propose a probabilistic implementation for the connection-less routing model $\mathcal{M}_k$, defined in Section 5.2.2 at Page 106.

Specifically, we consider a parametrised network whose internal nodes can behave probabilistically; the main idea is that, at any given time, the next hop in a routing path is chosen probabilistically. In practice, this implementation is similar to the one proposed in Section 5.2.3. The only differences between such an implementation and the one we proposed are listed below:

- Node n_1 can now detect values along channel c_1. When this happens, it stores the received value in its local buffer, represented as a multiset $\mathscr{M}$,
- At any given time, a node can decide to broadcast one of the values in its buffer to one of its neighbour; recall that the neighbour to which the value will be broadcast is chosen by a node by broadcasting the value along a channel whose transmission can be detected by the intended next hop in a routing path. The node n selects the next node in a routing path probabilistically among all the neighbour of node n. This is in contrast with the implementation proposed in Section 5.2.3, where a node could only select the next hop in a routing path among the nodes which were closer to the destination node o,
- We drop some of the constraints we impose on the network topology; specifically, we just require that there is a path from the input node i to the input node o in the connectivity graph.

Formally, our probabilistic (parametrised) implementation $\mathcal{N}_k = \Gamma_N \rhd N_k$ for a routing model $\mathcal{M}_k$ is defined as follows:

- $\text{nodes}(N_k) = \{n_1, \cdots, n_j\}$ for some index $j \geq 2$

- $\mathsf{Input}(\mathcal{N}_k) = \{i\}, \mathsf{Output}(\mathcal{N}_k) = \{o\}$; further, we assume that whenever $\Gamma_N \vdash i \to n_h$ for the only index $h = 1$. Similarly, if $\Gamma_N \vdash n_h \to o$ for the only index $h = 2$
- $\Gamma_N \vdash n_1 \twoheadrightarrow n_2$
- For any node in $n \in \text{nodes}(N_k)$ there exists a directed path from the internal node n to the output node o; further, for such nodes we let $\text{length}(n, \Gamma_N)$ be the length of the minimal path between the node n and the output node o and $\text{next}(h, \Gamma_N) = \{h' \mid \Gamma_N \vdash n_h \to n_{h'} \text{ and } \text{length}(n_{h'}, \Gamma_N) = \text{length}(n_h, \Gamma_N) - 1\}$
- For any node $h = 2, \cdots, j$ it holds $\text{length}(n_h, \Gamma_N) \leq \text{length}(n_1, \Gamma_N)$,
- For any index $h = 1, \cdots, j$ we assume a probability distribution $\Lambda(h) : \{1, \cdots, j\} \to [0, 1]$ such that $h' \in \lceil \Lambda(h) \rceil$ if and only if $\Gamma_N \vdash n_h \to n_{h'}$
- For any index $h = 2, \cdots, j$, we assume the existence of a channel $c_h \in \mathbf{Ch}$, different from the channel c used in the routing model $\mathcal{M}_k$
- Let h, l be two indexes ranging over $3, \cdots, j$ and $0, \cdots, k$,respectively; further, let $\mathscr{M}$ be a finite multiset of values. We make use of the following process definitions

$$
\begin{aligned}
Q^2_{\mathscr{M}} &\Leftarrow c_2?(x).Q^2_{(\mathscr{M}\cup\{x\})} + \left(\sum_{v\in\mathscr{M}} c!\langle v\rangle.Q^2_{(\mathscr{M}\setminus\{x\})}\right) \\
Q^h_{\mathscr{M}} &\Leftarrow \bigoplus_{h':\Gamma_N\vdash n_h\to n'_h} \Lambda_h(h')\cdot\left[(c_h?(x).Q^h_{(\mathscr{M}\cup\{x\})}) + \left(\sum_{v\in\mathscr{M}} c_{h'}!\langle v\rangle.Q^h_{(\mathscr{M}\setminus\{v\})}\right)\right] \\
R^0_{\mathscr{M}} &\Leftarrow \bigoplus_{h:\Gamma_N\vdash n_1\to n_h} l\left[(c_1?(x).R^0_{(\mathscr{M}\cup\{x\})}) + \left(\sum_{v\in\mathscr{M}} c_h!\langle v\rangle.R^0_{(\mathscr{M}\setminus\{v\})}\right)\right] \\
R^{l+1}_{\mathscr{M}} &\Leftarrow \bigoplus_{h:\Gamma_N\vdash n_1\to n_h} [(c?(x).R^l_{(\mathscr{M}\cup\{x\})}) + (c_1?(x).R^l_{(\mathscr{M}\cup\{x\})}) + \\
&+ \left(\sum_{v\in\mathscr{M}} c_h!\langle v\rangle.R^{l+1}_{(\mathscr{M}\setminus\{v\})}\right)]
\end{aligned}
$$

to define the system term N_k as

$$N_k = n_1[\![R^k_\emptyset]\!] \mid \prod_{h=2}^{j} n_h[\![Q^h_\emptyset]\!]$$

Remark 8.2.1. Note that we used an abuse of notation in the definition of the network $\Gamma_N \rhd N_k$; in fact, recall that process definitions must be state-based networks; however, the process definitions $Q^h_{\mathscr{M}}$, $h \neq 2$ and $R^l_{\mathscr{M}}$, $l \geq 0$ violate this constraint.

In practice, we can always rewrite a process definition of the form $P \Leftarrow \bigoplus_{i\in I} p_i \cdot S_i$, by introducing a collection of process definitions $A_i \Leftarrow \{\bigoplus_{i\in I} p_i \cdot A_i / P\}S_i$; a system term of the form $n[\![P]\!]$ can then be rewritten as $n[\![\bigoplus_{i\in I} p_i \cdot A_i]\!]$.

As it is not difficult to note, the behaviour of the network $\Gamma_N \rhd N$ resembles that of the implementation provided for sequential routing; however, multisets are used in internal nodes to equip them with a buffer. As we already did for the sequential routing model and implementation, we can exhibit a (simple) simulation between the connectionless routing model and the network $\mathcal{N}_k = \Gamma_N \rhd N_k$.

Theorem 8.2.2. For any $k \geq 0$, $\mathcal{M}_k \lhd^s \overline{\mathcal{N}_k}$. As a consequence, $\mathcal{M}_k \sqsubseteq_{\text{may}} \mathcal{N}_k$.

Part III

Time and Collisions

Chapter 9

A Timed Calculus for Collisions

In this Chapter we study the behavioural semantics of a simple Calculus of Collision-prone Communicating Processes (CCCP); the contents included in this Chapter and in Chapter 10 are an extension of the topics covered in [10], which at the present date is still in preparation.

The calculus we present is designed around the following concepts:

- Value-passing broadcast (channels are not passed around)
- Time-consuming communication; each value has its own amount of time which is needed for it to be sent along a channel.
- Communication collision; if more values are being transmitted over a channel, then receivers will not be able to receive any of them correctly.
- Perfect receivers. A receiver will start receiving every time it detects the channel it is listening to is busy (that is, some station is sending some value along that channel). However, a value can be received correctly by a receiver only if the two entities involved in the communication synchronised are at the beginning of the transmission; that is, the sender and the receiver start respectively to send and to receive a value together.
- Flat communication topology

In this Chapter we cover the following topics:

- A syntax for terms of CCCP, together with some illumination examples; this topic is illustrated in Section 9.1
- A reduction semantics to describe the dynamics of systems, presented in Section 9.2
- Standard time properties, listed in Section 9.3 :
  - time determinism
  - maximal progress
- A LTS semantics, which is introduced in Section 9.4
- A comparison between the reduction semantics and the LTS semantics, Section 9.5
- A brief discussion of related work, in Section 9.6

Processes:

W	::=	P	inactive process
	$\mid$	$c[x].P$	active receiver
	$\mid$	$W_1 \mid W_2$	parallel composition
	$\mid$	$\nu c : (n, v).W$	channel restriction
P, Q	::=	nil	termination
	$\mid$	$c\,!\langle u \rangle.P$	broadcast
	$\mid$	$\lfloor c?(x).P \rfloor Q$	receiver with timeout
	$\mid$	$\tau.P$	internal activity
	$\mid$	$P + Q$	choice
	$\mid$	$\sigma.P$	delay
	$\mid$	$[b]P, Q$	matching
	$\mid$	$\langle c \rangle P, Q$	exposure check
	$\mid$	X	process variable
	$\mid$	$\mathsf{fix}\, X.P$	recursion

Channel Environments: $\Gamma : \mathcal{C} \to (\mathbb{N} \cup \{\infty\}) \times \mathcal{V}^+$

Configurations: $\Gamma \rhd W$

Table 9.1: The Syntax

9.1 The calculus

In Table 9.1, we define the syntax of CCCP. We use letters $c, d, \ldots$ for channel names, belonging to the set $\mathbf{Ch}$, x, y, z for variables, u for values, and v for closed values, i.e. values that do not contain variables; we write $\mathbf{Val}$ to denote the set of all possible closed values. Closed values actually represent messages that are transmitted as TCP/IP packets. We extend the set of possible closed values with two special values $\mathbf{x}$ and err denoting unknown and corrupted messages, respectively. The former value can not be used in syntax of our processes but it will be useful in the semantics. Thus, we let $\mathbf{Val}^+ = \mathbf{Val} \cup \{\mathbf{x}, \mathsf{err}\}$ and we use the letter w to range over $\mathbf{Val}^+$. We write $\overline{u}$ to denote a tuple $u_1, \ldots, u_k$ of values.

We associate a strictly positive integer δ_v to each closed value v, denoting the length of time slots required for such a value to be transmitted along a channel. Upon successful reception the variable x of P is instantiated with the transmitted message.

In our calculus, we distinguish between non-active and active processes. $c[x].P$ is a process which is currently receiving a value along a channel c; its computation cannot evolve until the transmission along such a channel has terminated. The information on the length of a transmission and of the value which is being received are stored in different data structures called *channel environments* and discussed later in this Section.

The symbol nil denotes the skip process. The construct $\sigma.P$ models a sleeping process which after one time interval evolves into P. We write $\sigma^n.P$ for a process sleeping for the next n time intervals. The sender process $c\,!\langle v \rangle.P$ allows to broadcast the value v along channel c; once the transmission starts the process will be busy in transmitting v for the next δ_v instants time, evolving into the process $\sigma^{\delta_v}.P$. A communication starts even if there are no listeners: broadcasting is a *non-blocking* action. Actually, broadcasting is always enabled and it can not be delayed: a broadcasting summand $c\,!\langle v \rangle.P$ can not be delayed to the next time interval (as we will see in Section 9.3, this property is called *maximal progress*).

The process $\lfloor c?(x).P \rfloor Q$ denotes a receiving process subject to timeout; if a value is detected as being transmitted along channel c the process engages in the communication and starts receiving, in which case it will evolve in the active receiver process $c[x].P$, otherwise it timeouts and evolves in process Q after an instant of time has passed.

Process $[b]P, Q$ is the standard "if then else" construct: here b is a boolean expression from some decidable Theory; we assume an evaluation function $[\![\cdot]\!]$ which maps boolean statements in elements of the set

{true, false}.

The calculus is equipped with a second matching construct, $\langle c \rangle P, Q$. This process checks if there is some process transmitting along channel c, in which case it waits a time slot before evolving in P; otherwise, it waits an instant of time, then it evolves in Q. We always require that after an exposure check a time slot needs to pass before the computation of a process can continue; this is because in practice exposure check requires a station trying to detect a transmission over a channel; since communication in wireless systems is *half-duplex*, we want to prevent activities in our calculus such as broadcasting a value in the same time slot in which an exposure check has been performed.

The process $\mathsf{fix} X.P$ denotes guarded recursion. Finally, we write $W_1 | W_2$ to denote standard parallel composition, and $\nu c : (n, v).W$ for process W in which channel c is restricted. The restriction operator is parametrised in a natural number n and a value v; intuitively, the latter correspond to the value being sent along the restricted channel, while the former to the amount of times instants needed to finish the transmission of such a value.

Remark 9.1.1. The recursion construct allows us to define receivers which are not subject to timeout, specifically through the construct $\mathsf{fix} X.\lfloor c?(x).P \rfloor X$; with an abuse of notation, the above term is denoted as $c?(x).P$.

In both terms $c?(x).P$ and $c[x].P$ the variable x is bound in P. Similarly, in process $\mathsf{fix} X.P$ the process variable X is bound in P, while channel name c is bound in $\nu c : (n, v).W$. This gives rise to the standard notions of *free (process) variables* and *bound (process) variables*, as well as free channel names and bound channel names. The set of free channel names in a term W is denoted by $\mathrm{fn}(W)$. Also, we can introduce the standard notion of *α-conversion*, and we identify processes and configurations up to α-conversion. A term is said to be *closed* if it does not contain free (process) variables. We always work with closed configurations; the absence of free variables is trivially maintained at run-time.

We write $\{^w/_x\}W$ for the substitution of the variable x with the value w in the process W. Similarly, we write $\{^P/_X\}W$ for the substitution of the process variable X with the process P in W.

A channel environment Γ is a total function from the set of channel names C to the Cartesian product of the natural numbers $\mathbb{N}$ and the set of closed values $\mathbf{Val}^+$. Intuitively, if $\Gamma(c) = (k, w)$ then channel c is busy (or *exposed*) for the next k intervals of time in transmitting w. If $\Gamma(c) = (0, \mathbf{x})$ then channel c is free, i.e. no process is currently transmitting. at c.

In the following we will need some definitions for channel environments. We say that $\Gamma \vdash_t c : n$ if $\Gamma(c) = (n, v)$ for some values $v \in \mathbf{Val}^+$, and $\Gamma \vdash_v c : v$ if $\Gamma(c) = (n, v)$, for some element in $\mathbb{N}^\infty = \mathbb{N} \cup \infty$. We also use $\Gamma \vdash c :$ free if $\Gamma(c) = (0, \mathbf{x})$ and $\Gamma \vdash c :$ exp if $\Gamma(c) = (n, v)$ for some $n > 0$ and $v \in \mathbf{Val}^+$. The former notation means that channel c is not exposed in Γ, while the latter is its negation. Finally, we write $\Gamma \vdash c$ deliver v if $\Gamma(c) = (1, v)$. That is, the transmission of value v along channel c in environment Γ will end in the next time slot (unless some other transmission along the same channel takes place before).

We also define some operations over channel environments; given an environment Γ, we define $\Gamma \ominus 1$ as the channel environment such that

$$(\Gamma \ominus 1)(c) = \begin{cases} (n-1, v) & \text{if } \Gamma(c) = (n, v) \text{ and } n \geq 2 \\ (0, \mathbf{x}) & \text{otherwise} \end{cases}$$

At least intuitively, $(\Gamma \ominus 1)$ is the environment obtained from Γ after a time slot has passed.

Let c be a channel, w be a value and suppose that $\Gamma \vdash_t c : n$ for some $n \in \mathbb{N}^\infty$; the environment $\mathrm{upd}_c^v(\Gamma)$ is defined as $\Gamma[c \mapsto (n', v')]$, where

$$\begin{aligned} n' &= \max\{n, \delta_v\} \\ v' &= \begin{cases} w & \text{if } n = 0 \\ \mathsf{err} & \text{otherwise} \end{cases} \end{aligned}$$

Informally speaking, $\mathrm{upd}_c^v(\Gamma)$ represents the environment obtained from Γ when a station broadcasts a value

along channel c. If the channel is free, then it becomes busy transmitting value v for the next δ_v instants of times. Otherwise, the channel will remain busy for as long as it requires to end both the current transmission and the transmission of value v; further, since channel c is exposed to two different transmissions at the same time, a collision has occurred, and the value that will be received by any station at the end of the transmission is the error value err.

Definition 9.1.2. *Structural congruence* We introduce a structural congruence over CCCP, denoted by $\equiv$; this is defined as the smallest congruence relation which is a commutative monoid with respect to both parallel composition and choice, and which satisfies the following equations:

- $\nu c : (n_c, v_c).\nu d : (n_d, v_d).W \equiv \nu d : (n_d, v_d).\nu c : (n_c, v_c).W$, provided $c \neq d$
- $\nu c : (n_1, v_1).\nu c : (n_2, v_2).W \equiv \nu c : (n_2, v_2).W$
- $(\nu c : (n, v).W_1)|W_2 \equiv \nu c : (n, v).(W_1|W_2)$, provided $c \notin \mathrm{fn}(W_2)$
- $[b]P, Q \equiv P$ if $[\![b]\!] = \mathsf{true}$
- $[b]P, Q \equiv Q$ if $[\![b]\!] = \mathsf{false}$
- $\mathsf{fix}\, X..P \equiv \{^P/_X\}P$

We write $\Gamma_1 \rhd W_1 \equiv \Gamma_2 \rhd W_2$, if $\Gamma_1 = \Gamma_2$ and $W_1 \equiv W_2$. □

In the sequel we use a number of notational conventions. We often use the terms l, l' to denote lists of elements in $\mathbf{Ch} \times \mathbb{N}^\infty \times \mathbf{Val}^+$; such elements are represented in lists using the notation $c : (n, v)$ already introduced for restriction, while the operator :: denotes list concatenation. If $l = c_1 : (n_1, v_1) :: \cdots :: c_k : (n_k, v_k)$, we use the shortcut $\nu l.W$ to denote the term $\nu c_1 : (n_1, v_1).\nu c_2 : (n_2, v_2).\cdots.\nu c_k : (n_k, v_k).W$. If the list l is empty, the term $\nu l.W$ coincides with W. Given a list of the form $c_1 : (n_1, v_1) :: \cdots c_k : (n_k, v_k)$, $\mathrm{ch}(l)$ is defined to be equal to $c_1 :: \cdots, c_k$, while $\mathrm{Val}(l)$ is defined to be the list $(n_1, v_1) :: \cdots :: (n_k, v_k)$. Given an environment Γ and a list $l = c_1 : (n_1, v_1) :: \cdots :: c_k : (n_k, v_k)$, the channel environment $\Gamma[l]$ is defined to be $\Gamma[c_1 \mapsto (n_1, v_1)] \cdots [c_k \mapsto (n_k, v_k)]$.

We write $c!\langle v \rangle$ for $c!\langle v \rangle.\mathsf{nil}$, τ for $\tau.\mathsf{nil}$ and σ^δ for $\sigma^\delta.\mathsf{nil}$. When we do not need the exposure information of a restricted channel, we also use the shortcut $\nu c.W$ for $\nu c : (n, v).W$

The following predicates will be useful later. For any configuration W and channel c, the boolean predicate $\mathrm{rcv}(W, c)$ is defined to be true if $W \equiv \nu l.(W'|(\lfloor c?(x).P \rfloor Q + R))$ and c does not appear in $\mathrm{ch}(l)$; otherwise the predicate is false. In other words, the predicate $\mathrm{rcv}(W, c)$ is true if W contains a process which is waiting to detect a message along the unrestricted channel c. The predicate $\mathsf{snd}(W, c)$ returns true if $W \equiv \nu l.(W'|c!\langle v \rangle.P + Q)$, provided that c does not appear in $\mathrm{Ch}(l)$. That is, $\mathsf{snd}(W, c)$ is true whenever W contains a process which is able to broadcast some value along the unrestricted channel c. false otherwise.

in the following we will only focus with a subset of configurations, which we call well-formed; these are those configurations $\Gamma \rhd W$ in which the information contained in Γ is consistent with the structure of process W.

Definition 9.1.3. [Well-formedness] A configuration $\Gamma \rhd W$ is said to be *well-formed* if whenever $W \equiv \nu l.(W'|c[x].P)$ then $\Gamma[l] \vdash c : \mathrm{exp}$. □

A well-formed configuration is one in which an active receiver along a (either restricted or unrestricted) channel is allowed only whenever such a channel is exposed. Henceforth we will always assume that a configuration is well-formed, unless otherwise stated.

9.2 Reduction Semantics

The dynamics of the calculus is given in terms of a reduction relation.

(R-NoColl)
$$\frac{\neg \mathrm{rcv}(W,c) \quad \Gamma \vdash c : \mathrm{free}}{\Gamma \rhd (c\,!\langle v\rangle.P + Q \mid \prod_{j\in J} \lfloor c?(x).P_j \rfloor Q_j + R_j \mid W) \twoheadrightarrow_u \mathrm{upd}_c^v(\Gamma) \rhd (\sigma^{\delta_v}.P \mid \prod_{j\in J} c[x].P_j \mid W)}$$

(R-Coll)
$$\frac{\Gamma \vdash c : \exp}{\Gamma \rhd c\,!\langle v\rangle.P + Q \mid W \twoheadrightarrow_u \mathrm{upd}_c^v(\Gamma) \rhd \sigma^{\delta_v}.P \mid W}$$

(R-LateWakeUp)
$$\frac{\Gamma \vdash c : \exp}{\Gamma \rhd \lfloor c?(x).P \rfloor Q + R \mid W \twoheadrightarrow_u \Gamma \rhd c[x].\{^{\mathsf{err}}/_x\}P \mid W}$$

(R-Internal)
$$\frac{}{\Gamma \rhd \tau.P + Q \mid W \twoheadrightarrow_u \Gamma \rhd P \mid W}$$

(R-Time)
$$\frac{\forall j{\in}J \forall r{\in}R.\Gamma \vdash c_{rj} : \mathrm{free} \quad \forall k \in K.(\Gamma \vdash_t c_k : n_k), n_k > 1 \quad \forall l \in L.\Gamma \vdash c_l \text{ deliver } w_l}{\Gamma \rhd (\prod_{j\in J} \sum_{r\in R} \lfloor c_{rj}?(x).P_{rj} \rfloor Q_{rj} + \sum_{s\in S} \sigma.P_{sj} \mid \prod_{k\in K} c_k[x].P_k \mid \prod_{l\in L} c_l[x].P_l) \twoheadrightarrow_\sigma \twoheadrightarrow_\sigma \Gamma\ominus 1 \rhd (\prod_{j\in J} \sum_{r\in R} Q_{rj} + \sum_{s\in S} P_{sj} \mid \prod_{k\in K} c_k[x].P_k \mid \prod_{l\in L} \{^{w_l}/_x\}P_l)}$$

(R-If)
$$\frac{\Gamma \vdash c : \exp}{\Gamma \rhd \langle c\rangle P, Q | W \twoheadrightarrow_u \Gamma \rhd \sigma.P | W}$$

(R-Else)
$$\frac{\Gamma \vdash c : free}{\Gamma \rhd \langle c\rangle P, Q | W \twoheadrightarrow_u \Gamma \rhd \sigma.Q | W}$$

(R-RstrTmd)
$$\frac{\Gamma[c \mapsto (n,v)] \rhd W \twoheadrightarrow_\sigma \Gamma' \rhd W'}{\Gamma \rhd \nu c : (n,v).W \twoheadrightarrow_\sigma \Gamma \ominus 1 \rhd \nu c : \Gamma'(c).W'}$$

(R-RstrUntmd)
$$\frac{\Gamma[c \mapsto (n,v)] \rhd W \twoheadrightarrow_u \Gamma' \rhd W'}{\Gamma \rhd \nu c : (n,v).W \twoheadrightarrow_u \Gamma'[c \mapsto \Gamma(c)] \rhd \nu c : \Gamma'(c).W'}$$

(R-Struct)
$$\frac{\Gamma \rhd W_1 \equiv \Gamma \rhd W_2 \quad \Gamma \rhd W_2 \twoheadrightarrow_x \Gamma' \rhd W_2' \quad \Gamma' \rhd W_1' \equiv \Gamma' \rhd W_2' \quad \twoheadrightarrow_x \in \{\twoheadrightarrow_u, \twoheadrightarrow_\sigma\}}{\Gamma \rhd W_1 \twoheadrightarrow_x \Gamma' \rhd W_1'}$$

Table 9.2: Reduction Semantics

Definition 9.2.1 (Reduction Semantics)**.** The *reduction semantics* of CCCP, denoted $\twoheadrightarrow$, is defined as $\twoheadrightarrow = \twoheadrightarrow_u \cup \twoheadrightarrow_\sigma$, where relations $\twoheadrightarrow_u$ and $\twoheadrightarrow_\sigma$ are the smallest binary relations over configurations satisfying the rules in Table 9.2.

Intuitively speaking, $\twoheadrightarrow_u$ denotes untimed reduction, while $\twoheadrightarrow_\sigma$ denotes timed reduction, which corresponds to the passage of time.

Remark 9.2.2. We could have defined a single reduction relation for processes; however, having two different forms of reduction, one associated with the passage of time and the other associated with instantaneous activities, comes handful when discussing properties of processes which concern the passage of time.

We use $\twoheadrightarrow_u^*$ to denote the reflexive transitive closure of $\twoheadrightarrow_u$, while $\twoheadrightarrow_\sigma^*$ is defined to be equivalent to $\twoheadrightarrow_u^* \twoheadrightarrow_\sigma \twoheadrightarrow_u^*$. Finally, $\twoheadrightarrow^*$ denotes the reflexive transitive closure of $\twoheadrightarrow$.

Let us discuss the rules of Table 9.2. Rules (R-Coll) and (R-NoColl) describe how systems evolve when broadcasting a message on a channel c, with and without collisions, respectively. In both cases the station

which performs the broadcast remains idle for the next δ_v instants of time, for during such a time interval it will be busy transmitting the value v, and the channel environment is updated accordingly. If the channel c is free in Γ (which case is described by Rule (R-COLL)) then every process which is waiting to detect a value along such a channel begins the reception, evolving into an active receiver. All the other components will be unaffected by the broadcast performed along channel c. In the case that $\Gamma \vdash c : \mathsf{exp}$ none of the parallel component, apart from the one performing the broadcast, is affected.

Rule (R-RCVLATE) models a receiver that starts receiving when the channel is already in use. In this case such a receiver starts the reception; however, since it missed part of the transmission, it is doomed to receive the error value err. Rule (R-INTERNAL) models internal actions within nodes.

Rule (R-TIME) models the passage of time; this rule establishes that

- active receivers of values not being delivered will continue receive a value after time has passed
- active receivers along a channel which delivers a value v will end receiving such a value and will perform the required substitutions
- all the other components, which are waiting for time to pass before continuing their computation, can now continue performing some activities.

Rules (R-RSTRUNTMD) and (R-RSTRTMD) model the behaviour of restricted configurations. Here, the idea is that of observing the behaviour of a configuration $\Gamma \rhd W$, where $\Gamma(c) = (n, v)$, to infer the behaviour of a configuration $\Gamma' \rhd \nu c : (n, v).W$, provided that $\Gamma(d) = \Gamma'(d)$ whenever $d \neq c$. The rules also establish that untimed reductions only influence the exposure information of the restricted channel, while the value of c in the channel environment Γ' is not affected. Conversely, for timed reductions, both the exposure information of the restricted channel and of the exposure channel Γ' have to be updated.

As usual in process calculi, the reduction semantics relies on structural congruence, $\equiv$, defined above. Basically, $\equiv$ brings the participants of a potential interaction into contiguous positions.

Note that we can model processes for which time cannot pass; one example is given by $\mathsf{fix} X.\tau.P$, which can only perform an untimed reduction (this can be proved using rules (R-INTERNAL) and (R-STRUCT)), after which it evolves in itself.

Definition 9.2.3. [Well-timedness] We say that a process $\Gamma \rhd W$ is well-timed if whenever $\Gamma \rhd W \twoheadrightarrow^* \Gamma' \rhd W'$ there exists a configuration $\Gamma'' \rhd W''$ such that $\Gamma' \rhd W' \twoheadrightarrow^*_\sigma \Gamma'' \rhd W''$. □

Before discussing various properties enjoyed by our calculus, let us prove that well-formedness is preserved at run-time.

Theorem 9.2.4. [Well-formedness preservation] Let $\Gamma \rhd W$ be a well-formed configuration. If $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$ then $\Gamma' \rhd W'$ is well-formed as well.

Proof. The proof is performed by structural induction on the proof of the reduction $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$. See Appendix C, Section C.1, Page 237, for an outline of the proof. □

9.3 Time Properties

In this Section we prove some desirable properties which are enjoyed by processes of our calculus.

Theorem 9.3.1 formalises the deterministic nature of time passing. Informally speaking, we do not allow the passage of time to resolve non-deterministic choices.

Theorem 9.3.1 (Time Determinism)**.** Let $\Gamma \rhd W$ be a well-formed configuration. If $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma_1 \rhd W_1$ and $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma_2 \rhd W_2$ then $\Gamma_1 \rhd W_1 \equiv \Gamma_2 \rhd W_2$.

Proof. Straightforward by the definition of rules **(Time)** and **(RstrTmd)**. For a detailed outline see the Appendix, Page 237 □

In [36, 55], the maximal progress property says that processes communicate as soon as a possibility of communication arises. However, unlike [36, 55], in our calculus message transmission requires a positive amount of time. So, we generalise the property saying that transmissions cannot be delayed.

Theorem 9.3.2 (Maximal Progress)**.** Let $\Gamma \rhd W$ be a well-formed configuration. If there are Γ_1 and W_1 such that $\Gamma \rhd W \twoheadrightarrow_u \Gamma_1 \rhd W_1$, then $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma' \rhd W'$ for no configuration $\Gamma' \rhd W'$.

Proof. An outline of the proof is included in the Appendix, Page 238 □

Remark 9.3.3. Although we have maximal progress, it is possible to explicitly model processes in which a broadcast can be delayed. For example, in process

$$\Gamma \rhd \tau.c\,!\langle v\rangle.\mathsf{nil} + \tau.\sigma.c\,!\langle v\rangle.\mathsf{nil}$$

there are two possible choices, as a consequence of some internal activity:

- value v will be broadcast along channel c, or
- no value will be broadcast, yet no activity will be performed in the first time instant. However, after time has passed, the broadcast of value v along channel c is fired.

In general, it is possible to delay a broadcast indefinitely; this is modelled by process

$$\mathsf{fix}\, X.\tau.c\,!\langle v\rangle.\mathsf{nil} + \tau.\sigma.X$$

Finally, the exposure time of a channel can be decremented only by timed reductions; this property guarantees that communication along a channel is flushed with time. This is modelled in our calculus by requiring that untimed reductions may only increment the exposure time of a channel. To this end, given two channel environments Γ_1, Γ_2 such that for any channel c it holds that $\Gamma_1 \vdash_t n_1$, $\Gamma_2 \vdash_t n_2$ implies $n_1 \leq n_2$, then we say that $\Gamma_1 \leq \Gamma_2$.

Proposition 9.3.4 (Exposure Consistency)**.** Let $\Gamma \rhd W$ be a well-formed configuration. If $\Gamma \rhd W \twoheadrightarrow_u \Gamma' \rhd W'$, then for any channel c it holds $\Gamma(c)_{\mathbf{1}} \leq \Gamma'(c)_{\mathbf{1}}$.

Proof. The proof of this Proposition can be obtained by noting that for any channel environment Γ, channel c and value v it holds $\Gamma \leq (\mathrm{upd}_c^v(\Gamma))$, then by performing a structural induction on the proof of the reduction $\Gamma \rhd W \twoheadrightarrow_u \Gamma' \rhd W'$. □

9.4 Labelled transition Semantics

In this Section we present another transition semantics for CCCP terms, the labelled transition semantics; their SOS rules are split in different tables, according to the kind of activity they model.

Table 9.3 contains the rules governing transmission. Rule (Snd) indicates that broadcasts are non-blocking; they can happen at any time, in particular independently of the state of the network; the notation σ^{δ_v} represents the time delay operator σ iterated δ_v times. So when the process $c\,!\langle v\rangle.P$ broadcasts it has to wait δ_v time units before the residual P is activated. On the other hand reception of a message, by a time-guarded listener $\lfloor c?(x).P \rfloor Q$, depends on the state of the network. If the channel c is free then rule (B-RCV) indicates that reception can start; the listener evolves into the active receiver $c[x].P$. On the other hand if the channel is already exposed then by (B-RCVFAIL) the transmission is ignored and the reception is doomed to fail.

If the code W can not receive on channel c, $\neg\mathrm{rcv}(P, c)$, then the transmission is also ignored. The remaining two rules in Table 9.3 (B-SYNC) and (B-RCVPAR) serves to synchronise parallel stations on the same transmission [32].

(B-SND)
$$\Gamma \rhd c\,!\langle v\rangle.P \xrightarrow{c!v} \mathrm{upd}^{v}_{c}(\Gamma) \rhd \sigma^{\delta_v}.P$$

(B-RCV)
$$\frac{\Gamma \vdash c : \mathrm{free}}{\Gamma \rhd \lfloor c?(x).P \rfloor Q \xrightarrow{c?v} \mathrm{upd}^{v}_{c}(\Gamma) \rhd c[x].P}$$

(B-RCVFAIL)
$$\frac{\Gamma \vdash c : \mathrm{exp}}{\Gamma \rhd W \xrightarrow{c?v} \mathrm{upd}^{v}_{c}(\Gamma) \rhd W}$$

(B-RCVIGN)
$$\frac{\neg \mathrm{rcv}(W, c)}{\Gamma \rhd W \xrightarrow{c?v} \mathrm{upd}^{v}_{c}(\Gamma) \rhd W}$$

(B-SYNC)
$$\frac{\Gamma \rhd W_1 \xrightarrow{c!v} \Gamma' \rhd W_1' \quad \Gamma \rhd W_2 \xrightarrow{c?v} \Gamma' \rhd W_2'}{\Gamma \rhd W_1 | W_2 \xrightarrow{c!v} \Gamma' \rhd W_1' | W_2'}$$

(B-RCVPAR)
$$\frac{\Gamma \rhd W_1 \xrightarrow{c?v} \Gamma' \rhd W_1' \quad \Gamma \rhd W_2 \xrightarrow{c?v} \Gamma' \rhd W_2'}{\Gamma \rhd W_1 | W_2 \xrightarrow{c?v} \Gamma' \rhd W_1' | W_2'}$$

Table 9.3: Intensional semantics: transmission

(B-TIMENIL)
$$\Gamma \rhd \mathsf{nil} \xrightarrow{\sigma} \Gamma \ominus 1 \rhd \mathsf{nil}$$

(B-SLEEP)
$$\Gamma \rhd \sigma.P \xrightarrow{\sigma} \Gamma \ominus 1 \rhd P$$

(B-ACTRCV)
$$\frac{\Gamma \vdash_t c : n, n > 1}{\Gamma \rhd c[x].P \xrightarrow{\sigma} \Gamma \ominus 1 \rhd c[x].P}$$

(B-ENDRCV)
$$\frac{\Gamma \vdash_t c : 1, \Gamma \vdash_v c = w}{\Gamma \rhd c[x].P \xrightarrow{\sigma} \Gamma \ominus 1 \rhd \{^{w}/_{x}\}P}$$

(B-SUMTIME)
$$\frac{\Gamma \rhd P \xrightarrow{\sigma} \Gamma' \rhd P' \quad \Gamma \rhd Q \xrightarrow{\sigma} \Gamma' \rhd Q'}{\Gamma \rhd P + Q \xrightarrow{\sigma} \Gamma' \rhd P' + Q'}$$

(B-TIMEOUT)
$$\frac{\Gamma \vdash c : \mathrm{free}}{\Gamma \rhd \lfloor c?(x).P \rfloor Q \xrightarrow{\sigma} \Gamma \ominus 1 \rhd Q}$$

Table 9.4: Intensional semantics: timed transitions

The rules for the passage of time, $\Gamma \rhd W \xrightarrow{\sigma} \Gamma' \rhd W'$, are given in Table 9.4 and are straightforward. In the rules (B-ACTREC) and (B-ENDRCV) we see that the active receiver $c[x].P$ continues to wait for the transmitted value to make its way through the network; when the allocated transmission time elapses the value is then delivered and the receiver evolves to $\{^{w}/_{x}\}P$. Rule (B-SUMTIME) establishes that the passage of time is deterministic. Finally (B-TIMEOUT) implements the idea that $\lfloor c?(x).P \rfloor Q$ is a time-guarded receptor; when time passes it evolves into the alternative Q. However this only happens if the channel c is not exposed. What happens if it is exposed is explained in the Table 9.5.

This is devoted to internal transitions $\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'$. Intuitively the process $\lfloor c?(x).P \rfloor Q$ is ready to start receiving a value on channel c. However if c is already exposed in the network this means that a transmission is already taking place. Since the process has therefore missed the start of the transmission it will therefore receive an error value; this is the import of (B-RCVLATE). The remaining rules are straightforward; note that we have defined our rule in a way such that the evaluation of exposure checks require a τ-action.

The final set of rules, in Table 9.6, are structural. In particular (B-RESI) and (B-RESV) show how restricted channels are handled. Intuitively moves from the configuration $\Gamma \rhd \nu c : (n, v).W$ are inherited from the configuration $\Gamma[c \mapsto (n, v)] \rhd W$; here the network $\Gamma[c \mapsto (n, v)]$ is the same as Γ except that c has associated with it (temporarily) the information (n, v). However if this move mentions the restricted channel c then the inherited move is rendered as an internal action τ, (B-RESI). Moreover the information associated with the restricted channel in the residual is updated.

$$\frac{\Gamma \vdash c : \mathrm{exp}}{\Gamma \rhd \lfloor c?(x).P \rfloor Q \xrightarrow{\tau} \Gamma \rhd c[x].\{^{\mathsf{err}}/_{x}\}P}\ (\text{B-RcvLate}) \qquad \frac{}{\Gamma \rhd \tau.P \xrightarrow{\tau} \Gamma \rhd P}\ (\text{B-Tau})$$

$$\frac{\Gamma \vdash c : \mathrm{exp}}{\Gamma \rhd \langle c \rangle P, Q \xrightarrow{\tau} \Gamma \rhd \sigma.P}\ (\text{B-ExpThen}) \qquad \frac{\Gamma \vdash c : \mathrm{free}}{\Gamma \rhd \langle c \rangle P, Q \xrightarrow{\tau} \Gamma \rhd \sigma.Q}\ (\text{B-ExpElse})$$

Table 9.5: Intensional semantics: - internal activity

We end this Section by performing some sanity check on the intensional semantics.

The first properties we prove is that, in the labelled transition semantics, the update of channel environments is consistent with the action of the transition which is performed.

Proposition 9.4.1. [Properties of Channel Environments] Let $\Gamma \rhd W$ be a well-formed configuration.

1. Whenever $\Gamma \rhd W \xrightarrow{c!v} \Gamma' \rhd W'$, then $\Gamma' = \mathrm{upd}^{v}_{c}(\Gamma)$,
2. whenever $\Gamma \rhd W \xrightarrow{c?v} \Gamma' \rhd W'$, then $\Gamma' = \mathrm{upd}^{v}_{c}(\Gamma)$,
3. whenever $\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'$, then $\Gamma' = \Gamma$,
4. whenever $\Gamma \rhd W \xrightarrow{\sigma} \Gamma' \rhd W'$, then $\Gamma' = \Gamma \ominus 1$.

Proof. All the statements above can be proved by rule induction. Statement (1) is needed to prove (3) when dealing with the case of Rule (B-Snd − restr). An outline of the proof of Proposition 9.4.1(3) is provided in the Appendix, Page 238 □

Next we prove that input transitions are always enabled in configurations.

Lemma 9.4.2 (Receive enabled)**.** Let $\Gamma \rhd W$ be a well-formed configuration. Then

1. $\neg \mathrm{rcv}(W, c)$ implies $\Gamma \rhd W \xrightarrow{c?v} \Gamma' \rhd W'$ and $W' = W$
2. $\Gamma \vdash c : \mathrm{exp}$ implies $\Gamma \rhd W \xrightarrow{c?v} \Gamma' \rhd W'$ and $W' = W$
3. $\mathrm{rcv}(W, c)$ and $\Gamma \vdash c : \mathrm{free}$ iff there is W', $W' \neq W$, such that $\Gamma \rhd W \xrightarrow{c?v} \Gamma' \rhd W'$, for any v.

Proof. By transition induction and by inspection of the rules, (B-Rcv), (B-RcvFail) and (B-RcvPar). □

9.5 Properties of the Calculus

Below we report a number of basic properties of our labelled transition semantics. The first part of this Section is technical in its content, and may be skipped by the reader uninterested in details. Here we first prove some basic results about how channel environments affects, and are affected, by the execution of some action. Then we turn our attention to system terms in configurations, by analysing the structure we require from them to perform a given transition.

Finally, we apply the results developed in this Section to prove the *Harmony Theorem*, which states that the labelled transition semantics of CCCP is consistent with its reduction semantics .

An important property of channel environments is that only those channels which have free occurrences in a system term may affect the behaviour of a configuration.

Proposition 9.5.1. whenever $\Gamma \rhd W \xrightarrow{\lambda} \Gamma' \rhd W'$ with $\lambda \neq \sigma$, then for any c, n, v such that $c \notin \mathrm{fn}(W)$ and $\lambda \neq c?v$ it holds

$$\Gamma[c \mapsto (n, v)] \rhd W \xrightarrow{\lambda} \Gamma'[c \mapsto (n, v)] \rhd W'$$

(B-TauPar)
$$\frac{\Gamma \rhd W_1 \xrightarrow{\tau} \Gamma' \rhd W_1'}{\Gamma \rhd W_1 | W_2 \xrightarrow{\tau} \Gamma' \rhd W_1' | W_2}$$

(B-TimePar)
$$\frac{\Gamma \rhd W_1 \xrightarrow{\sigma} \Gamma' \rhd W_1' \quad \Gamma \rhd W_2 \xrightarrow{\sigma} \Gamma' \rhd W_2'}{\Gamma \rhd W_1 | W_2 \xrightarrow{\sigma} \Gamma' \rhd W_1' | W_2'}$$

(B-SumRcv)
$$\frac{\Gamma \rhd P \xrightarrow{c?v} \Gamma' \rhd P' \quad \mathrm{rcv}(P, c) \quad \Gamma \vdash c : \mathrm{free}}{\Gamma \rhd P + Q \xrightarrow{c?v} \Gamma' \rhd P'}$$

(B-Sum)
$$\frac{\Gamma \rhd P \xrightarrow{\lambda} \Gamma' \rhd P' \quad \lambda \in \{\tau, c!v\}}{\Gamma \rhd P + Q \xrightarrow{\lambda} \Gamma' \rhd P'}$$

(B-Then)
$$\frac{[\![b]\!] = \mathsf{true} \quad \Gamma \rhd P \xrightarrow{\lambda} \Gamma' \rhd P'}{\Gamma \rhd [b]P, Q \xrightarrow{\lambda} \Gamma' \rhd P'}$$

(B-Else)
$$\frac{[\![b]\!] = \mathsf{false} \quad \Gamma \rhd Q \xrightarrow{\lambda} \Gamma' \rhd Q'}{\Gamma \rhd [b]P, Q \xrightarrow{\lambda} \Gamma' \rhd P'}$$

(B-Rec)
$$\frac{\{\mathsf{fix}\, X.P/X\}P \xrightarrow{\lambda} W}{\Gamma \rhd \mathsf{fix}\, X.P \xrightarrow{\lambda} W}$$

(B-ResI)
$$\frac{\Gamma[c \mapsto (n, v)] \rhd W \xrightarrow{c!v} \Gamma' \rhd W'}{\Gamma \rhd \nu c : (n, v).W \xrightarrow{\tau} \Gamma'[c \mapsto \Gamma(c)] \rhd \nu c : \Gamma'(c).W}$$

(B-ResV)
$$\frac{\Gamma[c \mapsto (n, v)] \rhd W \xrightarrow{\lambda} \Gamma' \rhd W', \; c \notin \lambda}{\Gamma \rhd \nu c : (n, v).W \xrightarrow{\lambda} \Gamma'[c \mapsto \Gamma(c)] \rhd \nu c : ((n, v)).W}$$

Table 9.6: Intensional semantics: - structural rules

Proof. By rule induction on the proof of the transition $\Gamma \rhd W \xrightarrow{\lambda} \Gamma' \rhd W'$; see the Appendix, Page 239, for an outline. □

We also prove another property regarding channel environments, which will be needed later in the paper. Given two channel environments Γ_1, Γ_2, we say that they are *exposure consistent* (denoted as $\Gamma_1 \simeq_{\mathrm{exp}} \Gamma_2$) if $\Gamma_1 \vdash c : \mathrm{free}$ iff $\Gamma_2 \vdash c : \mathrm{free}$ for any channel c.

Proposition 9.5.2. Let Γ_1, Γ_2 be channel environments such that $\Gamma_1 \simeq_{\mathrm{exp}} \Gamma_2$. Let also W be a term such that $\Gamma_1 \rhd W$ is well-formed. Then, whenever $\Gamma_1 \rhd W \xrightarrow{\lambda} \Gamma_1 \rhd W'$, $\lambda \neq \sigma$, there exists Γ_2' such that $\Gamma_1' \simeq_{\mathrm{exp}} \Gamma_2'$ and $\Gamma_2 \rhd W \xrightarrow{\lambda} \Gamma_2' \rhd W'$.

Proof. By Rule induction, considering that the rules for non-timed actions only check whether a channel is exposed or not. □

Next we focus on configurations in which a given channel is restricted. Here we show that it is possible to infer the actions of such a configuration by looking at the configuration obtained by removing the channel restriction, and in which the contents of the channel environments have been updated to be consistent with the exposure information of the restricted channel.

Proposition 9.5.3. Suppose $\Gamma \rhd \nu c : (n, v).W \xrightarrow{\lambda} \Gamma' \rhd W'$; then $W' = \nu c : (n', v').W_1$ for some n', v', W_1 such that

1. if $\lambda = \tau$ then either
   - $\Gamma[c \mapsto (n, v)] \rhd W \xrightarrow{\tau} \Gamma'[c \mapsto (n', v')] \rhd W_1$, with $(n', v') = (n, v)$ or
   - $\Gamma[c \mapsto (n, v)] \rhd W \xrightarrow{c!} \Gamma'[c \mapsto (n', v')] \rhd W_1$, where $(n', v') = \mathrm{upd}_c^v(\Gamma[c \mapsto (n, v)])(c)$

2. if $\lambda = d?v$, with $d \neq c$, then $\Gamma[c \mapsto (n,v)] \rhd W \xrightarrow{d?v} \Gamma'[c \mapsto (n',v')] \rhd W_1$; further $(n',v') = (n,v)$
3. if $\lambda = c?v$ then $W_1 = W$ and $(n',v') = (n,v)$,
4. if $\lambda = d!w$ then $\Gamma[c \mapsto (n,v)] \rhd W \xrightarrow{d!w} \Gamma'[c \mapsto (n',v')] \rhd W_1$; further $(n',v') = (n,v)$
5. if $\lambda = \sigma$ then $\Gamma[c \mapsto (n,v)] \rhd W \xrightarrow{\sigma} \Gamma'[c \mapsto (n',v')] \rhd W_1$; here $(n',v') = (n,v) \ominus 1$[1].

Proof. By case analysis on the last rule applied in the proof of the transition $\Gamma \rhd \nu c : (n,v).W \xrightarrow{\lambda} \Gamma' \rhd W'$. □

This Proposition can be easily extended to configurations in which multiple channels are restricted.

Corollary 9.5.4. Suppose $\Gamma \rhd \nu l.W \xrightarrow{\lambda} \Gamma' \rhd W'$; then $W' = \nu l'.W_1$ for some l', W_1 such that

1. if $\lambda = \tau$ then either
   - $\Gamma[l] \rhd W \xrightarrow{\tau} \Gamma'[l'] \rhd W_1$, with $l' = l$ or
   - $\Gamma[c \mapsto (n,v)] \rhd W \xrightarrow{c!} \Gamma'[l'] \rhd W_1$, with c appearing in $\mathrm{ch}(l)$. Further, if $l = l_1 :: c : (n,v) :: l_2$, and c does not appear in $\mathrm{ch}(l_2)$, then $l' = l_1 :: c : (\mathrm{upd}^v_c([c \mapsto (n,v)])(c)) :: l_2$
2. if $\lambda = d?v$, and d does not appear in $\mathrm{ch}(l)$, then $\Gamma[l] \rhd W \xrightarrow{d?v} \Gamma'[l'] \rhd W_1$; further $l' = l \ominus 1$[2]
3. if $\lambda = c?v$ then $W_1 = W$, $l' = l$
4. if $\lambda = d!w$ for some $d \notin \mathrm{ch}(l)$, then $\Gamma[l] \rhd W \xrightarrow{d!w} \Gamma'[l'] \rhd W_1$; further $l' = l$
5. if $\lambda = \sigma$ then $\Gamma[l] \rhd W \xrightarrow{\sigma} \Gamma'[l'] \rhd W_1$; here $l' = l - 1$

Proof. By induction on the length of l, the inductive step being a simple application of Proposition 9.5.3. □

Despite being very technical, all the Propositions above are very useful when analysing what is the structure of a configuration that can perform a given action. We give an example by showing that whenever a configuration performs a an output action along a channel c, then it has a sending component of the form $c!\langle v\rangle.P + Q$, where c appears as a free name. Formally:

Proposition 9.5.5. Whenever $\Gamma \rhd W \xrightarrow{c!} \Gamma' \rhd W'$, then

- if $\Gamma(c)_{\mathbf{1}} = 0$, then $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(c!\langle v\rangle.P + Q | W_1)$ for some l, P, Q, W_1, W_1' such that
  1. c does not appear in $\mathrm{ch}(l)$
  2. $\Gamma \rhd \nu l.W_1 \xrightarrow{c?v} \Gamma' \rhd \nu l.W_1'$
  3. $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l.(\sigma^{\delta_v}.P | W_1')$.
- otherwise $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(c!\langle v\rangle.P + Q | W_1)$ for some l, c, v, P, Q, and W_1 such that $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l.(\sigma^{\delta_v}.P | W_1)$ and c does not appear in $\mathrm{Ch}(l)$.

Proof. See the Appendix, Page 240. □

Similar propositions can be proved in an analogous way when dealing with input actions and internal actions.

Proposition 9.5.6. Let $\Gamma \rhd W \xrightarrow{c?v} \Gamma' \rhd W'$. Then $\Gamma \rhd W \equiv \nu l.W_1$, $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l.W_1'$ for some l, W_1, W_1' such that

1. if c does not appear in $\mathrm{ch}(l)$ and $\Gamma(c)_1 = 0$ then $\Gamma \rhd W_1 \equiv \Gamma \rhd \nu l.(\Pi_{j\in J}(\lfloor c?(x).P_j \rfloor Q_j + R_j) | W_2)$ for some l, W_1, a finite index set J and collections of processes $\{P_j\}_{j\in J}, \{Q_j\}_{j\in J}, \{R_j\}_{j\in J}$ such that

[1] here, with an abuse of notation, the operation $\ominus$ has been restricted from channel environments to elements of the form (n,v)

[2] Here we made another abuse of notation, by extending the operation $\ominus$ to list of elements of the form (n,v); however, defining how such an operator affects such a list is straightforward.

(a) $\neg \mathrm{rcv}(W_2, c)$

(b) $\Gamma' \rhd W_1' \equiv \Gamma' \rhd \nu l.(\Pi_{j\in J}(c[x].P_j)|W_2)$.

2. otherwise $W_1' = W_1$.

Due to channel restrictions, τ-actions are the most difficult to deal with. Here we present the four different scenarios that can cause a τ transition to happen.

Proposition 9.5.7. Let $\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'$. Then, either one of the following is true:

1. $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(\tau.P + Q|W_1)$ for some l, P, Q, W_1 such that $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l.(P|W)$
2. $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(\lfloor c?(x).P \rfloor Q + R|W_1)$, for some l, c, P, Q, R and W_1 such that
   - c does not appear in $\mathrm{ch}(l)$
   - $\Gamma(c)_{\mathbf{1}} > 0$
   - $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l.(c[x].\{\mathsf{err}/x\}P|W_1)$
3. $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(\lfloor c?(x).P \rfloor Q + R|W_1)$, for some l, c, P, Q, R and W_1 such that
   - $l = l_1 :: c : (n, v) :: l_2$ for some l_1, l_2, n, v such that c does not appear in l_2 and $n > 0$
   - $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l(c[x].\{\mathsf{err}/x\}P|W_1)$
4. $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(c\,!\langle v\rangle.P + Q|W_1)$ for some l, c, P, Q, l', W_1, W_1' and Γ'' such that
   - $l = l_1 :: c : (n, v) :: l_2$ for some l_1, l_2 such that c does not appear in $\mathrm{ch}(l_2)$
   - $\Gamma[l] \rhd W_1 \xrightarrow{c?v} \Gamma'' \rhd W_1'$
   - $l' = l_1 :: c : \Gamma''(c) :: l_2$
   - $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l'.(\sigma^{\delta_v}.P|W_1')$.

Let us now turn our attention to time delays. The following property is not used in this Section, but it will be useful later.

Proposition 9.5.8. Let Γ_1, Γ_2 be two channel environments such that $\Gamma_1 \simeq_{\mathrm{exp}} \Gamma_2$ and $\Gamma_1 \vdash c$ deliver v if and only if $\Gamma_2 \vdash c$ deliver v. Then, whenever $\Gamma_1 \rhd W \xrightarrow{\sigma} \Gamma_1' \rhd W_1$ and $\Gamma_2 \rhd W \xrightarrow{\sigma} \Gamma_2' \rhd W_2$, it holds $W_1 = W_2$.

Proof. By rule induction on the proof of the transition $\Gamma_1 \rhd W \xrightarrow{\sigma} \Gamma_1' \rhd W_1$. □

Next we deal with parallel components in non-restricted environments.

Proposition 9.5.9. [Parallel components] Let $\Gamma \rhd W_1|W_2$ be well-formed configurations.

1. if $\Gamma \rhd W_1|W_2 \xrightarrow{\tau} \Gamma \rhd W$ if and only if either
   - $W = W_1'|W_2$, with $\Gamma \rhd W_1 \xrightarrow{\tau} \Gamma \rhd W_1'$ or
   - $W = W_1|W_2'$, with $\Gamma \rhd W_2 \xrightarrow{\tau} \Gamma \rhd W_2'$.
2. $\Gamma \rhd W_1|W_2 \xrightarrow{c?v} \Gamma' \rhd W$ if and only if there are W_1' and W_2' such that $\Gamma \rhd W_1 \xrightarrow{c?v} \Gamma' \rhd W_1'$, $\Gamma \rhd W_2 \xrightarrow{c?v} \Gamma' \rhd W_2'$ and $W = W_1'|W_2'$.
3. $\Gamma \rhd W_1|W_2 \xrightarrow{c!} \Gamma' \rhd W$ if and only if there are W_1' and W_2' such that
   - $\Gamma \rhd W_1 \xrightarrow{c!} \Gamma' \rhd W_1'$, $\Gamma \rhd W_2 \xrightarrow{c?v} \Gamma' \rhd W_2'$ and $W = W_1'|W_2'$
   - or $\Gamma \rhd W_1 \xrightarrow{c?v} \Gamma' \rhd W_1'$, $\Gamma \rhd W_2 \xrightarrow{c!} \Gamma' \rhd W_2'$ and $W = W_1'|W_2'$.
4. $\Gamma \rhd W_1|W_2 \xrightarrow{\sigma} \Gamma' \rhd W$ if and only if there are W_1' and W_2' such that $\Gamma \rhd W_1 \xrightarrow{\sigma} \Gamma' \rhd W_1'$, $\Gamma \rhd W_2 \xrightarrow{\sigma} \Gamma' \rhd W_2'$ and $W = W_1'|W_2'$

Proof. The proof can be performed by structural induction on the proof of a transition of the form $\Gamma \rhd W_1|W_2 \xrightarrow{\lambda} W$. The technical details are similar to those of all the other proofs in this Section. □

The last property we need establishes that transitions of the labelled transition semantics are preserved in structurally congruent configurations.

Proposition 9.5.10. Suppose $\Gamma \rhd W_1 \xrightarrow{\lambda} \Gamma' \rhd W_1'$, and let $\Gamma \rhd W_2$ such that $W_1 \equiv W_2$. Then there exists $W_2' \equiv W_1'$ such that $\Gamma \rhd W_2 \xrightarrow{\lambda} \Gamma' \rhd W_2'$.

Proof. By case analysis on the clauses defined for structural congruence, then by structural induction on the derivation $\Gamma \rhd W_1 \xrightarrow{\lambda} \Gamma' \rhd W_1'$. □

We have proved all the results that we need to show that the reduction semantics is consistent with the labelled transition semantics.

Theorem 9.5.11. [Harmony Theorem] Let $\Gamma \rhd W$ be a well-formed configuration.

1. If $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$ then either $\Gamma \rhd W \xrightarrow{\tau} \equiv \Gamma' \rhd W'$ or $\Gamma \rhd W \xrightarrow{c!v} \equiv \Gamma' \rhd W'$.
2. If $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma' \rhd W'$ then $\Gamma \rhd W \xrightarrow{\sigma} \equiv \Gamma \rhd W'$.
3. If either $\Gamma \rhd W \xrightarrow{\tau} \Gamma \rhd W'$ or $\Gamma \rhd W \xrightarrow{c!v} \Gamma \rhd W'$ then $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$.
4. If $\Gamma \rhd W \xrightarrow{\sigma} \Gamma' \rhd W'$ then $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma' \rhd W'$.

Proof. Each of these Statement is proved separately. Statements (1) and (2) can be proved by performing a rule induction on the length of the proof of the reduction $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$, and then using the rules of the labelled transition semantics to build a transition of the form $\Gamma \rhd Q \xrightarrow{\lambda} \equiv \Gamma' \rhd W'$.

For Statement (3), we distinguish between the cases $\Gamma \rhd W \xrightarrow{c!v} \Gamma' \rhd W'$ and $\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'$. In the former case, Proposition 9.5.5 ensures that $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(!\langle c\rangle.\nu P + Q|W_1)$. The proof is continued by performing an induction on the length of l, using propositions 9.5.6 and 9.5.9(2). The case where $\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'$ is treated similarly.

Finally, Statement (4) is proved by performing a rule induction on the proof of the derivation $\Gamma \rhd W \xrightarrow{\sigma} \Gamma' \rhd W'$.

□

9.6 Related Work

The calculus that we have been developed in this Chapter has been mainly inspired by the *Timed Calculus for Wireless Systems* (*TCWS*), defined in [47]. The main differences between our calculus and TCWS can be summarised as follows:

- in our calculus the network topology has assumed to be flat, while in TCWS a network topology is embedded in the syntax of a network. Having a flat topology allowed us to develop a reduction semantics in our framework, which has not been provided for TCWS; another advantage of having a flat topology concerns the proof techniques for behavioural theories. Since this is the topic of the next chapter, we defer this discussion for the moment,
- we use channel environments to store the information of values stored inside a channel. As a consequence, our syntax for describing a network is considerably simpler than the one of TCWS's processes. In particular, it is not necessarily longer to use active senders to model a station which is currently transmitting a value; further, active receivers do not need to come with a semantic tag corresponding to the value being received by a station.

Other papers in which collisions in wireless communications are taken into account are [48, 43]. This calculus is considerably different from ours; specifically, no concept of time is used by the authors. Instead, they model the transmission of a value by defining two different actions; the first one corresponds to a starting communication, while the second one models the end of a transmission which was already taking place.

Chapter 10

Barbed Equivalence and Full Abstraction

In this Chapter we define a behavioural equivalence for CCCP processes, and we prove a full-abstraction result for it.

The notion of behavioural equivalence we propose is based on *Milner* and *Sangiorgi*'s reduction barbed congruence [49]. The definition of barbed congruence strongly relies on two crucial concepts: a reduction semantics to describe how a system evolves, which we already provided in Chapter 9 and a notion of observable (or barb) which says what can be observed in a system. As we will see in this Chapter, an observation on a channel c corresponds to detecting that such a channel is exposed.

At least intuitively, two terms are barbed congruent if they have the same observables in all the possible evolutions under all possible contexts. In the case of CCCP, we only consider contextuality with respect to the parallel operator |, so that the term *barbed equivalence* is more appropriate.

Due to the quantification over all contexts, it is often difficult to prove directly that two networks are barbed equivalent; instead, we propose an extensional semantics for CCCP terms whose actions coincides with those activities which can be observed by a context and we prove that, if we focus on well-timed processes, our notion of barbed equivalence coincides with *weak bisimulation* in the proposed extensional semantics.

This Chapter is organised as follows; in Section 10.1 we define barbed equivalence for well-timed networks. In Section 10.2 we present our extensional semantics and we define weak bisimulation.

In Section 10.3.1 we prove that our bisimulation relation is contextual with respect to the parallel operator; this result can be used to show that weak bisimulation is a sound proof technique for showing that two configurations are barbed equivalent.

We show also that whenever two configurations are barbed equivalent it is possible to find a weak bisimulation between them; this is done by showing that there exists a distinguishing context for each of the actions of the extensional semantics. This topic is covered in Section 10.3.2.

Then we provide a simple application of our proof method for showing that two configurations are barbed equivalent; this is done in Section 10.4.

We conclude the Chapter with a brief comparison between related works in Section 10.5.

10.1 Barbed Equivalence

In this Section we define our contextual behavioural equivalence; as we already mentioned, this is based on the notion of barbed congruence [49]. In order to obtain a contextual behavioural equivalence by using their approach we have to decide on an appropriate notion of *barb*, or *observation* on systems.

In the case of *CCS* a process has a barb over an action a if it can synchronise with the external environment along such a channel. In CCCP, however, transmission is timed; at least intuitively the external environment should be able to observe whether a process in a configuration is transmitting some value along a (free) channel c; as we will see later, in fact, this task can be accomplished by simply performing an exposure check for the

channel c; however, this result holds only for well-timed configurations. This intuition leads to a notion of barb for CCCP configurations which is based on the exposure state of the channel environment, rather than on the structure of the term of such a configuration.

Definition 10.1.1 (Barbs)**.** Let $\Gamma \rhd W$ be a well-formed configuration. We say it has a *strong barb on c*, written $\Gamma \rhd W \downarrow_c$, if $\Gamma \vdash c : \text{exp}$. One of the main advantages of Definition 10.1.1 is that it is independent from the Syntax of CCCP terms; that is, changing the syntax of the terms does not require to change the definition of the barbs.

We write $\Gamma \rhd W \Downarrow_c$, a *weak barb*, if there exists a configuration $\Gamma' \rhd W'$ such that $\Gamma \rhd W \rightarrowtriangle^* \Gamma' \rhd W'$ and $\Gamma' \rhd W' \downarrow_c$.

Once we have established a suitable notion of barb, we can define our behavioural equivalence by following the approach of [49]. Here the authors define their behavioural equivalence to be the largest relation which is symmetric, contextual, barb preserving and reduction closed.

Definition 10.1.2. Let $\mathcal{R}$ be a relation over well-formed configurations.

(1) $\mathcal{R}$ is said to be *barb preserving* if $\Gamma_1 \rhd W_1 \Downarrow_c$ implies $\Gamma_2 \rhd W_2 \Downarrow_c$, whenever $(\Gamma_1 \rhd W_1)\, \mathcal{R}\, (\Gamma_2 \rhd W_2)$.

(2) It is *(untimed) reduction-closed* if $(\Gamma_1 \rhd W_1)\, \mathcal{R}\, (\Gamma_2 \rhd W_2)$ and $\Gamma_1 \rhd W_1 \rightarrowtriangle \Gamma_1' \rhd W_1'$ imply there is some $\Gamma_2' \rhd W_2'$ such that $\Gamma_2 \rhd W_2 \rightarrowtriangle^* \Gamma_2' \rhd W_2'$ and $(\Gamma_1' \rhd W_1')\, \mathcal{R}\, (\Gamma_2' \rhd W_2')$.

(3) It is *contextual* if $\Gamma_1 \rhd W_1 \;\mathcal{R}\; \Gamma_2 \rhd W_2$, implies $\Gamma_1 \rhd (W_1 | W) \;\mathcal{R}\; \Gamma_2 \rhd (W_2 | W)$ for all processes W such that both $\Gamma_1 \rhd W_1 | W$ and $\Gamma_2 \rhd W_2 | W$ are well-formed. □

□

Note that contextuality is not defined for the restriction operator $\nu c : n, v$. When contextuality in a barbed congruence relation is not defined for all the operators of the language the term *barbed equivalence* is used instead of barbed congruence.

With these concepts we now have everything in place for the standard definition of contextual equivalence between systems:

Definition 10.1.3. *Reduction barbed equivalence*, written $\simeq$, is the largest symmetric relation over well-formed configurations which is barb preserving, reduction-closed and contextual. □

In the following the notion of a fresh channel will be important; we say that c is *fresh* for the configuration $\Gamma \rhd W$ if it does not occur free in W and $\Gamma \vdash c : \text{free}$. Note that if $\Gamma \rhd W$ is well-formed then we can always pick some fresh channel for it.

Before showing some examples that illustrate how reduction barbed equivalence is applied to configurations, let us show that barbs are indeed observables; that is, it is possible to define term T equipped with two fresh channels fail, eureka such that any *well-timed* configuration C can reach (via a sequence of transitions) a configuration C' which has a strong (hence also a weak) barb on eureka, but it has no weak barb on fail.

Theorem 10.1.4 (Detecting Barbs)**.** Let $\Gamma \rhd W$ be a well-timed configuration and consider the configuration $C | T$, where

$$T = \text{fix}\, X.((\langle c \rangle \text{eureka}!\langle \text{ok} \rangle, \text{fail}!\langle \text{no} \rangle + \tau.X) + \text{fail}!\langle \text{no} \rangle)$$

and $\delta_{\text{ok}} = \delta_{\text{no}} = 1$. Then $\Gamma \rhd W \Downarrow_c$ if and only if $\Gamma \rhd W | T \rightarrowtriangle^* C$ for some configuration C such that $C \downarrow_{\text{eureka}}$, $C \not\Downarrow_{\text{fail}}$.

Proof. The proof of this statement is very technical. See Appendix C, Section C.2, Page 240, for a detailed outline. □

Remark 10.1.5. Note that Theorem 10.1.4 holds only for well-timed configurations. In fact the test T used in the Theorem to detect a barb on a channel c is not valid if ill-timed processes are considered.

For example, consider the process BAD = $\mathsf{fix}\, X.(\tau.X)$; we have already shown in Chapter 9 that this process is ill-timed; let also Γ be a configuration such that $\Gamma \vdash c : \mathrm{exp}$. The configuration $\Gamma \rhd \mathrm{BAD}|T$ has no weak barb on channel eureka, for the testing component T requires time to pass before broadcasting a value along such a channel (hence making it exposed) however, process BAD prevents time to pass. Since time can pass in composed processes only if it passes in all of its components, it follows that time cannot pass for $\mathrm{BAD}|T$, hence $\Gamma \rhd \mathrm{BAD}|T$ can never reach a configuration in which channel eureka is exposed.

Henceforth we assume that a configuration is both well-formed and well-timed, unless otherwise stated.

An interesting property of barbed equivalent configuration is that they are exposure equivalent; this is because we can always test for the exposure state of a (non-restricted) channel in a configuration.

Proposition 10.1.6. Suppose $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$. Then $\Gamma_1 \simeq_{exp} \Gamma_2$.

Proof. We prove that if $\Gamma_1 \vdash c : \mathrm{free}$ then $\Gamma_2 \vdash c : \mathrm{free}$; the result follows from symmetry of $\simeq$.

To this end, let fail and eureka be two fresh channels, and consider the testing station

$$T = \langle c \rangle \mathsf{nil}, \mathrm{eureka}!\langle \mathrm{ok} \rangle | \sigma.(\mathrm{fail}!\langle \mathrm{no} \rangle + \tau.\mathsf{nil})$$

where $\delta_{\mathrm{ok}} = \delta_{\mathrm{no}} = 1$.

For this configuration, it is possible to prove that $\Gamma \rhd W|T \rightarrow\!\!\triangleright_u^* C$ for some configuration $C = \Gamma' \rhd W'$ such that $C \downarrow_{\mathrm{eureka}}, C \not\Downarrow_{\mathrm{fail}}$ if and only if $\Gamma' \vdash c : \mathrm{free}$. Note that this is an untimed weak reduction; the proof of this statement is similar in details (though less technical) to that of Theorem 10.1.4.

Now consider two configurations $\Gamma_1 \rhd \simeq \Gamma_2 \rhd W_2$, and suppose $\Gamma_1 \vdash c : \mathrm{free}$. For the configuration $\Gamma_1 \rhd W_1$, we have that $\Gamma_1 \rhd W_1|T \rightarrow\!\!\triangleright^* C$, where $C \downarrow_{\mathrm{eureka}}, C \not\Downarrow_{\mathrm{fail}}$. For $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$ it follows that $\Gamma_2 \rhd W_2|T \rightarrow\!\!\triangleright^* C'$, where $C' \Downarrow_{\mathrm{eureka}}$ and $C' \not\Downarrow_{\mathrm{fail}}$. By unfolding the definition of barb, we obtain that $C' \rightarrow\!\!\triangleright^* C''$ (hence $\Gamma_2 \rhd W_1|T \rightarrow\!\!\triangleright^* C''$) and $C'' \downarrow_{\mathrm{eureka}}, C'' \not\Downarrow_{\mathrm{fail}}$. By the statement above, we have that $\Gamma_2 \rhd W_2 \rightarrow\!\!\triangleright_u^* \Gamma_2' \rhd W_2'$ for some channel environment Γ_2' such that $\Gamma_2' \vdash c : \mathrm{free}$.

Now note that if $\Gamma_2' \vdash c : \mathrm{free}$ then $\Gamma_2'(c) = (0, \mathbf{x})$. By a generalisation of Proposition 9.3.4 to weak untimed reduction, whose proof is straightforward, we also have that $\Gamma_2(c) \leq \Gamma'(c)$, hence $\Gamma_2(c) = (0, \mathbf{x})$, or equivalently $\Gamma \vdash c : \mathrm{free}$, as we wanted to prove.

□

In the remainder of this Section we explore via examples the implications of this definition.

Example 10.1.7. Let us assume that $\Gamma \vdash c : \mathrm{free}$. Then it is easy to see that

$$\Gamma \rhd c\,!\langle v_o \rangle.P \not\simeq \Gamma \rhd c\,!\langle v_1 \rangle.P \tag{10.1}$$

under the assumption that v_o and v_1 are different values. For let T be the testing context

$$\lfloor c?(x).[x = v_o]\mathrm{eureka}!\langle \mathrm{ok} \rangle, \mathsf{nil} \rfloor \mathsf{nil}$$

where eureka is fresh, and ok is some arbitrary value. Then $\Gamma \rhd c\,!\langle v_o \rangle.P|T$ has a weak barb on eureka, for such a channel eventually becomes exposed in the unique computation of configuration $\Gamma \rhd !\langle c \rangle.v_o P|T$; in fact, the only reduction (up-to structural congruence) defined for the network is given by

$$\Gamma \rhd !\langle c \rangle.v_o P|T \rightarrow\!\!\triangleright \Gamma' \rhd \sigma^{\delta_{v_o}}.P|c[x].[x = v_o]\mathrm{eurekaok}!\langle \mathsf{nil} \rangle, \mathsf{nil}$$

which can be derived by applying Rule (R-NoColl) of the reduction semantics. Here $\Gamma' = \mathrm{upd}_{cv}^{=}(\Gamma)\Gamma[c \mapsto (\delta^{v_o}, v_0)]$. We can now apply Rule (R-Time) to the latter configuration, and iterate the procedure to the resulting configuration for exactly δ_o^v times, to obtain the sequence of reductions

$$\Gamma \rhd !\langle c \rangle.v_o P|T \rightarrow\!\!\triangleright \Gamma' \rhd \sigma^{\delta_{v_o}}.P|c[x].[x = v_o]\mathrm{eurekaok}!\langle \mathsf{nil} \rangle, \mathsf{nil} \rightarrow\!\!\triangleright \cdots \rightarrow\!\!\triangleright \Gamma' \ominus \delta_v^o \rhd P|[v_o = v_o]\mathrm{eureka}\mathit{arb}!\langle \mathsf{nil} \rangle, \mathsf{nil}$$

A final application of Rule (R-NoColl) and (R-Struct) leads to the reduction

$$\Gamma' \ominus \delta_v^o \rhd P|[v_o = v_o]\text{eurekaok}!\langle \mathsf{nil}\rangle, \mathsf{nil} \twoheadrightarrow \Gamma'' \rhd P|\sigma^{\delta_{\text{ok}}}$$

where $\Gamma'' \vdash \text{eureka} : \text{exp}$. We have proved that $\Gamma \rhd !\langle c\rangle.v_o P|T \twoheadrightarrow^* \Gamma'' \rhd P|\sigma^{\delta_{\text{ok}}}$ and the latter configuration has a strong barb on channel eureka; therefore $\Gamma \rhd !\langle c\rangle.v_o P|T \Downarrow_{\text{eureka}}$.

In a similar way we can prove that $\Gamma \rhd !\langle c\rangle.v_1 P|T$ has a unique computation in whose configurations channel eureka is always free. Therefore the former configuration has no (weak) barb on channel eureka. Since $\simeq$ is contextual and barb preserving, (10.1) above follows. □

Example 10.1.8. Let again Γ be a channel environment such that $\Gamma \vdash c$: free, and let v_o, v_1 two values such that $\delta^{v_o} = \delta_i^v$.

The test we have used in Example 10.1.7 cannot be used to test to distinguish between the configurations $\Gamma \rhd Q_1$ and $\Gamma \rhd Q_2$, where

$$Q_1 = c!\langle v_o\rangle | c\,!\langle v_1\rangle.P \qquad \text{and} \qquad Q_2 = c!\langle v_1\rangle | c\,!\langle v_0\rangle.P$$

In both these configurations a collision will occur on the channel c and a station such as T above will only ever receive the error value err. So there is reason to hope that

$$\Gamma \rhd Q_1 \;\simeq\; \Gamma \rhd Q_2$$

However we must wait for the next section for proof techniques for establishing such equivalences. □

Examples 10.1.7 and 10.1.8 show that the broadcast the content of a value being broadcast along a free channel is not important, for a context can only check the content of a transmission along a channel when the latter switches from being exposed to being free (that is, the transmission has terminated).

A priori reductions ignore the passage of time, and therefore one might suspect that reduction barbed congruence is impervious to the precise timing of activities. But the next example demonstrates that this is not the case.

Example 10.1.9. [Observing the passage of time] Consider the two processes

$$Q_1 = c!\langle v_o\rangle \qquad\qquad Q_2 = \sigma.Q_1$$

and again let us assume that $\Gamma \vdash c$: free. There is very little difference between the behaviour of $\Gamma \rhd Q_1$ and $\Gamma \rhd Q_2$; both will transmit (successfully) the value v_o, although the latter is a little slower. However this slight difference can be observed. Consider the test T defined by

$$\langle c\rangle\text{eureka}!\langle \text{ok}\rangle, \mathsf{nil}$$

Then $C_2 = \Gamma \rhd (Q_1|T)$ has a weak barb on eureka. To see this it just suffices to note that we have the reduction step

$$\Gamma \rhd (Q_1|T) \twoheadrightarrow_u \Gamma \rhd (Q_1|\sigma.\text{eureka}!\langle \text{ok}\rangle)$$

However, the unique possible transition for $C_2 = \Gamma \rhd (Q_2|T)$ is of the form $C_2 \twoheadrightarrow C_2' = \Gamma_2 \rhd \sigma.\mathsf{nil}|\sigma.\mathsf{nil}$ in which the transmission along c is initiated. This is inferred with the aid of Rule (B-()$expElse$)[1] Now it is trivial to note that $C_2 \not\Downarrow_{\text{eureka}}$, so that $C_1 \not\simeq C_2$.

For the test to work correctly it is essential that $\Gamma \vdash c$: exp. Indeed we will later see that if $\Gamma' \vdash c$: exp then $\Gamma' \rhd Q_1 \simeq \Gamma' \rhd Q_2$. □

[1] Recall that the reduction semantics and the Labelled transition Semantics coincide, as stated in the Harmony Theorem.

$$\frac{\Gamma \rhd W \xrightarrow{c?v} \Gamma' \rhd W'}{\Gamma \rhd W \overset{c?v}{\longmapsto} \Gamma' \rhd W'}\ (\text{B-INPUT}) \qquad \frac{\Gamma \rhd W \xrightarrow{\sigma} \Gamma' \rhd W'}{\Gamma \rhd W \overset{\sigma}{\longmapsto} \Gamma' \rhd W'}\ (\text{B-TIME})$$

$$\frac{\Gamma \rhd W \xrightarrow{c!v} \Gamma' \rhd W'}{\Gamma \rhd W \overset{\tau}{\longmapsto} \Gamma' \rhd W'}\ (\text{B-SHH}) \qquad \frac{\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'}{\Gamma \rhd W \overset{\tau}{\longmapsto} \Gamma' \rhd W'}\ (\text{B-TAUEXT})$$

$$\frac{\Gamma \vdash c : \text{free}}{\Gamma \rhd W \overset{\iota(c)}{\longmapsto} \Gamma \rhd W}\ (\text{B-FREE}) \qquad \frac{\Gamma \vdash c \text{ deliver } v \quad \Gamma \rhd W \xrightarrow{\sigma} \Gamma' \rhd W'}{\Gamma \rhd W \overset{\gamma(c,v)}{\longmapsto} \Gamma \rhd W'}\ (\text{B-DELIVER})$$

Table 10.1: Extensional actions

Behind this example is the general principle that reduction barbed congruence is actually sensitive to the passage of time; see Proposition 10.3.8 in Section 10.3.2. This example also emphasises the power in the ability to test for channel exposure.

However the precise transmission time associated with channels in a network is not preserved by contextual equivalence.

10.2 Extensional Semantics

The intention here is to give an co-inductive characterisation of the contextual equivalence $\simeq$ between (well-formed and well-timed) configurations, in terms of a standard bisimulation equivalence over some extensional LTS. We first discuss the extensional actions used to define this LTS, then we recall the standard definition of (weak) bisimulation over an arbitrary LTS and then illustrate its application to the calculus CCCP via the LTS induced by the extensional actions.

10.2.1 Extensional Actions

The question here is what activity of a wireless system is observable externally; Example 10.1.9 indicates that the passage of time is observable. From Lemma 9.4.2, we know that all systems are always ready to receive transmissions, but we will have to take into account the effect of these transmissions. In contrast the discussion in Example 10.1.7 indicates that, due to the possibility of collisions, the treatment of transmissions, the actions $\xrightarrow{c!v}$ is more subtle. It turns out that the transmission itself is not important; instead we must take into consideration the successful delivery of the transmitted value.

In Table 10.1 we give the rules defining the extensional actions, $C \overset{\alpha}{\longmapsto} C'$, which can take one of the forms:

- Input: $C \overset{c?v}{\longmapsto} C'$. This is inherited directly from the intensional semantics.
- Time: $C \overset{\sigma}{\longmapsto} C'$, also inherited from the intensional semantics.
- Internal: $C \overset{\tau}{\longmapsto} C'$. Internal activities correspond (up-to structural congruence), to reduction steps; we know from the Harmony Theorem (Theorem 9.5.11) that reductions are captured by both internal and broadcast transitions in the labelled transition semantics.
- Delivery: $C \overset{\gamma(c,v)}{\longmapsto} C'$. This corresponds to the successful delivery of the value v which was in transmission along the channel c.
- Free: $C \overset{\iota(c)}{\longmapsto} C$, a predicate indicating that channel c is not exposed, and therefore ready to start a potentially successful transmission.

Remark 10.2.1. The reader could be surprised to know that output transitions are modelled in the extensional semantics as internal activities. While this should not be surprising in the case in which a broadcast is performed along a channel which is already exposed (recall that, in the labelled transition semantics, a value broadcast along an already exposed channel does not affect the other parallel components in a term, but it only affects the channel environment), there is still the case in which a broadcast is fired along a non-exposed channel. In this case other stations can detect the broadcast and start receiving.

Indeed, a broadcast which is fired along a non-exposed channel consists of an activity which can be detected. However, it can be modelled in our extensional semantics by using the novel action $\iota(c)$. In fact, whenever $\Gamma \vdash c : \text{free}$ and $\Gamma \rhd W \xrightarrow{c!v} \Gamma' \rhd W'$, it holds that $\Gamma' \vdash c : \text{exp}$. The intensional transition above corresponds, in the extensional semantics, to the sequence of transitions $\Gamma \rhd W \xmapsto{\iota(c)} \Gamma \rhd W$, $\Gamma \rhd W' \xmapsto{\tau} \Gamma' \rhd W'$ for which $\Gamma' \rhd W' \xmapsto{\iota(c)}\!\!\!\!\!\!/\;\;$.

Conversely, if we have a sequence in the extensional semantics as the one above, then we are sure that the transition $\Gamma \rhd W \xmapsto{\tau} \Gamma' \rhd W'$ transition corresponds to a broadcast fired along a channel. In fact, if it were a τ-action, it would hold $\Gamma = \Gamma'$ by Proposition 9.4.1(3). Since $\Gamma \vdash c : \text{free}$ then we cannot have $\Gamma \vdash c : \text{exp}$.

10.2.2 Bisimulation Equivalence

Weak extensional actions are defined as usual, as $C \xmapsto{\tau}{}^{*} \xmapsto{\alpha} \xmapsto{\tau}{}^{*} C'$, for $\alpha \neq \tau$, for which we use the notation $C \overset{\hat{\alpha}}{\Longmapsto} C'$. For $\alpha = \tau$, we abbreviate $C \overset{\hat{\tau}}{\Longmapsto} C'$ with $C \Longmapsto C'$, to denote $C \xmapsto{\tau}{}^{*} C'$. We now have the standard definition of weak bisimulation equivalence in the resulting LTS which for convenience we recall.

Definition 10.2.2. Let $\mathcal{R}$ be a binary relation over configurations. We say that $\mathcal{R}$ is a bisimulation if for every extensional action α, whenever $C_1 \,\mathcal{R}\, C_2$

(i) $C_1 \xmapsto{\alpha} C_1'$ implies $C_2 \overset{\hat{\alpha}}{\Longmapsto} C_2'$, for some C_2', satisfying $C_1' \,\mathcal{R}\, C_2'$

(ii) conversely, $C_2 \xmapsto{\alpha} C_2'$ implies $C_1 \overset{\hat{\alpha}}{\Longmapsto} C_1'$, for some C_1', such that $C_1' \,\mathcal{R}\, C_2'$.

We write $C_1 \approx C_2$, if there is a bisimulation $\mathcal{R}$ such that $C_1 \,\mathcal{R}\, C_2$. □

We already pointed out that our aim is to show that in a well-timed setting weak bisimulation and weak barbed equivalence coincide. Next we show that the actions we have introduced in the extensional semantics, namely $\iota(c)$ and $\Gamma(c, v)$, are necessary in order to show that weak bisimilarity is contextual.

Example 10.2.3. [On the rule (B-FREE)] Suppose that we drop the rule (B-FREE) in the extensional semantics; consider the configurations

$$\begin{aligned} \Gamma_1 \rhd W_1 &= \tau.\mathsf{nil} \\ \Gamma_2 \rhd W_2 &= c!\langle v \rangle \end{aligned}$$

where $\Gamma_1(c) = (1, v)$, $\Gamma_2(c) = (0, \cdot)$.

It is straightforward to note that $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$; in fact, recall that an output action in the intensional semantics always correspond to a τ action in its extensional counterpart[2].

However, we also have that $\Gamma_1 \rhd W_1 \not\approx \Gamma_2 \rhd W_2$; this can be proved by simply exhibiting a context that distinguishes the two configurations above. To this end, consider the system $T = [\exp(c)]\mathsf{nil}, \text{eureka}!\langle \text{ok} \rangle$. It is immediate to note that $\Gamma_2 \rhd W_2 | T$ has a barb on the channel eureka, while $\Gamma_1 \rhd W_1 | T$ has not. □

Example 10.2.4. [On the rule (B-DELIVER)] Suppose that we drop rule (B-DELIVER) in the extensional semantics. Let $\Gamma_2 \rhd W_2$ be the configuration of Example 10.2.3, and consider the configuration

$$\Gamma_2 \rhd W_3 = c!\langle w \rangle.$$

[2] That is, the LTS generated by these two configurations, with respect to the extensional semantics, are isomorphic

In this case it is easy to exhibit a bisimulation between $\Gamma_2 \rhd W_2$ and $\Gamma_2 \rhd W_3$; however, if we consider the system $T' = c(x).[x = v]\text{eureka}!\langle \text{ok} \rangle, \text{nil}$ we find that $\Gamma_2 \rhd W_2$ has a barb on eureka, while $\Gamma_2 \rhd W_3$ has not. Thus, in this case it holds that $\Gamma_2 \rhd W_2 \approx \Gamma_2 \rhd W_3$, but $\Gamma_2 \rhd W_2 \not\approx \Gamma_2 \rhd W_3$. □

10.3 Full Abstraction

In this section we show that the coinductive proof method based on the bisimulations of the previous section are both sound and complete with respect to the contextual equivalence of Section 10.1. In this first sub-section we give the non-trivial proof that bisimulation equivalence is contextual; this requires the development of some auxiliary results on the composition of extensional actions. From this contextuality result the soundness of bisimulations for contextual equivalence follows easily. The second section is devoted to the converse: if two configurations are related by the contextual equivalence there is also some bisimulation which relates them.

10.3.1 Soundness

We begin this Section by showing that two bisimilar configurations are also exposure equivalent. This property will be needed in the proof of soundness to distinguish whether an extensional τ-action corresponds to a broadcast fired along a non-exposed channel.

Lemma 10.3.1. [Channel exposure wrt $\approx$] Whenever $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$ then $\Gamma_1 \vdash c : \text{free}$ if and only if $\Gamma_2 \vdash c : \text{free}$.

The proof of Lemma 10.3.1 relies on the following result:

Lemma 10.3.2. Whenever $\Gamma \rhd W \stackrel{\tau}{\Longrightarrow} \Gamma' \rhd W'$ it holds $\Gamma_1 \leq \Gamma_2$.

Proof. First we note that whenever $\Gamma \rhd W \stackrel{\tau}{\Longrightarrow} \Gamma' \rhd W'$ then

$$\Gamma \rhd W \xrightarrow{\lambda_1} \Gamma_1 \rhd W_1 \xrightarrow{\lambda_2} \cdots \xrightarrow{\lambda_{n-1}} \Gamma_{n-1} \rhd W_{n-1} \xrightarrow{\lambda_n} \Gamma' \rhd W'$$

for some integer $n \geq 0$ and such that for any $i = 1, \cdots, n$ it holds either $\lambda_i = \tau$ or $\lambda_i = c!v$. Recall that whenever $\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'$ then $\Gamma = \Gamma'$ (Proposition 9.4.1(3)), while whenever $\Gamma \rhd W \xrightarrow{c!v} \Gamma' \rhd W'$ then $\Gamma \rhd \leq \Gamma' = \text{upd}_c^v(\Gamma)$.

Now it is possible to prove the statement by induction on the number n of intensional transitions that constitute the weak extensional τ transition. □

Corollary 10.3.3. For any configuration $\Gamma \rhd W$ it holds $\Gamma \rhd W \xrightarrow{\iota(c)}$ if and only if $\Gamma \rhd W \stackrel{\iota(c)}{\Longrightarrow}$.

Proof. If $\Gamma \rhd W \xrightarrow{\iota(c)}$ then it follows trivially that $\Gamma \rhd W \stackrel{\iota(c)}{\Longrightarrow}$.

Conversely, suppose that $\Gamma \rhd W \stackrel{\iota(c)}{\Longrightarrow}$. In this case we have that $\Gamma \rhd W \stackrel{\tau}{\Longrightarrow} \Gamma' \rhd W' \xrightarrow{\iota(c)} \Gamma' \rhd W' \stackrel{\tau}{\Longrightarrow} \Gamma'' \rhd W''$ for some configurations $\Gamma' \rhd W'$ and $\Gamma'' \rhd W''$ such that $\Gamma' \vdash c : \text{free}$. Therefore $\Gamma'(c) = (0, \mathbf{x})$; by Lemma 10.3.2 it follows that $\Gamma \leq \Gamma'$, hence $\Gamma(c) = (0, \mathbf{x})$, or equivalently $\Gamma \vdash c : \text{free}$.

Now it is possible to apply Rule (B-FREE) of the extensional semantics to obtain $\Gamma \rhd W \xrightarrow{\iota(c)} \Gamma \rhd W$. □

Proof of Lemma 10.3.1 Suppose $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$. We prove that, for any channel c, $\Gamma_1 \vdash c : \text{free}$ implies $\Gamma_2 \vdash c : \text{free}$; then the result follows from the fact that $\approx$ is symmetric.

If $\Gamma_1 \vdash c : \text{free}$ for some channel c, then $\Gamma_1 \rhd W_1 \xrightarrow{\iota c} \Gamma_1 \rhd W_1$, and since $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$ it follows that $\Gamma_2 \rhd W_2 \stackrel{\iota(c)}{\Longrightarrow} \Gamma_2' \rhd W_2'$, where $\Gamma_1 \rhd W_1 \approx \Gamma_2' \rhd W_2'$.

By Corollary 10.3.3 it follows that $\Gamma_2 \rhd W_2 \xrightarrow{\iota(c)} \Gamma_2 \rhd W_2$, hence $\Gamma_2 \vdash c : \text{free}$.

We are now ready to prove that weak bisimilarity is sound with respect to barbed congruence. This result can be proved by showing that weak bisimilarity for configurations is preserved by the parallel operator |.

Theorem 10.3.4. [$\approx$ is contextual] Let $\Gamma_1 \rhd W_1$ and $\Gamma_2 \rhd W_2$ be two well-formed configurations such that $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$. Then, $\Gamma_1 \rhd (W_1|W) \approx \Gamma_2 \rhd (W_2|W)$ for all systems W such that $\Gamma_1 \rhd W_1|W$ and $\Gamma_2 \rhd W_2|W$ are well-formed.

Proof. We show that the relation $\mathcal{S}$ defined as follows

$$\{(\Gamma_1 \rhd W_1|W, \Gamma_2 \rhd W_2|W) : \ \Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2 \wedge \forall i{\in}[1..2].\, \Gamma_i \rhd W_i|W \text{ well-form.}\}$$

is a bisimulation. We proceed by case analysis on why $\Gamma_1 \rhd W_1|W \xrightarrow{\alpha} \widehat{\Gamma_1} \rhd \widehat{W_1}$.

1. Let $\Gamma_1 \rhd W_1|W \xrightarrow{\tau} \widehat{\Gamma_1} \rhd \widehat{W_1}$ by an application of rule (B-SHH) because $\Gamma_1 \rhd W_1|W \xrightarrow{c!v} \widehat{W_1}$, for some c and v. This transition can only be derived by an application of rule (B-SYNC), where, for instance,

   - $\Gamma_1 \rhd W_1 \xrightarrow{c!v} W_1'$
   - $\Gamma_1 \rhd W \xrightarrow{c?v} W'$
   - $\widehat{W_1} = W_1'|W'$

   (the symmetric case is similar).

   By an application of rule (B-SHH) it follows that $\Gamma_1 \rhd W_1 \stackrel{\tau}{\longmapsto} \Gamma_1' \rhd W_1'$, with $\Gamma_1' = \mathrm{upd}_c^v(\Gamma_1)$. Since $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$, there is $\Gamma_2' \rhd W_2'$ such that $\Gamma_2 \rhd W_2 \Longmapsto \Gamma_2' \rhd W_2'$ and $\Gamma_1' \rhd W_1' \approx \Gamma_2' \rhd W_2'$. Now, there are two possibilities.

   (a) Let $\Gamma_1 \vdash c : \exp$. By Lemma 9.4.2(2), in the transition $\Gamma_1 \rhd W \xrightarrow{c?v} W'$ it must be $W' = W$. Since $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$ and $\Gamma_1' \rhd W_1' \approx \Gamma_2' \rhd W_2'$, by two applications of Lemma 10.3.1 it follows that:

   - for any channel d, $\Gamma_1 \vdash d : \mathrm{free}$ iff $\Gamma_2 \vdash d : \mathrm{free}$
   - for any channel d, $\Gamma_1' \vdash d : \mathrm{free}$ iff $\Gamma_2' \vdash d : \mathrm{free}$.

   We recall that $\Gamma_1' = \mathrm{upd}_c^v(\Gamma_1)$, and hence Γ_1' and Γ_1 may only differ for the entry at channel c. As a consequence, also Γ_2 and Γ_2' may only differ for the same entry.

   Now, let us analyse the transitions which constitute the weak derivation

   $$\Gamma_2 \rhd W_2 \Longmapsto \Gamma_2' \rhd W_2' \ .$$

   In particular, let

   $$\Gamma_2 \rhd W_2 \Longmapsto \Gamma_2^n \rhd W_2^n \stackrel{\tau}{\longmapsto} \Gamma_2^{n+1} \rhd W_2^{n+1} \Longmapsto \Gamma_2' \rhd W_2' \ .$$

   There are two possibilities.

   i. $\Gamma_2^n \rhd W_2^n \stackrel{\tau}{\longmapsto} \Gamma_2^{n+1} \rhd W_2^{n+1}$ is derived by an application of rule (B-TAUEXT) because $\Gamma_2^n \rhd W_2^n \xrightarrow{\tau} W_2^{n+1}$. This case is easy.

   ii. $\Gamma_2^n \rhd W_2^n \stackrel{\tau}{\longmapsto} \Gamma_2^{n+1} \rhd W_2^{n+1}$ is derived by an application of rule (B-SHH) because $\Gamma_2^n \rhd W_2^n \xrightarrow{d!w} W_2^{n+1}$, for some d and w. Since Γ_2 and Γ_2' may only differ for the entry at channel c, also Γ_2^n and Γ_2^{n+1} may only differ for the same entry. This is because the derivation $\Gamma_2 \rhd W_2 \Longmapsto \Gamma_2' \rhd W_2'$ is untimed, and once a channel becomes exposed it remains so for the whole derivation. By Lemma 10.3.1, $\Gamma_1 \vdash c : \exp$ implies $\Gamma_2 \vdash c : \exp$. By definition of rule (B-SHH), $\Gamma_2^{n+1} \vdash d : \exp$. Since only the entry at c may change during the derivation it follows that $\Gamma_2^n \vdash d : \exp$ (also for $d = c$). By Lemma 9.4.2(2), this implies $\Gamma_2^n \rhd W \xrightarrow{c?v} W$. By an application of rule (B-SYNC) and one application of rule (B-SHH) we can derive

   $$\Gamma_2^n \rhd W_2^n|W \stackrel{\tau}{\longmapsto} \Gamma_2^{n+1} \rhd W_2^{n+1}|W \ .$$

   As a consequence,

   $$\Gamma_2 \rhd W_2|W \Longmapsto \Gamma_2' \rhd W_2'|W$$

with $(\Gamma_1' \rhd W_1'|W, \Gamma_2' \rhd W_2'|W) \in \mathcal{S}$.

(b) Let $\Gamma_1 \vdash c$: free. There are two sub-cases.

i. Let $\neg\mathrm{rcv}(W,c)$. This case is similar to case 1a. In fact, by Lemma 9.4.2(1) the system W is not affected by the transmission at c. More precisely, the transition $\Gamma_1 \rhd W_1|W \xrightarrow{c!v} \widehat{W_1}$ can only be derived by an application of rule (B-SYNC) because

- $\Gamma_1 \rhd W_1 \xrightarrow{c!v} W_1'$
- $\Gamma_1 \rhd W \xrightarrow{c?v} W$
- $\widehat{W_1} = W_1'|W$.

ii. Let $\mathrm{rcv}(W,c)$. By Lemma 9.4.2(3) the transition $\Gamma_1 \rhd W \xrightarrow{c?v} W'$ must have $W' \neq W$. Since $\Gamma_1 \rhd W_1 \approx \Gamma_2 \rhd W_2$, by Lemma 10.3.1 it follows that $\Gamma_2 \vdash c$: free. As $\Gamma_1' = \mathrm{upd}_c^v(\Gamma_1)$, it follows that $\Gamma_1' \vdash c$: exp. Since $\Gamma_1' \rhd W_1' \approx \Gamma_2' \rhd W_2'$, by Lemma 10.3.1 it follows that $\Gamma_2' \vdash c$: exp. As a consequence, the derivation $\Gamma_2 \rhd W_2 \Longmapsto \Gamma_2' \rhd W_2'$ must contain a transition which changes the exposure state of channel c. More precisely, the derivation must have the form

$$\Gamma_2 \rhd W_2 \Longmapsto \Gamma_2^n \rhd W_2^n \xmapsto{\tau} \Gamma_2^{n+1} \rhd W_2^{n+1} \Longmapsto \Gamma_2' \rhd W_2'$$

where the transition $\Gamma_2^n \rhd W_2^n \xmapsto{\tau} \Gamma_2^{n+1} \rhd W_2^{n+1}$ is due to an application of rule (B-SHH) because $\Gamma_2^n \rhd W_2^n \xrightarrow{c!w} W_2^{n+1}$, for some w, with $\Gamma_2^n \vdash c$: free and $\Gamma_2^{n+1} \vdash c$: exp. By Lemma 9.4.2(3) it follows that $\Gamma_2^n \rhd W \xrightarrow{c?w} W'$, for any w. By an application of rule (B-SYNC) and one of rule (B-SHH), it follows that

$$\Gamma_2^n \rhd W_2^n|W \xmapsto{\tau} \Gamma_2^{n+1} \rhd W_2^{n+1}|W' \quad .$$

For any other transition composing the derivation $\Gamma_2 \rhd W_2 \Longmapsto \Gamma_2' \rhd W_2'$ we can apply the same reasoning of case 1a. In particular, for those transitions which are derived by an application of rule (B-SHH) because of a transition labelled $d!w'$, the channel d must be exposed before and (obviously) after the transition. So, by Lemma 9.4.2(2) the systems W and W' can perform the corresponding action $d?w'$ remaining unchanged. More precisely, we have

$$\Gamma_2 \rhd W_2|W \Longmapsto \Gamma_2^n \rhd W_2^n|W \xmapsto{\tau} \Gamma_2^{n+1} \rhd W_2^{n+1}|W' \Longmapsto \Gamma_2' \rhd W_2'|W'$$

with $(\Gamma_1' \rhd W_1'|W', \Gamma_2' \rhd W_2'|W') \in \mathcal{S}$.

□

Theorem 10.3.5. Bisimilarity implies reduction barbed equivalence.

Proof. It suffices to prove that bisimilarity is reduction-closed, barb preserving and contextual. Reduction closure follows from the definition of extensional τ-actions and the Harmony Theorem, Theorem 9.5.11. The preservations of barbs follows directly from Lemma 10.3.1. Contextuality follows from Theorem 10.3.4. □

10.3.2 Completeness

In this Section we prove that, whenever two configurations are barbed equivalent, then there exists a weak bisimulation (in the extensional semantics) between them.

The idea here is to show that $\simeq$ is a bisimulation in the extensional LTS. We address the various requirements individually. First some technical results.

Lemma 10.3.6. Suppose $\Gamma_1 \rhd W_1|\mathrm{eureka}!\langle\mathrm{ok}\rangle \simeq \Gamma_2 \rhd W_2|\mathrm{eureka}!\langle\mathrm{ok}\rangle$ where eureka is fresh to both configurations. Then $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$.

Outline. Let the relation $\mathcal{R}$ over configurations be defined by letting

$$\Gamma_1 \rhd W_1 \,\mathcal{R}\, \Gamma_2 \rhd W_2$$

whenever $\Gamma_1 \rhd W_1 | \mathit{fresh}!\langle \mathrm{ok} \rangle \simeq \Gamma_2 \rhd W_2 | \mathit{fresh}!\langle \mathrm{ok} \rangle$ for some fresh channel *fresh*.

One can check that $\mathcal{R}$ is reduction-closed, contextual and barb-preserving, from which the result follows. □

Proposition 10.3.7. [Preserving extensional τs] Suppose $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$ and $\Gamma_1 \rhd W_1 \overset{\tau}{\longmapsto} \Gamma_1' \rhd W_1'$. Then $\Gamma_2 \rhd W_2 \overset{\tau}{\Longmapsto} \Gamma_2' \rhd W_2'$ such that $\Gamma_1' \rhd W_1' \simeq \Gamma_2' \rhd W_2'$.

Proof. There are two possible reasons why $\Gamma_1 \rhd W_1 \overset{\tau}{\longmapsto} \Gamma_1' \rhd W_1'$:

(i) $\Gamma_1 \rhd W_1 \xrightarrow{\tau} W_1'$ and $\Gamma_1' = \Gamma_1$

(ii) $\Gamma_1 \rhd W_1 \xrightarrow{c!v} W_1'$ and $\Gamma_1' = \mathrm{upd}_c^v(\Gamma_1)$.

We consider the first case; the proof for the second is virtually identical.

Let eureka be a fresh channel; that is it must satisfy $\mathrm{eureka} \notin \mathrm{fn}(W)$ and $\Gamma_1 \vdash \mathrm{eureka} : \mathrm{free}$. Let ok be a message which requires one time unit to be transmitted, i.e. $\delta_{\mathrm{ok}} = 1$. By an application of rules (B-TauPar) and (B-TauExt) we derive

$$\Gamma_1 \rhd W_1 | \mathrm{eureka}!\langle \mathrm{ok} \rangle \overset{\tau}{\longmapsto} \Gamma_1' \rhd W_1' | \mathrm{eureka}!\langle \mathrm{ok} \rangle$$

with $\Gamma_1' \rhd W_1' | \mathrm{eureka}!\langle \mathrm{ok} \rangle \Downarrow_{\mathrm{eureka}}$ and $\Gamma_1' \vdash \mathrm{eureka} : \mathrm{free}$. By Theorem 9.5.11 this transition corresponds in the reduction semantics to

$$\Gamma_1 \rhd W_1 | \mathrm{eureka}!\langle \mathrm{ok} \rangle \twoheadrightarrow \Gamma_1' \rhd W_1' | \mathrm{eureka}!\langle \mathrm{ok} \rangle$$

Because $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$, this step must be matched by a sequence of reductions

$$\Gamma_2 \rhd W_2 | \mathrm{eureka}!\langle \mathrm{ok} \rangle \twoheadrightarrow^* C \tag{10.2}$$

such that $\Gamma_1' \rhd W_1' | E \simeq C$. Depending on whether the transmission at eureka is part of the sequence of reductions or not, the configuration C must be one of the following:

$$\begin{array}{lcll} C_1 & = & \Gamma_2' \rhd W_2' | \mathrm{eureka}!\langle \mathrm{ok} \rangle & \text{with} \quad \Gamma_2' \vdash \mathrm{eureka} : \mathrm{free} \\ C_2 & = & \Gamma_2' \rhd W_2' | \sigma.\mathsf{nil} & \text{with} \quad \Gamma_2' \vdash \mathrm{eureka} : \mathrm{exp} \\ C_3 & = & \Gamma_2' \rhd W_2' | \mathsf{nil} & \text{with} \quad \Gamma_2' \vdash \mathrm{eureka} : \mathrm{free} \end{array}$$

As eureka is a fresh channel and $C_3 \not\Downarrow_{\mathrm{eureka}}$, it follows that C cannot be C_3. Since $\Gamma_1' \rhd W_1' | E \simeq C$ and $\Gamma_1' \vdash \mathrm{eureka} : \mathrm{free}$, by Proposition 10.1.6 it follows that C cannot be C_2. So, the only possibly is $C = C_1$. By Lemma 10.3.6 it follows that $\Gamma_1' \rhd W_1' \simeq \Gamma_2' \rhd W_2'$. It remains to show that $\Gamma_2 \rhd W_2 \overset{\tau}{\Longmapsto} \Gamma_2' \rhd W_2'$.

To this end we can extract out from the reduction sequence (10.2) above a reduction sequence

$$\Gamma_2 \rhd W_2 \twoheadrightarrow^* \Gamma_2' \rhd W_2'$$

We show that each step in this sequence, say $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$, is actually also a τ step in the extensional LTS, $\Gamma \rhd W \overset{\tau}{\longmapsto} \Gamma' \rhd W'$, from which the result follows.

Recall from Theorem 9.5.11 that there are three possibilities for the reduction step $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$. If it is either (Internal), i.e. $\Gamma \rhd W \xrightarrow{\tau} W'$, or a (Transmission), i.e. $\Gamma \rhd W \xrightarrow{c!v} W'$, then by definition $\Gamma \rhd W \overset{\tau}{\longmapsto} \Gamma' \rhd W'$ follows. Condition (ii), (Time), is not possible because in the original sequence (10.2) above the testing component eureka!⟨ok⟩ can not make a σ move. □

Proposition 10.3.8. [Preserving extensional σs] Suppose $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$. Then $\Gamma_1 \rhd W_1 \xrightarrow{\sigma} \Gamma_1' \rhd W_1'$ implies $\Gamma_2 \rhd W_2 \stackrel{\sigma}{\Longmapsto} \Gamma_2' \rhd W_2'$ such that $\Gamma_1' \rhd W_1' \simeq \Gamma_2' \rhd W_2'$.

Proof. We will use a testing context:

$$T = \sigma.(\tau.\text{eureka}!\langle \text{ok}\rangle + \text{fail}!\langle \text{no}\rangle)$$

where eureka and fail are fresh channels. Let $\Gamma_1 \rhd W_1 | T \rightarrow^* \Gamma_1' \rhd W_1' | \text{eureka}!\langle \text{ok}\rangle$ $(= C_1)$. Since $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$ we must have a series of reduction steps

$$\Gamma_2 \rhd W_2 | T \rightarrow^* C_2 \tag{10.3}$$

such that $C_1 \simeq C_2$. Because $C_1 \Downarrow_{\text{eureka}}$ and $C_1 \not\Downarrow_{\text{fail}}$ (recall that fail is fresh) the same must be true of C_2. As $\Gamma_1' \vdash \text{eureka} : \text{free}$, it follows that C_2 must take the form $\Gamma_2' \rhd W_2' | \text{eureka}!\langle \text{ok}\rangle$. By Lemma 10.3.6 we have that $\Gamma_1' \rhd W_1' \simeq \Gamma_2' \rhd W_2'$. It remains to establish that $\Gamma_2 \rhd W_2 \stackrel{\sigma}{\Longmapsto} \Gamma_2' \rhd W_2'$.

We proceed as in the previous proposition, by extracting out of (10.3) the contributions from $\Gamma_2 \rhd W_2$; we know that because of the presence of the time delay in T, and by maximal progress (Theorem 9.3.2), exactly one of the reduction steps involves the passage of time. So we have

$$\Gamma_2 \rhd W_2 \rightarrow^* \Gamma \rhd W \stackrel{\sigma}{\longmapsto} \Gamma' \rhd W' \rightarrow^* \Gamma_2' \rhd W_2'$$

Then these reduction steps can be turned into extensional τ steps, giving the required

$$\Gamma_2 \rhd W_2 \Longmapsto \Gamma \rhd W \stackrel{\sigma}{\longmapsto} \Gamma' \rhd W' \Longmapsto \Gamma_2' \rhd W_2'$$

□

Theorem 10.3.9. [Completeness] In CCCP, $C_1 \simeq C_2$ implies $C_1 \approx C_2$.

Proof. It is sufficient to show that the relation

$$\mathcal{S}\,\{(\Gamma_1 \rhd W_1\,, \Gamma_2 \rhd W_2) :\ \Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2\}$$

is a bisimulation. To do so we show that for each extensional action α, $\simeq$ satisfies the corresponding transfer property given in Definition 10.2.2. Cases $\alpha = \sigma$ and $\alpha = \tau$ follow from the two previous propositions. The proofs for the remaining cases follow in a similar manner, by using an appropriate testing context. Below we provide these testing contexts:

(i) $T_{c?v}[-] - \ \big|\ c\,!\langle v\rangle.\text{eureka}!\langle \text{ok}\rangle | \sigma^{\delta_v - 1}.\text{fail}!\langle \text{no}\rangle$

(ii) $T_{\iota(c)}[-] - \ \big|\ [\exp(c)]\mathsf{nil}, \text{eureka}!\langle \text{ok}\rangle | \text{fail}!\langle \text{no}\rangle$

(iii) $T_{\gamma(c,v)}[-] - \ \big|\ \nu d{:}(0,-).(c[x].[x{=}v]d!\langle \text{ok}\rangle, \mathsf{nil} | \sigma.[\exp(d)]\text{eureka}!\langle \text{ok}\rangle, \mathsf{nil} | \sigma.\text{fail}!\langle \text{no}\rangle)$.

We just provide an outline of the proof for the most interesting case, which is given by $\gamma(c, v)$.

First, suppose that $\Gamma \rhd W \stackrel{\gamma(c,v)}{\Longmapsto} \Gamma' \rhd W'$. Since $\Gamma \rhd W \stackrel{\gamma(c,v)}{\Longmapsto} \Gamma' \rhd W'$ there exist two configurations $\Gamma_1 \rhd W_1$ and $\Gamma_2 \rhd W_2$ such that

$$\Gamma \rhd W \stackrel{\tau}{\Longmapsto} \Gamma_1 \rhd W_1 \xrightarrow{\gamma(c,v)} \Gamma' \rhd W' \stackrel{\tau}{\Longmapsto} \Gamma' \rhd W'$$

Note that $\Gamma_1 \vdash c$ deliver v, hence $\Gamma(c) = (1, v)$. We assume that also $\Gamma(c) = 1, v$, so that $\Gamma \rhd T_{\gamma(c,v)}[\Gamma \rhd W]$ is well defined. Since $\Gamma \leq \Gamma'$ by Lemma 10.3.2 the only other possible alternative is that $\Gamma \vdash c : \text{free}$, in which case the above configuration is not well-defined. However, we will not need to deal with this possibility when

showing that the relation $\mathcal{S}$ is a weak bisimulation. Note also that $\Gamma_1 \rhd W_1 \xrightarrow{\sigma} \Gamma_2 \rhd W_2$, as defined by the Rule (B-DELIVER).

We select a computation (that is, a sequence of reduction steps) $T_{\gamma(c,v)}[\Gamma \rhd W] \rightarrow^* C'$ in a way such that $C' \downarrow_{\text{eureka}}, C' \downarrow_{\text{eureka}}, C' \Downarrow_{\text{fail}}$.

The reader can check that we have the following transition in the extensional semantics.

$$
\begin{array}{lcl}
\Gamma \rhd T_{\gamma(c,v)}[\Gamma \rhd W] & = & \Gamma \rhd W | \nu d{:}(0,-).(c[x].[x{=}v]d!\langle \mathsf{ok}\rangle, \mathsf{nil} | \sigma.[\exp(d)]\text{eureka}!\langle \mathsf{ok}\rangle, \mathsf{nil} | \sigma.\text{fail}!\langle \mathsf{no}\rangle) \\
& \rightarrow_u^* & \Gamma_1 \rhd W_1 | \nu d{:}(0,-).(c[x].[x{=}v]d!\langle \mathsf{ok}\rangle, \mathsf{nil} | \sigma.[\exp(d)]\text{eureka}!\langle \mathsf{ok}\rangle, \mathsf{nil} | \sigma.\text{fail}!\langle \mathsf{no}\rangle) \\
& \rightarrow_\sigma & \Gamma_2 \rhd W_2 | \nu d{:}(0,-).([v{=}v]d!\langle \mathsf{ok}\rangle, \mathsf{nil} | [\exp(d)]\text{eureka}!\langle \mathsf{ok}\rangle, \mathsf{nil} | \text{fail}!\langle \mathsf{no}\rangle) \\
& \rightarrow_u^* & \Gamma' \rhd W' | \nu d : (0, v).([v{=}v]d!\langle \mathsf{ok}\rangle, \mathsf{nil} | [\exp(d)]\text{eureka}!\langle \mathsf{ok}\rangle, \mathsf{nil} | \text{fail}!\langle \mathsf{no}\rangle) \\
& \rightarrow_u & \Gamma' \rhd W' | \nu d : (1, \mathsf{ok}).(\sigma.\mathsf{nil} | [\exp(d)]\text{eureka}!\langle \mathsf{ok}\rangle, \mathsf{nil} | \text{fail}!\langle \mathsf{no}\rangle) \\
& \rightarrow_u^* & \Gamma'' \rhd W'' | \nu d : (1, \mathsf{ok}).(\sigma.\mathsf{nil} | \sigma.\text{eureka}!\langle \mathsf{ok}\rangle | \text{fail}) \\
& \rightarrow_u & \Gamma'' \rhd W'' | \nu d : (1, \mathsf{ok}).(\sigma.\mathsf{nil} | \sigma.\text{eureka}!\langle \mathsf{ok}\rangle | \text{fail}!\langle \mathsf{no}\rangle) \\
& \rightarrow_u & \Gamma''[\text{fail} \mapsto (1, \mathsf{no})] \rhd W'' | \nu d : (1, \mathsf{ok}).(\sigma.\mathsf{nil} | \sigma.\text{eureka}!\langle \mathsf{ok}\rangle | \sigma.\mathsf{nil} \\
& \rightarrow_u & (\Gamma'' \ominus 1) \rhd W' | \nu d : (0, -).(\mathsf{nil} | \text{eureka}!\langle \mathsf{ok}\rangle | \mathsf{nil}) \\
& \rightarrow_u & (\Gamma'' \ominus 1)[\text{eureka} \mapsto (1, \mathsf{ok}) \rhd W' | \nu d : (0, -).(\mathsf{nil} | \sigma.\mathsf{nil} | \mathsf{nil})
\end{array}
$$

Some of the transitions above can be derived for we assuming that $\text{eureka}, \text{fail} \notin \text{fn}(W)$. Note that $\Gamma'' \ominus 1 \vdash \text{fail} : \text{free}$, so that the last configuration does not have a weak barb on fail. Further, $\Gamma'' \ominus 1)[\text{eureka} \mapsto (1, \mathsf{ok}) \vdash \text{eureka} : \text{exp}$, so that the last configuration has a strong barb on channel eureka.

Now suppose that $T_{\gamma(c,v)}[\Gamma \rhd W] \rightarrow^* C$ for some configuration C such that $C \downarrow_{\text{eureka}}$, $C \Downarrow_{\text{fail}}$. In this case the reader can check that it holds $C = \Gamma_1 \rhd W_1 | \nu d : (0, -\sigma).(\mathsf{nil} | \sigma.\mathsf{nil} | \mathsf{nil})$ for some network W_1 and channel environment Γ_1 for which $\Gamma_1 \vdash \text{eureka} : \text{exp}$ and $\Gamma_1 \vdash \text{fail} : \text{free}$.

Informally speaking, in order to reach this configuration the testing component in $T_{\gamma(c,v)}[\Gamma \rhd W]$ should have finished receiving a value in 1 time slot, after which the received value is compared with v; the match had a positive outcome, then the station which performed the check has broadcast a value of length 1 along a restricted channel d. Another parallel component of the test (which can detect transmissions over the restricted channel d) has performed an exposure check on the restricted channel c. Since it has found such a channel to be exposed, then it waited another slot of time before broadcasting along channel eureka.

The important part here resides in the fact that the test, at the end of the first time instant, terminates listening to a transmission along channel c; this can happen only if the tested component $\Gamma \rhd W$ has delivered value v along channel c in the first instant of time. That is, $\Gamma \rhd W \stackrel{\gamma(c,v)}{\Longmapsto} \Gamma' \rhd W'$ for some configuration $\Gamma' \rhd W'$.

We can now show that, whenever $\Gamma \rhd W_1 \simeq \Gamma_2 \rhd W_2$ and $\Gamma \rhd W_1 \stackrel{\gamma(c,v)}{\longmapsto} \Gamma' \rhd W_1'$, then $\Gamma_2 \rhd W_2 \stackrel{\gamma(c,v)}{\Longmapsto} \Gamma_2' \rhd W_2'$ for some $\Gamma_2' \rhd W_2'$ such that $\Gamma_1' \rhd W_1' \simeq \Gamma_2' \rhd W_2'$.

Suppose $\Gamma_1 \rhd W_1 \stackrel{\delta_{c,v}}{\longmapsto} \Gamma_1' \rhd W_1'$; note that since $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$ it follows that $\Gamma_1 \vdash c : \text{exp}$.

Then $T_{\gamma(c,v)}[\Gamma_1 \rhd W_1] \rightarrow^* C_1$ for some C_1 such that $C_1 \downarrow_{\text{eureka}}$ and $C_1 \Downarrow_{\text{fail}}$.

By Definition of Barbed equivalence and the assumption $\Gamma_1 \rhd W_1 \simeq \Gamma_2 \rhd W_2$ it follows that Then $T_{\gamma(c,v)}[\Gamma_2 \rhd W_2] \rightarrow^* C_2$ for some C_2 such that$C_1 \simeq C_2$; in particular $C_2 \downarrow_{\text{eureka}}$ and $C_2 \Downarrow_{\text{fail}}$, from which it follows that $\Gamma_2 \rhd W_2 \stackrel{\gamma(c,v)}{\Longmapsto} \Gamma_2' \rhd W_2'$. The statement that $\Gamma_1' \rhd W_1' \simeq \Gamma_2' \rhd W_2'$ follows from a generalisation lemmas 10.3.6, propositions 10.3.7, 10.3.8 and the fact that, once the action $\gamma(c, v)$ has been performed by a configuration in $T_{\gamma(c,v)}[\Gamma_1 \rhd W_1]$ the testing parallel component evolves in a unique way which is independent from the tested configuration.

□

10.4 Simple Applications of The Proof Method

Below we report some examples of weakly bisimilar configurations. All the values have transmission length 1, including the error value err.

Let Γ be such that $\Gamma \vdash c : \text{free}$. Then

$$\Gamma \rhd \nu d : (0, \mathbf{x}).(\tau.d!\langle \text{noise} \rangle + \tau.d!\langle \text{deliver} \rangle || \lfloor d?(x).[x = \text{noise}](c!\langle \text{noise} \rangle), c!\langle v \rangle \rfloor c!\langle \mathsf{err} \rangle | \sigma.c!\langle v \rangle)$$

$$\approx$$

$$\Gamma \rhd \tau.\sigma.c!\langle v \rangle + \tau.\sigma.c!\langle \mathsf{err} \rangle$$

A more interesting situation consider two networks with different channel environments. In the first one the channel c is corrupted, that is $\Gamma^{\infty} \vdash_n c : \infty$. This constraint is used for modelling a practical situation in which there is too much interference (for example in a range of frequencies) for transmission of values to be performed correctly.

In the second configuration, instead, we assume that channel c is initially exposed; however, two different stations prevent the transmission occurring along such a channel to terminate.

Formally, let Γ, Γ^{∞} such that $\Gamma \vdash c : \text{exp}$ and $\Gamma^{\infty}(c) \vdash_n c : \infty$. Then

$$\Gamma \rhd \mathsf{fix}\, X.c\,!\langle v \rangle.X | \sigma.\mathsf{fix}\, X..c\,!\langle v \rangle.X$$

$$\approx$$

$$\Gamma^{\infty} \rhd \mathsf{nil}$$

10.5 Related Work

We already mentioned that our calculus has been inspired in large part by TCWS, [47]. Here we compare our notion of barbed congruence with that proposed in the same paper.

First, a big improvement in our framework lies in the definition of barb. Using channel environments allows barbs to being easier than those defined for TCWS. Even better, our definition of barb depends solely on the status of a channel environment, and it is completely unrelated to the syntax of a network.

Another improvement is that we managed to prove that, for a large class of networks, bisimulations over the extensional LTS are not only sound, but also complete, with respect to our notion of barbed equivalence. In [47] the bisimulation relation induced by their extensional LTS (which is referred to as *augmented LTS* in the paper) has been proved to be sound, but not complete.

To the best of our knowledge there exist no other papers in the literature which consider behavioural equivalences for networks in the presence of collisions.

Chapter 11

Conclusions

In this Thesis we formalised wireless networks, at different levels of abstraction, and we studied different notions of behavioural equivalences for them.

First we defined a calculus of wireless networks where communication between stations is assumed to be reliable. We defined testing preorders in the style of Hennessy and de Nicola [17, 33] and we exhibited some proof methodologies that have been proved to be sound and complete for them.

We also dedicated a chapter to the analysis of case studies, by modelling formally specifications which are common to wireless and distributed systems in terms of a wireless network; then we exhibited parametric implementations for such models. We exploited our proof techniques to show that the exhibited implementations are behaviourally equivalent to the proposed models. The proposed implementations are parametric in properties that we require a network to satisfy, showing that our proof techniques can be used not only for the analysis of a single wireless network, but also for the analysis of network protocols.

In a second part we extended the calculus to allow wireless stations to exhibit probabilistic behaviour. The testing preorders have been extended in the same style as [20, 19], and sound proof techniques for them have been developed. Surprisingly enough, such techniques fail to be complete. We have redesigned the implementations of the models exhibited in the first part to allow probabilistic behaviour, and we have shown that our proof techniques, despite not being complete, can be used to analyse interesting practical situations.

Finally, we focused on a collision prone timed calculus for broadcast system; here, in contrast with the other calculi proposed in the Thesis, the concept of network topology has been abandoned, and the assumption that the topological structure of a network is flat has been made. The notion of behavioural equivalence we proposed for processes in this calculus is based on Milner and Sangiorgi's barbed congruence [49]. We have provided a characterisation result of such a behavioural equivalence in terms of weak bisimulation. We have also shown that in our calculus transmission of a value itself is not an observable activity; rather, what can be observed is the exposure status of a channel and the end of transmission. We also exhibited a rather simple case study that shows how our bisimulation proof technique can be put into practice.

11.1 Future Research and Development

Our research, however, is far from being complete. While we have successfully managed to define a framework in which compositional reasoning over wireless systems is possible, together with a probabilistic variant, we believe that there are many questions that still have to be answered in order to claim that our calculi can be used as a formal tool for the analysis of wireless networks and protocols.

The first task to be accomplished is that of expanding the number of practical situations in which our calculus can be used. We believe that, in the non-probabilistic setting, more realistic implementations of routing policies, in which the possibility of congestion of nodes and strategies for flow control are present, can be analysed. Handling these situations requires to rethink the implementations of the models that we have

developed in the thesis. For example, introducing the assumption that nodes are subject to congestions in a network would consist of assuming that the process running at a node can be associated with a multiset whose cardinality is bounded by some constant. In order to ensure that a network enforces a connectionless routing policy in this framework would also lead to redefine the code running at each node.

We are also investigating applications in which we assume that stations have a limited power of energy, and it would be nice to establish if our framework, when applied to model power-consuming wireless stations, can be related with other works already existing in the literature, such as [7]. Note that modelling a network whose nodes are subject to energy consumption in our framework is a relatively easy task. Informally speaking, it is sufficient to assume that each value has an integer number associated, corresponding to the amount of energy required by a node to broadcast that channel. Then each node can be equipped with an integer denoting its quantity of residual energy, which has to be decreased every time it performs a broadcast of a value, according to the amount of energy required for broadcasting the latter.

In the probabilistic setting, we believe that our proof techniques can be used to analyse virtually every distributed algorithm which fits the *Las Vegas model*. We are not sure whether our proof techniques can be used to handle protocols which are based on the *Monte Carlo Model*; this is because, as we have seen, our proof techniques cannot be used in the case a network performs a multicast to the external environment with overall probability strictly less than 1. Here the most important improvement would be that of finding a complete characterisation for both the may and must testing preorders has not been answered in this thesis, and in future we will work on revisiting our notions of simulation and deadlock simulations to obtain completeness.

In both the standard and probabilistic frameworks, we consider the task of developing variants of the calculus which enjoy more features particular to wireless systems and process calculi to be interesting. Among the features that we want to add to our calculus the ones we value most are higher order communication and the introduction of mobility. Higher order communication would allow us to analyse situations in which the external environment can broadcast a process to a network, which is then executed at some internal node of the latter. This would allow us to model interesting situations, related in particular to the *Grid Computing*. Also, we believe that introducing mobility in our framework would be very useful when trying to model sensor wireless networks, which are widely gaining more and more interest from the research community.

Further, we remark that the analysis of the case studies that we have shown in this thesis has been rather difficult, due to the large number of parameters that we defined for nodes in the implementation networks. In order to ease the proof of the statements it should be necessary to develop an equational theory for showing whether two networks are testing related each other. Having such an equational theory would reduce the problem of checking whether two networks are equivalent to simple calculations. Further, in the future an automated tool should be developed to help a user to prove the equivalence of two networks. To this end, algorithms for checking deadlock trace equivalence (in the standard framework) and deadlock simulations (in the probabilistic setting) should be investigated.

In the case of our calculus for collisions, we have encountered many problems when trying to exhibit some case studies. This is mainly because of our restriction to a setting where the topology of a network is assumed to be flat. However, we managed to model simple network topologies in our calculus by using broadcasts along different channels and channel restrictions, leading to the definition of networks which implement a routing protocol in a collision prone setting. A case study in which such networks are compared with a network which implements *TDMA* multiple access technique is included in [10]. However, due to our assumption that a network always has a flat topology, we believe that our calculus cannot be used to analyse a wide number of practical applications. Rather, we believe that our framework can be extended to overcome the problem, mostly by introducing a local broadcast feature for it. This would allow the analysis of problems of particular interest in the literature of protocols at the MAC Sublayer, such as the *hidden station* and *exposed station* problems.

It remains to be see how our calculus can be extended to incorporate local broadcast communication, and if our notions of barbed congruence and weak bisimulations can be extended to the novel calculus in an easy manner.

Appendix A

Proofs of the Propositions in Part I

A.1 Structural properties of networks

Proof of Lemma 2.4.1 The best way to prove this lemma is that of building a proof system for checking structural congruence between processes. First we check that two processes are structurally equivalent if and only if a proof can be obtained in our proof system; then we prove that, for any processes P, Q, R such that $P \equiv Q$ and $P \xrightarrow{\alpha} R$, it also holds that $Q \xrightarrow{\alpha} R$. The Proof System we use to check that $P \equiv Q$ for any processes P, Q is depicted in Figure A.1; we write $\Vdash P \equiv Q$ whenever there exists a proof of $P \equiv Q$ in such a proof system.

Now we show that for any processes P, Q, it holds $P \equiv Q$ iff $\Vdash P \equiv Q$. Let $\mathcal{R} = \{(P, Q) \mid \Vdash P \equiv Q\}$. It is immediate to note that $\mathcal{R}$ satisfies the requirements (1i)-(1v) of Definition 2.1.3(1), so that $\mathcal{R}$ is included in $\equiv$. As $\equiv$ is the least equivalence relation between processes that satisfies such constraints, it is also immediate to note that $\equiv$ is included in $\mathcal{R}$. Therefore the $P \equiv Q$ iff $P \mathcal{R} Q$ iff $\Vdash P \equiv Q$.

Let now P, Q be two processes such that $P \equiv Q$. We show by structural induction on the proof that $P \equiv Q$ that $P \xrightarrow{\alpha} R$ for some process R iff $Q \xrightarrow{\alpha} R$. We only show the most interesting cases.

Suppose the last rule applied in the proof of $P \equiv Q$ is (S-P$-$PLUS). Then $Q = P \mid \mathbf{0}$. If $P \xrightarrow{\alpha} R$, we can apply Rule (S-SUML) to infer $Q \xrightarrow{\alpha} R$. Conversely, if $Q \xrightarrow{\alpha} R$, then the last rule applied in the proof of this derivation is (S-SUM$-$L). This is because $Q = P + \mathbf{0}$, and the process $\mathbf{0}$ has no possible transitions. Consequently, $P \alpha R$, as required by the premise of Rule (S-SUM$-$L).

If the last rule applies in the proof of $P \equiv Q$ is (S-P$-$COMM), then there exist P_1, P_2 such that $P = P_1 + P_2$, and $Q = P_2 + P_1$. Suppose $P_1 + P_2 \alpha R$, then the last rule applied in the proof of this transition is either (S-SUM$-$L) or (S-SUM$-$R). In the former case, we have that $P_1 \xrightarrow{\alpha} R$, so that by an application of rule (S-SUM$-$R) we have $P_2 + P_1 \xrightarrow{\alpha} R$. In the latter case, it holds $P_2 \xrightarrow{\alpha} R$, so that it suffices to apply rule (S-SUM$-$L) to infer $P_2 + P_1 \xrightarrow{\alpha} R$.
Conversely, suppose $P_2 \xrightarrow{\alpha} R$. This case is analogous to the one above.

Let now (S-P$-$THEN) be the last rule applied in the proof of $P \equiv Q$. In this case we have that $P =$ if b then Q else P' for some b, P' such that $[\![b]\!] =$ true. If $P \xrightarrow{\alpha} R$, the last rule applied in the proof of this derivation is (S-THEN), so that we have $Q \xrightarrow{\alpha} R$. Conversely, if $Q \xrightarrow{\alpha} R$, we can apply rule (S-THEN) (recall that $[\![b]\!] =$ true) to infer if b then Q else $P' \xrightarrow{\alpha} R$.

The last case we analyse is that in which Rule (S-P$-$UNFOLD) is the last that has been applied in the proof of $P \equiv Q$. Then we have $P = A\langle \tilde{e} \rangle$ for some process definition $A(\tilde{x})$ and list of expressions $\tilde{e}$, while $Q = \{\tilde{v}/\tilde{x}\}P'$ for some process P' and list of values $\tilde{v}$ such that $A(\tilde{x}) \Leftarrow P'$ and $\widetilde{[\![e]\!]} = \tilde{v}$. Suppose that $P \xrightarrow{\alpha} R$. Then the last rule applied in the proof of such a transition is (S-PDEF). Hence we have $\{\tilde{v}/\tilde{x}\}P' \xrightarrow{\alpha} R$. Conversely, suppose $Q \xrightarrow{\alpha} R$. As $Q = \{\tilde{v}/\tilde{x}\}P'$, $\widetilde{[\![e]\!]} = \tilde{v}$ and $A(\tilde{x}) \Leftarrow P'$, we can apply Rule (S-PDEF) to infer $A(\tilde{e}) \xrightarrow{\alpha} R$.

Proof of Lemma 2.4.2 We prove the two implications separately. The only if implication is straightforward; given any process P', Q it is easy to show (by first applying Rule (S-τ), then (S-SUM$-$L)) that $\tau.P' + Q \xrightarrow{\tau} P'$. Thus, we can apply Lemma 2.4.1 to show that, for any P such that $P \equiv \tau.P' + Q$, we have $P \xrightarrow{\tau} P'$.

$$\frac{}{P \equiv Q}\ M = N \qquad \text{(S-P–REFL)}$$

$$\frac{P \equiv Q}{Q \equiv P} \qquad \text{(S-P–SYMM)}$$

$$\frac{}{P \equiv P \mid \mathbf{0}} \qquad \text{(S-P–ZERO)}$$

$$\frac{}{P + Q \equiv Q + P} \qquad \text{(S-P–COMM)}$$

$$\frac{}{P \equiv \text{if } b \text{ then } P \text{ else } Q}\ [\![b]\!] = \text{true} \qquad \text{(S-P–THEN)}$$

$$\frac{}{Q \equiv \text{if } b \text{ then } P \text{ else } Q}\ [\![b]\!] = \text{false} \qquad \text{(S-P–ELSE)}$$

$$\frac{A(\tilde{x}) \Leftarrow P}{A\langle \widetilde{e} \rangle \equiv \{\widetilde{v}/\widetilde{x}\}P}\ \widetilde{[\![e]\!]} = \tilde{v} \qquad \text{(S-P–UNFOLD)}$$

$$\frac{P \equiv R \qquad R \equiv Q}{P \equiv Q} \qquad \text{(S-P–TRANS)}$$

Figure A.1: Proof system to check structural congruence between processes

For the if implication the proof is carried out by Rule Induction on the last rule applied in the proof of the transition $P \xrightarrow{\tau} Q$. We perform a case analysis on the last rule applied in the derivation, then we exhibit a term Q such that $P \equiv \tau.P' + Q$.

If the last rule applied is (S-τ), then $P = \tau.P'$. If we let $Q = \mathbf{0}$, it follows that $P \equiv \tau.P' + Q$ by Definition 2.1.3(1).

Now suppose the last rule applied is (S-SUM – L). Thus $P = P_1 + Q_1$ for some P_1, Q_1 such that $P_1 \xrightarrow{\tau} P'$. By inductive hypothesis, there exists Q' such that $P_1 \equiv \tau.P_1 + Q_1$; thus

$$P \equiv P_1 + Q_1 \equiv (\tau.P' + Q') + Q_1 \equiv \tau.P' + (Q' + Q_1)$$

so that $Q = Q' + Q'_1$.
If the last Rule applied is (S-SUM – R), then $P = Q_1 + P_1$ for some P_1, Q_1 such that $P_1 \xrightarrow{\tau} P'$. By inductive hypothesis, there exists Q' such that $P_1 \equiv \tau.P_1 + Q_1$; therefore we have

$$P \equiv P_1 + Q_1 \equiv Q_1 + (\tau.P' + Q') \equiv (Q_1 + \tau.P') + Q' \equiv (\tau.P' + Q_1) + Q' \equiv \tau.P' + (Q_1 + Q')$$

Then $Q = Q_1 + Q'$

If the last rule applied is (S-THEN), we have that $P = \text{if } b \text{ then } P \text{ else }_1 Q_1$ for some P_1, Q_1 and b such that $[\![b]\!] = \text{true}$, $P_1 \xrightarrow{\tau} P'$. By inductive hypothesis it holds $P' \equiv \tau.P' + Q$, and by an application of Definition 2.1.3 (1) we have $P \equiv P_1 \equiv \tau.P' + Q$. The case for rule (S-ELSE) is similar.

The last case to be checked is given by rule (S-PDEF) being the last rule applied in the derivation $P \xrightarrow{\tau} P'$. In this case we have that $P = A\langle \tilde{v} \rangle$ for some process definition $A(\tilde{x})$ and list of values $\tilde{v}$; also, $A(x) \Leftarrow P_1$ for some process P_1 such that $\{\tilde{v}/\tilde{x}\}P_1 \xrightarrow{\tau} \tau.P' + Q$, where Q is an arbitrary process. As $P = A\langle \tilde{v} \rangle$ and $A(x) \Leftarrow P_1$, we can apply Definition 2.1.3(1) to obtain $P \equiv \{\tilde{v}/\tilde{x}\}P_1 \equiv \tau.P' + Q$.

Proof of Proposition 2.4.5 We prove the three statements separately. Each of them is proved by Rule Induction on the proof of the transition $\Gamma \rhd M_1 \mid M_2 \xrightarrow{\lambda} N$.

(i) Suppose $\Gamma \rhd M_1 \mid \xrightarrow{m.c?v} N$. Note that, if either Rule (B-REC), (B-DEAF) or (B-DISC) have been applied as the last rule to infer $\Gamma \rhd M_1 \mid M_2 \xrightarrow{m.c?v} N$, then $M_1 \mid M_2 = n[\![P]\!]$ for some process P and node n. However, this is not possible, so that these cases are vacuous. Let then Rule (B-PROP) be the last rule which has been applied in the proof of $\Gamma \rhd M_1 \mid M_2 \xrightarrow{m.c?v} N$; in this case there exist M', M'', N', N'' such that $\Gamma \rhd M' \xrightarrow{m.c?v} N'$, $\Gamma \rhd M'' \xrightarrow{m.c?v} N''$ and $M_1 \mid M_2 = M' \mid M''$, $N = N' \mid N''$. Now we have three possible cases:

- $M' = M_1$. In this case we have that $M'' = M_2$. Let $N_1 = N'$, $N_2 = N''$. Then we have $\Gamma \rhd M_i \xrightarrow{m.c?v} N_i$, $i = 1,2$ and $N = N_1 \,|\, N_2$.
- $M' = M_1 \,|\, M_3$ for some M_3. By inductive hypothesis we have that $\Gamma \rhd M_1 \,|\, N_1$ and $\Gamma \rhd M_3 \xrightarrow{m.c?v} N_3$ for some N_1, N_3 such that $N' = N_1 \,|\, N_3$. Also, as $M_1 \,|\, M_2 = M' \,|\, M''$, we have that $M_1 \,|\, M_2 = M_1 \,|\, M_3 \,|\, M''$, therefore $M_2 = M_3 \,|\, M''$. Since $\Gamma \rhd M_3 \xrightarrow{m.c?v} N_3$ and $\Gamma \rhd M'' \xrightarrow{m.c?v} N''$, by applying Rule (B-PROP) we obtain $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_3 \,|\, N''$. Let then $N_2 = N_3 \,|\, N''$. It remains to show that $N = N_1 \,|\, N_2$. However, we already know that $N = N' \,|\, N''$, and $N' = N_1 \,|\, N_3$. Thus $N = N_1 \,|\, N_3 \,|\, N''$, which is exactly $N = N_1 \,|\, N_2$
- $M_1 = M' \,|\, M_3$ for some M_3. In this case we have that $M'' = M_3 \,|\, M_2$, and by inductive hypothesis there exist N_3 and N_2 such that $\Gamma \rhd M_3 \xrightarrow{m.c?v} N_3$, $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$ and $N'' = N_3 \,|\, N_2$. Now, by applying rule (B-PROP) to the transitions $\Gamma \rhd M' \xrightarrow{m.c?v} N'$ and $\Gamma \rhd M_3 \xrightarrow{m.c?v} N_3$, we obtain $\Gamma \rhd M_1 \xrightarrow{m.c?v} N' \,|\, N_3$ (recall that $M_1 = M' \,|\, M_3$). If we let $N_2 = N' \,|\, N_3$, we obtain $N = N' \,|\, N'' = N' \,|\, N_3 \,|\, N_2 = N_1 \,|\, N_2$.

(ii) Suppose $\Gamma \rhd M_1 \,|\, M_2 \xrightarrow{m.c!v} N$; the case in which the last rule applied in the proof of the transition is Rule (B-BROAD) is vacuous. Otherwise, either Rule (B-SYNC − L) or (B-SYNC − R) has been applied as the last rule in the proof of $\Gamma \rhd M_1 \,|\, M_2 \xrightarrow{m.c!v} N$. We only consider the case of (B-SYNC − L), as the other one is symmetric. In this case we have that $M_1 \,|\, M_2 = M' \,|\, M''$ for some M', M'', N', N'' such that $\Gamma \rhd M' \xrightarrow{m.c!v} N'$, $\Gamma \rhd M'' \xrightarrow{m.c?v} N''$ and $N = N' \,|\, N''$. We have now three possible cases.

- $M_1 = M'$; it follows $M_2 = M''$. Let $N_1 = N'$, $N_2 = N''$; then we have $\Gamma \rhd M_1 \xrightarrow{m.c!v} M'$, $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$ and $N = N_1 \,|\, N_2$, as we wanted to prove
- $M_1 = M' \,|\, M_3$ for some M_3; it follows $M'' = M_3 \,|\, M_2$. By Lemma hlnets.lem:par.comp(i), proved above, we have that there exist N_3, N_2 such that $\Gamma \rhd M_3 \xrightarrow{m.c?v} N_3$, $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$ and $N'' = N_3 \,|\, N_2$. Now let $N_1 = N' \,|\, N_3$. We can apply Rule (B-SYNC − L) to the transitions $\Gamma \rhd M' \xrightarrow{m.c!v} N'$ and $\Gamma \rhd M_3 \xrightarrow{m.c?v} N_3$ to infer $\Gamma \rhd M_1 \,|\, N_1$ (recall $M_1 = M' \,|\, M_3, N_1 = N' \,|\, N_3$). It remains to show that $N = N_1 \,|\, N_2$. This is trivial, as $N = N' \,|\, N'' = N' \,|\, N_3 \,|\, N_2 = N_1 \,|\, N_2$.
- $M' = M_1 \,|\, M_3$ for some M_3; it follows that $M_2 = M_3 \,|\, M''$. By inductive hypothesis, we have that there exist N_1, N_3 such that $N' = N_1 \,|\, N_3$ and either (a) $\Gamma \rhd M_1 \xrightarrow{m.c!v} N_1$ and $\Gamma \rhd M_3 \xrightarrow{m.c?v} N_3$ or (b) $\Gamma \rhd M_1 \xrightarrow{m.c?v} N_1$ and $\Gamma \rhd M_3 \xrightarrow{m.c!v} N_3$. We only consider case (a), as (b) can be proved in a similar way. In this case, let $N_2 = N_3 \,|\, N''$. By an application of Rule (B-PROP) we obtain $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$. Thus we have $\Gamma \rhd M_1 \xrightarrow{m.c!v} N_1$, $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$; it remains to check that $N = N_1 \,|\, N_2$. This is straightforward, as $N = N' \,|\, N'' = N_1 \,|\, N_3 \,|\, N'' = N_1 \,|\, N_2$.

(iii) Suppose $\Gamma \rhd M_1 \,|\, M_2 \xrightarrow{m.\tau} N$. This case is similar to the one in which $\Gamma \rhd M_1 \,|\, M_2 \xrightarrow{m.c!v} N$; details have been omitted.

Proof of Proposition 2.4.6 The proof is similar to that of Lemma 2.4.1.

We define a proof system to check whether two system terms are structurally equivalent, and we prove that it is sound and complete with respect to the relation $\equiv$ for system terms. Then we show that, for any network, $\Gamma \rhd M, \Gamma \rhd M'$, and $\Gamma \rhd N'$ such that $M \equiv N$, we have that $\Gamma \rhd M \xrightarrow{\lambda} M'$ iff $\Gamma \rhd N \xrightarrow{\lambda} N'$ for some N' such that $M' \equiv N'$.

The proof system we use to prove that $M \equiv N$ is depicted in Figure A.2. Again, we write $\Vdash M \equiv N$ if the statement $M \equiv N$ can be proved in such a proof system. It is straightforward to show that the relation $\mathcal{R} = \{(M,N) \mid \Vdash M \equiv N\}$ and the structural congruence $\equiv$ are the same relation.

Now we prove that $\Gamma \rhd M \xrightarrow{\lambda} M'$ and $M \equiv N$ for some M, M', N, if and only if $\Gamma \rhd N \xrightarrow{\lambda} N'$ for some N' such that $M' \equiv N'$. We proceed by structural induction on the derivation $\Vdash M \equiv N$; we check only the most interesting cases.

Suppose the last rule that has been applied in the derivation $\Vdash M \equiv N$ is (B-T − PROC). Then $M = m[\![P]\!]$, $Nn[\![Q]\!]$ for some node m and processes P, Q such that $P \equiv Q$. Suppose $\Gamma \rhd M \xrightarrow{\lambda} M'$. We perform a case analysis on the last rule applied on the proof of this transition.

$$\frac{}{M \equiv N}\ M = N \qquad \frac{M \equiv N}{N \equiv M}$$

$$\frac{P \equiv Q}{m[\![P]\!] \equiv m[\![Q]\!]} \qquad \frac{}{M \equiv M \mid \mathbf{0}}$$

$$\frac{}{M \mid N \equiv N \mid M} \qquad \frac{M \equiv N}{M \mid L \equiv N \mid L}$$

$$\frac{M \equiv L \qquad L \equiv N}{M \equiv N}$$

Figure A.2: Proof system to check structural congruence between system terms

If Rule (B-BROAD) has been applied, then $\lambda = m.c!v$ for some channel c and value v. Further, we have that $P \xrightarrow{c!v} R$ for some process R and $M' = m[\![R]\!]$. By Lemma 2.4.1 we also have that $Q \xrightarrow{c!v} R$; by an application of Rule (B-BROAD) we can infer $\Gamma \rhd N \xrightarrow{m.c!v} m[\![R]\!]$.
If Rule (B-REC) has been applied, then $\lambda = n.c?v$ for some channel c, value v and node n such that $\Gamma \vdash m \leftarrow n$. Further, $P \xrightarrow{c?v} R$ for some process R, and $M' = m[\![R]\!]$. By Lemma 2.4.1 we have that $Q \xrightarrow{c?v} R$; now we can apply Rule (B-REC) to the last transition to infer $\Gamma \rhd N \xrightarrow{n.c?v} m[\![R]\!]$.
Let us look at the case in which the last rule applied in the proof of $\Gamma \rhd M \xrightarrow{\lambda} M'$ is Rule (B-DEAF). In this case $\lambda = n.c?v$ for some node n, channel c and value v. Further, the predicate $\mathrm{rcv}(M, c)$ is false, and $M' = M$. Also, as $M \mathcal{R} N$, we have $M \equiv N$, so that the predicate $\mathrm{rcv}(N, c)$ Thus, we can apply Rule (B-DEAD) to obtain $\Gamma \rhd N \xrightarrow{n.c?v} N$. Now it remains to check that $N \equiv M$, which follows by hypothesis.
The last case we need to check is that in which the last rule applied in the proof of the transition $\Gamma \rhd M \xrightarrow{\lambda} M'$ is Rule (B-DISC). In this case, $\lambda = n.c?v$ for some node n, channel c and value v such that $\Gamma \vdash m \nleftarrow n$. Further, $M' = M$. Now we can apply Rule (B-DIST) to $\Gamma \rhd N$ to infer $\Gamma \rhd N \xrightarrow{n.c?v} N$. Again, we have that $M \equiv N$ by hypothesis.

Suppose now that the last rule applied in the proof system of Figure A.2 to prove $M \equiv N$ is Rule (S-T − ZERO). Then $N = M \mid \mathbf{0}$. First suppose $\Gamma \rhd M \xrightarrow{\lambda} M'$; we prove that $\Gamma \rhd M \mid \mathbf{0} \xrightarrow{\lambda} M' \mid \mathbf{0}$ by performing a case analysis on λ. After we have proved this, it remains to show that $M' \equiv M' \mid \mathbf{0}$, which is straightforward by Definition 2.1.3 (2). Consider then the case where $\lambda = m.\tau$ for some m. An immediate application of Rule (B-τ − PROP − L) to the transition $\Gamma \rhd M \xrightarrow{m.\tau} M'$allows us to infer $\Gamma \rhd M \mid \mathbf{0} \xrightarrow{m.\tau} M' \mid \mathbf{0}$. If $\lambda = m.c?v$, recall that $\Gamma \rhd \mathbf{0} \xrightarrow{m.c?v} \mathbf{0}$ for any node m, channel c and value v. Thus, we can apply Rule (B-PROP) to the last transition and to $\Gamma \rhd M \xrightarrow{m.c?v} M'$ to infer $\Gamma \rhd M \mid \mathbf{0} \xrightarrow{m.c?v} \Gamma \rhd M' \mid \mathbf{0}$. Finally, consider the case in which $\lambda = m.c!v$. Here we can apply Rule (B-SYNC − L) to the transitions $\Gamma \rhd M \xrightarrow{m.c!v} M'$ and $\Gamma \rhd \mathbf{0} \xrightarrow{m.c?v} \mathbf{0}$ to obtain $\Gamma \rhd M \mid \mathbf{0} \xrightarrow{m.c!v} \Gamma \rhd M' \mid \mathbf{0}$.
Now, suppose $\Gamma \rhd M \mid \mathbf{0} \xrightarrow{\lambda} N'$. We perform a case analysis on λ to show that $N' = M' \mid \mathbf{0}$ for some M' such that $\Gamma \rhd M \xrightarrow{\lambda} M'$. Then it remains to notice that $M' \equiv M' \mid M' \mid \mathbf{0}$, which is straightforward from Definition 2.1.3(2). Consider first the case $\lambda = \tau$. By Lemma 2.4.5 (iii) we have that either (a) $\Gamma \rhd M \xrightarrow{m.\tau} M'$ for some M' such that $N' = M' \mid \mathbf{0}$, or (b) $\Gamma \rhd \mathbf{0} \xrightarrow{m.\tau} M''$ for some M'' such that $N' = M' \mid M''$. Case (a) is trivial, while it is impossible for (b) to hold; in fact, we have that $\Gamma \rhd \mathbf{0} \xrightarrow{m.\tau}$ for any node m. If $\lambda = m.c!v$, we can apply Lemma 2.4.5 (ii) to find that $\Gamma \rhd M \xrightarrow{m.c!v} M'$ for some M' such that $N' = M' \mid \mathbf{0}$. Note that here, as for in the case $\lambda = \tau$, the application of Lemma 2.4.5(ii) leads to two alternatives; however, one of the two cases requires that $\Gamma \rhd \mathbf{0} \xrightarrow{m.c!v} N''$ for some N'', which is not possible. The last case to check is the one in which $\lambda = m.c?v$. In this case we can apply Lemma 2.4.5(i) to infer $\Gamma \rhd M \xrightarrow{m.c?v} M'$ for some M' such that $N' = M' \mid \mathbf{0}$.

The last case we check is that in which Rule (S-T − COMP) is the last rule applied in the derivation $\Vdash M \equiv N$.

In this case we have that $M = M_1 \,|\, L$, $N = N_1 \,|\, L$ for some M_1, N_1, L such that $M_1 \equiv N_1$. Suppose $\Gamma \rhd M_1 \,|\, L \xrightarrow{\lambda} M'$. We show that $\Gamma \rhd N_1 \,|\, L \xrightarrow{\lambda} N'$, for some N' such that $M' \equiv N'$, by performing a case analysis on λ.
If $\lambda = m.\tau$ for some m we obtain, by an application of Lemma 2.4.5(iii) that either

1. $\Gamma \rhd M_1 \xrightarrow{m.\tau} M_1'$ for some M_1 such that $M' = M_1' \,|\, L$, or
2. $\Gamma \rhd L \xrightarrow{m.\tau} L'$ for some L' such that $M' = M_1 \,|\, L$

Let us consider the first case. By inductive hypothesis we have that $\Gamma \rhd N_1 \xrightarrow{m.\tau} \Gamma \rhd N_1'$ for some N_1' such that $M_1' \equiv N_1'$. By an application of rule (B-τ − PROP − L) we can infer $\Gamma \rhd N_1 \,|\, L \equiv \Gamma \rhd N_1' \,|\, L$. As $M_1' \equiv N_1'$, we can apply Rule (S-T − COMP) to the derivation $\Vdash M_1' \equiv N_1'$ to obtain $M_1' \,|\, L \equiv N_1' \,|\, L$.
We now turn our attention to the second case: here we have that $\Gamma \rhd L \xrightarrow{m.\tau} L'$. An application of Rule (B-τ − PROP − R) allows us to infer $\Gamma \rhd N_1 \,|\, L \xrightarrow{m.\tau} \Gamma \rhd N_1 \,|\, L'$. Now it suffices to apply Rule (S-T − COMP) to the derivation $\Vdash M_1 \equiv N_1$ to obtain $M_1 \,|\, L' \equiv N_1 \,|\, L'$.
Now we consider the case in which $\lambda = m.c!v$. Again, since $\Gamma \rhd M_1 \,|\, L \xrightarrow{m.c!v} M'$, we have two possible cases as a consequence of Lemma 2.4.5 (ii)

1. $\Gamma \rhd M_1 \xrightarrow{m.c!v} M_1'$, $\Gamma \rhd L \xrightarrow{m.c?v} L'$, and $M' = M_1' \,|\, L'$, or
2. $\Gamma \rhd M_1 \xrightarrow{m.c?v} M_1'$, $\Gamma \rhd L \xrightarrow{m.c!v} L'$, and $M' = M_1' \,|\, L'$

These two cases have to be analysed separately; in a way similar to that for the case $\lambda = m.\tau$, it can be proved that $\Gamma \rhd N_1 \,|\, L \xrightarrow{m.c!v} N_1' \,|\, L'$, from which it is trivial to show $M_1' \,|\, L' \equiv N_1' \,|\, L'$ (recall that $M_1' \equiv N_1'$). The last case we need to consider is the one in which $\Gamma \rhd M_1 \,|\, L \xrightarrow{m.c?v} M'$; this case is analogous to the previous ones, this time using Lemma 2.4.5(i).

Proof of Proposition 2.4.7 The only if implication is easy. For the if implication, we perform a Rule Induction on the proof of the transition $\Gamma \rhd M \xrightarrow{m.c!v} M'$.

If the last rule applied to infer $\Gamma \rhd M \xrightarrow{m.c!v} M'$ is (B-BROAD), then $M = m[\![P']\!]$ for some P' such that $P' \xrightarrow{c!v} P''$. By Lemma 2.4.3 there exist an expression e and a process Q' such that $[\![e]\!] = v$ and $P' \equiv c!\langle e\rangle.P'' + Q'$; thus, if we let $P = P''$, $Q = Q'$, we obtain $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, \mathbf{0}$. This is done by first applying constraint (i), then constraint (ii) of Definition 2.1.3 to $\Gamma \rhd m[\![P']\!]$. (2). It remains to show that $\Gamma \rhd m[\![c!\langle e\rangle.P + Q]\!] \,|\, \mathbf{0} \xrightarrow{m.c!v} \Gamma \rhd m[\![P]\!] \,|\, N'$ for some N'. To this end, let $N' = \mathbf{0}$. By lemma 2.4.3 we have that $c!\langle e\rangle.P + Q \xrightarrow{c!v} P$ (recall that $[\![e]\!] = v$); thus, by an application of Rule (B-BROAD) we have $m[\![c!\langle e\rangle.P + Q]\!] \xrightarrow{m.c!v} m[\![P]\!]$; further, $\Gamma \mathbf{0} \xrightarrow{m.c?v} \mathbf{0}$ by Rule (B-$\mathbf{0}$). Finally, we can apply (B-SYNC − L) to infer $\Gamma \rhd m[\![c!\langle e\rangle.P + Q]\!] \,|\, \mathbf{0} \xrightarrow{m.c!v} m[\![P]\!] \,|\, \mathbf{0}$.

Suppose now the last Rule applied in the proof of the transition $\Gamma \rhd M \xrightarrow{m.c!v} N$ is (B-SYNC − L). Thus we have $M = M_1 \,|\, M_2$ for some M_1, M_2 such that $\Gamma \rhd M_1 \xrightarrow{m.c!v} N_1$ and $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$ for some N_1, N_2 such that $N = N_1 \,|\, N_2$. By inductive hypothesis, we have that $M_1 \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, M'$ for some expression e, processes P, Q and system terms M', N' such that $[\![e]\!] = v$, $\Gamma \rhd M' \xrightarrow{m.c?v} N'$ and $N_1 \equiv \Gamma \rhd m[\![P]\!] \,|\, N'$. It is easy now to check that $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, (M' \,|\, M_2)$. Also, we have $N \equiv m[\![P]\!] \,|\, (N' \,|\, N_2)$. It remains to show that $\Gamma \rhd M' \,|\, M_2 \xrightarrow{m.c?v} N' \,|\, N_2$. This can be done by applying Rule (B-PROP) to the transitions $\Gamma \rhd M' \xrightarrow{m.c?v} N'$ and $\Gamma \rhd M_2 \xrightarrow{m.c?v} N_2$.

The last case to consider is that in which the last rule applied in the proof of the transition $\Gamma \rhd M \xrightarrow{m.c!v} N$ is (B-SYNC − R). This case is analogous to the previous one.

Proof of Proposition 2.4.9 The only if implication is easy. For the if implication, we proceed by Rule Induction on the proof of the transition $\Gamma \rhd M \xrightarrow{m.c?v} N$. We show that, whenever $\Gamma \rhd M \xrightarrow{m.c?v} N$, requirements (i)-(vi) stated in Proposition 2.4.9 are satisfied.

Suppose the last rule applied in the proof of the transition above is Rule (B-REC). Then we have that $M = n[\![P]\!]$, $N = n[\![P']\!]$ for some n, P, P' such that $\Gamma \vdash m \to n$ and $P \xrightarrow{c?v} P'$. By definition of well formed network (Definition 2.1.1) we have that $m \neq n$, so that Requirement (ii) is met. By Lemma 2.4.4 we also have that

$P = c?v.P'' + Q$ for some process Q such that $P' = \{v/x\}P''$. Further, let I be the index set $\{1\}$ and let $n_1 = n$, $P_1 = P''$, $Q_1 = Q$. Let also $M_1 = \mathbf{0}$, $M_2 = \mathbf{0}$. By Definition 2.1.3(2) we have that $M \equiv n_1[\![c?v.P_1 + Q_1]\!] \mid M_1 \mid M_2$, so that Requirement (i) is met. Further, as $\Gamma \vdash m \to n_1$ and I is the singleton set $\{1\}$, requirement (iii) is also met. As $M_1 = \mathbf{0}$, $N_1 = \mathbf{0}$, it is trivial to show that requirements (iv) and (v) are also satisfied. It remains to show that Requirement (vi) is satisfied, that is $N \equiv n_1[\![P_1]\!] \mid M_1 \mid N_1$. As $P_1 = P'$, $n_1 = n$, we already know that $N = n_1[\![P_1]\!]$. Now a trivial application of Definition 2.1.3(2), together with the definitions $M_1 = \mathbf{0}$, $N_1 = \mathbf{0}$, leads to $N \equiv n_1[\![\{v/x\}P_1]\!] \mid M_1 \mid N_1$.

If the last rule applied to infer $\Gamma \rhd M \xrightarrow{m.c?v} N$ is Rule (B-DEAF), then we have $M = N = n[\![P]\!]$ for some process P and node n such that $\neg\mathrm{rcv}(P, c)$. Also, we have $\Gamma \vdash m \to n$, so that by definition of well formed network it follows $m \neq n$. This shows that Requirement (ii) is met. Let now $I = \emptyset$, $M_1 = M$ and $M_2 = \mathbf{0}$. We already argued that the predicate $\mathrm{rcv}(M_1, c)$ is false, so that Requirement (iv) is satisfied. Also, as $M_2 = \mathbf{0}$, it is straightforward to check that Requirement (v) is met also. As the set $\{n_i\}_{i\in I}$ is empty, it is trivial to show that, for any $i \in \emptyset$, $\Gamma \vdash m \to n_i$. Now, recall that we have $\prod_{i\in\emptyset} N_i = \mathbf{0}$, where $\{N_i\}_{i\in\emptyset}$ is an empty collection of system terms. it is straightforward to prove $M \equiv \prod_{i\in I} n_i[\![c?\langle x\rangle.P_i + Q_i]\!] \mid M_1 \mid M_2$, and $N \equiv \prod_{i\in I} n_i[\![\{v/x\}P_i]\!] \mid M_1 \mid M_2$; these two equivalences prove that requirements (i) and (vi) are also satisfied.

Suppose that the last rule applied in the proof of the transition $\Gamma \rhd M \xrightarrow{m.c?v} N$ is (B-DISC). Again, we have that $M = N = n[\![P]\!]$ for some node n. Further, the side conditions of Rule (B-DISC) ensure that $m \neq n$, so that Requirement (ii) is met, and that $\Gamma \vdash m \twoheadrightarrow n$. Let $I = \emptyset$, $M_1 = \mathbf{0}$ and $M_2 = M$. Note that $\mathrm{nodes}(M_2) = \{n\}$, and since $\Gamma \vdash m \twoheadrightarrow n$ requirement (v) is met. In a way similar as the case above, we can show that $M \equiv \prod_{i\in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2$ (as for in the case of Rule (B-DEAF), the collections $\{P_i\}_{i\in I}, \{Q_i\}_{i\in I}$ and $\{n_i\}_{i\in I}$ are empty), while $N \equiv \prod_{i\in I} n_i[\![\{v/x\}P_i]\!] \mid M_1 \mid M_2$, and requirements (i) and (vi) are satisfied. Finally, it is trivial to show that requirements (iii), (iv) and (v) are met.

The last case we need to check is that in which the last rule applied in the proof of $\Gamma \rhd M \xrightarrow{m.c?v} N$ is Rule (B-PROP). In this case we have that $M = M' \mid M'$ and $N = N' \mid N''$ for some M', M'', N' and N'' such that

$$\Gamma \rhd M' \xrightarrow{m.c?v} N'$$
$$\Gamma \rhd M'' \xrightarrow{m.c?v} N''$$

By inductive hypothesis, we have that there exist two finite index sets J, K, collections of processes $\{P_j\}_{j\in J}$, $\{Q_j\}_{j\in J}$, $\{P_k\}_{k\in K}$, $\{Q_k\}_{k\in K}$ and system terms M'_1, M''_1, M'_2, M''_2 such that

1. $M_1 \equiv \prod_{j\in J} n_j[\![c?(x).P_j + Q_j]\!] \mid M'_1 \mid M'_2$ and
   $M_2 \equiv \prod_{k\in K} n_k[\![c?(x).P_k + Q_k]\!] \mid M''_1 \mid M''_2$
2. $m \notin \mathrm{nodes}(M'), m \notin \mathrm{nodes}(M')$
3. for each $j \in J$ $\Gamma \vdash m \to n_j$, and for every $k \in K$, $\Gamma \vdash m \to n_k$
4. $\neg\mathrm{rcv}(M'_1, c)$, $\neg\mathrm{rcv}(M'_2, c)$
5. for any $n \in \mathrm{nodes}(M''_1), \Gamma \vdash m \twoheadrightarrow n$, and for any $n \in \mathrm{nodes}(M''_2), \Gamma \vdash m \twoheadrightarrow n$
6. $N_1 \equiv \prod_{j\in J} n_j[\![\{v/x\}P_j]\!] \mid M'_1 \mid M'_2$, while
   $N_2 \equiv \prod_{k\in K} n_k[\![\{v/x\}P_k]\!] \mid M'_2 \mid M''_2$

Without loss of generality, we can assume that the sets J and K are disjoint. Let $I = J \cup K$, $M_1 = M'_1 \mid M''_1$, $M_2 = M'_2 \mid M''_2$ Now, we show that requirements (i)-(vi) of the statement are met by M and N.

(i) $M \equiv \prod_{i\in I} n_i[\![c?(x).P_i + Q_i]\!] \mid M_1 \mid M_2$: This follows from (1) above and Definition 2.1.3(2)

(ii) $m \notin \mathrm{nodes}(M)$. Recall that $M = M' \mid M''$; as $m \notin \mathrm{nodes}(M')$ and $m \notin \mathrm{nodes}(M'')$, we also have $m \notin \mathrm{nodes}(M') \cup \mathrm{nodes}(M'') = \mathrm{nodes}(M' \mid M'') = \mathrm{nodes}(M)$

(iii) for any $i \in I, \Gamma \vdash m \to n_i$: let $i \in I$; then either $i \in J$ or $i \in K$. If $i \in J$, then $\Gamma \vdash m \to n_i$ is a direct consequence of (3) above. Similarly, if $i \in K$, again by (3) above $\Gamma \vdash m \to n_i$

(iv) $\neq \mathrm{rcv}(M_1, c)$: suppose this statement is not true. Then, we have $M_1 \equiv n[\![c?(x).P + Q]\!] \,|\, L$ for some processes P, Q, node n and system term L. Since $M_1 = M_1' \,|\, M_1''$, either $M_1' \equiv n[\![c?(x).P + Q]\!] \,|\, L'$ for some system term L' or $M_1'' \equiv n[\![c?(x).P + Q]\!] \,|\, L''$ for some system term L''. Suppose that the first case holds. Then we have $\mathrm{rcv}(M_1', c) = \mathrm{true}$, which contradicts (4) above. In an analogous way, if the second case holds we have $\mathrm{rcv}(M_1'', c) = \mathrm{true}$, which again contradicts (4) above. Therefore we have $\neg\mathrm{rcv}(M_1, c)$

(v) for any $n \in \mathrm{nodes}(M_2), \Gamma \vdash m \twoheadrightarrow n$: recall that $M_2 = M_2' \,|\, M_2''$, from which it follows $\mathrm{nodes}(M_2) = \mathrm{nodes}(M_2') \cup \mathrm{nodes}(M_2'')$. By (5) above, we have that whenever $n \in \mathrm{nodes}(M_2'), \Gamma \vdash m \twoheadrightarrow n$, and whenever $n \in \mathrm{nodes}(M_2'')$, $\Gamma \vdash m \twoheadrightarrow n$. Thus, for any $n \in \mathrm{nodes}(M_2' \cup M_2'')$, we have $\Gamma \vdash m \twoheadrightarrow n$

(vi) $N \equiv n_i[\![\{v/x\}P_i]\!] \,|\, M_1 \,|\, M_2$: this follows again from Definition 2.1.3(2) and (6) above.

A.2 Algebraic properties of CNets

Proof of Lemma 3.2.1 We only prove the first statement, as the others are similar. Let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$, and suppose that $\mathcal{M} \between \mathcal{N}$ is defined.

We prove the two inclusions separately; first we prove that

$$(\mathsf{Input}(\mathcal{M}) \cup \mathsf{Input}(\mathcal{N})) \setminus \mathrm{nodes}(N) \subseteq \mathsf{Input}(\mathcal{M} \between \mathcal{N})$$

To this end, let $m \in (\mathsf{Input}(\mathcal{M}) \cup \mathsf{Input}(\mathcal{N})) \setminus \mathrm{nodes}(\mathcal{N})$. We need to show that $m \in \mathsf{Input}(\mathcal{M} \between \mathcal{N})$. As $\mathcal{M} \between \mathcal{N} = (\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N)$, this is equivalent to show that $(\Gamma_M \cup \Gamma_N) \vdash M \,|\, N \leftarrow m$.

We have two possible cases:

(a) $\Gamma_M \vdash M \leftarrow m$ and $m \notin \mathrm{nodes}(N)$. By definition (recall that a node in $\mathsf{Input}(\mathcal{M})$ is also included in $\mathsf{Int}(\mathcal{M})$) $m \notin \mathrm{nodes}(\mathcal{M})$ and $\Gamma_M \vdash n \leftarrow m$ for some $n \in \mathrm{nodes}(M)$. In particular, $m \notin \mathrm{nodes}(M \,|\, N)$, and $(\Gamma_M \cup \Gamma_N) \vdash n \leftarrow m$, with $n \in \mathrm{nodes}(M \,|\, N)$. Therefore, $(\Gamma_M \cup \Gamma_N) \vdash M \,|\, N \leftarrow m$

(b) $\Gamma_N \vdash N \leftarrow m$. By definition, $m \notin \mathrm{nodes}(N)$; further, there exists $n \in \mathrm{nodes}(N)$ such that $\Gamma_N \vdash n \leftarrow m$.

Since $m \in (\Gamma_N)_V$ and $\mathcal{M} \between \mathcal{N}$ is defined, it holds $m \notin \mathrm{nodes}(M)$. Thus, $m \notin \mathrm{nodes}(M \,|\, N)$ and $(\Gamma_M \cup \Gamma_N) \vdash n \leftarrow m$; by definition $(\Gamma_M \cup \Gamma_N) \vdash M \,|\, N \leftarrow m$.

Now we turn our attention to the opposite inclusion; that is, we want to show that

$$\mathsf{Input}(\mathcal{M} \between \mathcal{N}) \subseteq (\mathsf{Input}(\mathcal{M}) \cup \mathsf{Input}(\mathcal{N})) \setminus \mathrm{nodes}(N)$$

Let then m be a node such that $(\Gamma_M \cup \Gamma_N) \vdash M \,|\, N \leftarrow m$. That is, $m \notin \mathrm{nodes}(M)$, $m \notin \mathrm{nodes}(N)$, and there exists $n \in \mathrm{nodes}(M \,|\, N)$ such that $(\Gamma_M \cup \Gamma_N) \vdash n \leftarrow m$. There are two possibilities:

(a) $n \in \mathrm{nodes}(M)$. Since $\mathcal{M} \between \mathcal{N}$ is defined, we have that $\Gamma_N \nvdash n$. This implies that $\Gamma_N \vdash n \nleftarrow m$; since $(\Gamma_M \cup \Gamma_N) \vdash n \leftarrow m$, we have that $\Gamma_m \vdash n \leftarrow m$. Since $m \notin \mathrm{nodes}(M)$, this leads to $\Gamma_M \vdash M \leftarrow m$, or equivalently $m \in \mathsf{Input}(\mathcal{M})$; also, since $m \notin \mathrm{nodes}(N)$ we have that $m \in \mathsf{Input}(\mathcal{M}) \setminus \mathrm{nodes}(N)$

(b) $n \in \mathrm{nodes}(N)$. Since $\mathcal{N}$ is composable, and therefore well-formed, $\Gamma_N \vdash n$. For $\mathcal{M} \between \mathcal{N}$ is defined, we have $n \notin \mathrm{nodes}(M)$. Hence, it has to be $n \in \mathrm{nodes}(N)$. Also, since $m \notin \mathrm{nodes}(M)$, $n \notin \mathrm{nodes}(M)$, we have $\Gamma_M \vdash n \nleftarrow m$. This is because the network $\mathcal{M}$ is composable by hypothesis. Since $(\Gamma_M \cup \Gamma_N) \vdash n \leftarrow m$, it has necessarily to be $\Gamma_N \vdash n \leftarrow m$. By hypothesis $n \in \mathrm{nodes}(N)$, and we already noticed that $m \notin \mathrm{nodes}(N)$. Therefore, $m \in \mathsf{Input}(\mathcal{N})$, from which it follows $m \in (\mathsf{Input}(\mathcal{M}) \cup \mathsf{Input}(\mathcal{N})) \setminus \mathrm{nodes}(N)$.

Proof of Proposition 3.2.3 It is sufficient to check that $P_e(\mathcal{M},\mathcal{N}) = \text{true}$ and $P_e((\mathcal{M} \between \mathcal{N}),\mathcal{L}) = \text{true}$, iff $P_e(\mathcal{N},\mathcal{L}) = \text{true}$ and $P_e(\mathcal{M},(\mathcal{N} \between \mathcal{L})) = \text{true}$. To this end, let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$ and $\mathcal{L} = \Gamma_L \rhd L$. We prove the two implications separately.

Suppose $P_e(\mathcal{M},\mathcal{N}) = \text{true}$ and $P_e((\mathcal{M} \between \mathcal{N}),\mathcal{L}) = \text{true}$. By definition of P_e we have

$$(\text{nodes}(M) \cap (\Gamma_N)_V) = \emptyset \tag{A.1}$$

$$(\text{nodes}(M \mid N) \cap (\Gamma_L)_V) = \emptyset \tag{A.2}$$

We want to show that $\text{nodes}(N) \cap (\Gamma_L)_V = \emptyset$, and $\text{nodes}(M) \cap (\Gamma_N \cup \Gamma_L)_V = \emptyset$. The former statement is a straightforward consequence of Equation (A.2). The latter can be proved as follows: let $m \in \text{nodes}(M)$. By Equation (A.1) we have that $\Gamma_N \nvdash m$, so that it remains to show $\Gamma_L \nvdash m$. This is a trivial consequence of Equation (A.2); in fact, as $m \in \text{nodes}(M)$, we also have $m \in \text{nodes}(M \mid N)$, and therefore $\Gamma_L \nvdash m$.

Now suppose that $P_e(\mathcal{N},\mathcal{L}) = \text{true}$ and $P_e(\mathcal{M},(\mathcal{N} \between \mathcal{N})) = \text{true}$. By definition we have

$$(\text{nodes}(N) \cap (\Gamma_L)_V) = \emptyset \tag{A.3}$$

$$\text{nodes}(M) \cap (\Gamma_N \cup \Gamma_L)_V = \emptyset \tag{A.4}$$

We need to show that $\text{nodes}(M) \cap (\Gamma_N)_V = \emptyset$, and $\text{nodes}(M \mid N) \cap (\Gamma_L)_V = \emptyset$. The first statement is an immediate consequence of Equation (A.4), by noticing that $(\Gamma_N)_V \subseteq (\Gamma_N \cup \Gamma_L)_V$. For the second statement, let m be a node such that $\Gamma_L \vdash m$. By Equation (A.3) we have that $m \notin \text{nodes}(M)$. Also, by Equation (A.4) it holds that $m \notin \text{nodes}(M)$; in fact, since $\Gamma_L \vdash m$, we also have $\Gamma_N \cup \Gamma_L \vdash m$, and therefore $m \notin \text{nodes}(M)$. Since $m \notin \text{nodes}(M)$ and $m \notin \text{nodes}(N)$, we also have that $m \notin \text{nodes}(M \mid N)$, as we wanted to prove.

Proof of Theorem 3.2.6(1) Let $\mathcal{M} = \Gamma_M \rhd M$ be a composable network such that $M \equiv m[\![P]\!] \mid N$ for some node m, process P and system term N. We have to show that the network $\mathcal{G} = \Gamma_G \rhd m[\![P]\!]$, where Γ_G is defined by

$$(\Gamma_G)_V = \{m\} \cup \{n \in \mathsf{Int}(\Gamma_M \rhd M) \mid \Gamma_M \vdash m \leftrightarrows n\} \tag{A.5}$$

$$(\Gamma_G)_E = \{(m',n) \in (\Gamma_G)_V \mid \Gamma_M \vdash m' \to n\} \tag{A.6}$$

is a generating network. First note that $|\text{nodes}(m[\![P]\!])| = 1$, so that it suffices to show that $\mathcal{G}$ is composable. To this end, we show that $\mathcal{G}$ satisfies all the requirements of Definition 3.1.3.

1. $\Gamma_G \rhd m[\![P]\!]$ is well formed. This is trivial; in fact the system term $m[\![P]\!] \in \mathsf{sSys}$, as it has only one node with code associated. Further, $\text{nodes}(m[\![P]\!]) = \{m\}$, and $\Gamma_G \vdash m$ by Equation (A.5). It follows that $\text{nodes}(m[\![P]\!]) \subseteq (\Gamma_G)_V$.

2. Whenever $\Gamma_G \vdash n \to l$ for some nodes n, l, then either $n \in \text{nodes}(m[\![P]\!])$ or $l \in \text{nodes}(m[\![P]\!])$. Equivalently, we prove that whenever $\Gamma_G \vdash n \to l$ for some nodes n, l, then either $n = m$ or $l = m$.

   By Equation (A.6) we have that, whenever $\Gamma_G \vdash n \to l$, then $(\Gamma_G)_V \vdash n$ and $(\Gamma_G)_V \vdash l$. By Equation A.5 we have that either $n = m$ or $n \in \mathsf{Int}(\Gamma_M \rhd M)$ and $\Gamma_M \vdash m \leftrightarrows n$. If $n = m$, there is nothing to prove.

   Let then $n \in \mathsf{Int}(\Gamma_M \rhd M)$. By applying Equation A.5, this time to node l, we have that either $l = m$ or $l \in \mathsf{Int}(\Gamma_M \rhd M)$ with $\Gamma_M \vdash l \leftrightarrows m$. Again, in the former case there is nothing to prove, so suppose that $l \in \mathsf{Int}(\Gamma_M)$. By hypothesis we have that $\Gamma_G \vdash n \to l$. By Equation (A.6) it follows that $(\Gamma_G)_E \subseteq (\Gamma_M)_E$, so that we also know that $\Gamma_M \vdash n \to l$. For $\Gamma_M \rhd M$ is a composable network, then either $n \in \text{nodes}(M)$ or $l \in \text{nodes}(M)$. Without loss of generality, let $n \in \text{nodes}(M)$. We can show that $n = m$. In fact, recall that $\Gamma_G \vdash n$. Thus, either $n = m$, or $n \in \mathsf{Int}(\Gamma_M \rhd M)$; again, this follows from Equation (A.5). But we already know that $n \in \text{nodes}(M)$, so that the second case is not possible. Thus, the only possible case is $n = m$.

3. Whenever $n \in \mathsf{Int}(\Gamma_G \rhd P[\![])]\!]m$, then $\Gamma_G \vdash m \leftrightarrows n$. If $n \in \mathsf{Int}(\Gamma_G \rhd P[\![])]\!]m$, then $n \in \mathsf{Int}(\Gamma_M \rhd M)$; this is a

direct consequence of Equation A.5 and the fact that $n \neq m$ (note that $n \notin \text{nodes}(m[\![P]\!])$). We need to show that $\Gamma_G \vdash m \leftrightarrows n$.

To this end, let l be a node such that $\Gamma_G \vdash n \leftrightarrows l$. We show that $l = m$. By Equation (A.6) we have that $\Gamma_M \vdash n \leftrightarrows l$. Since $n \in \mathsf{Int}(\Gamma_M \triangleright M)$, and $\Gamma_M \triangleright M$ is a composable network, we have that $l \in \text{nodes}(M)$. Since $\Gamma_G \vdash l$, by Equation (A.5) we have that either $l \in \mathsf{Int}(\Gamma_M \triangleright M)$ or $l = m$. The former case is not possible, as we already proved that $l \in \text{nodes}(M)$. Thus it has to be $l = m$.

Proof of Theorem 3.2.6(2) Let $\mathcal{M} = \Gamma_M \triangleright M$ be a composable network such that $M \equiv m[\![P]\!] \,|\, N$ for some process P, node m and system term N. Let now Γ_N be defined by the equations below

$$(\Gamma_N)_V = \text{nodes}(N) \cup \{n \;\; \Gamma_M \vdash m' \leftrightarrows n \text{ for some } m' \in \text{nodes}(N)\} \tag{A.7}$$

$$(\Gamma_N)_E = (\Gamma_M)_E \setminus (\Gamma_G)_E \tag{A.8}$$

where $(\Gamma_G)_E$ has been defined in Equation A.6. We have to show that the network $\mathcal{N} = \Gamma_N \triangleright N$ is composable. To this end, it is sufficient to show that the network $\mathcal{N}$ satisfies all the requirements of Definition (3.1.3).

1. $\Gamma_N \triangleright N$ is well-formed. First note that $N \in \mathsf{sSys}$. In fact, by hypothesis we already know that $M \in \mathsf{sSys}$, and since $M \equiv m[\![P]\!] \,|\, N$, it follows that no node name in N appears more than once.

   It remains to show that $\text{nodes}(N) \subseteq (\Gamma_N)_V$. But this is a straight consequence of Equation (A.7).

2. Whenever $\Gamma_N \vdash n \to l$ for some nodes n and l, then either $n \in \text{nodes}(N)$ or $l \in \text{nodes}(N)$.

   Let n, l be two nodes such that $\Gamma_N \vdash n \to l$. By Equation (A.8) it follows that $n, l \in (\Gamma_N)_V$, and $\Gamma_M \vdash n \to l$. Since Γ_M is a composable network, we have that either $n \in \text{nodes}(M)$ or $l \in \text{nodes}(M)$.

   We only consider the case in which $n \in \text{nodes}(M)$, as the case $l \in \text{nodes}(M)$ is symmetric. Since $M \equiv m[\![P]\!] \,|\, N$, we have that either $n = m$ or $n \in \text{nodes}(N)$. In the latter case there is nothing to prove, while in the former we can perform a case analysis on node l.

   For this node, in fact, we have either $l \in \text{nodes}(M)$ or $l \in \mathsf{Int}(\mathcal{M})$. If $l \in \text{nodes}(M)$, then $l \neq m$, since from this it would follow $\Gamma_M \vdash m \to m$, which is not allowed. Since $l \neq m$, $l \in \text{nodes}(M)$ and $M \equiv m[\![P]\!] \,|\, N$, it has to be $l \in \text{nodes}(N)$.

   The case in which $l \in \mathsf{Int}(\mathcal{M})$ is not possible. In fact, in this case we would have $\Gamma_M \vdash m \to l$, with $l \in \mathsf{Int}(\mathcal{M})$. However, by Equation (A.6) it also follows $\Gamma_G \vdash m \to l$. Now we have $\Gamma_N \vdash m \to l$ and $\Gamma_G \vdash m \to l$, which contradicts Equation (A.6).

Proof of Theorem 3.2.6(4) Let $\mathcal{M} = \Gamma_M \triangleright M$ be a composable network such that $M \equiv m[\![P]\!] \,|\, N$ for some process P, node m and system term N. Let also $\mathcal{N} = \Gamma_N \triangleright N$, $\mathcal{G} = \Gamma_G \triangleright m[\![P]\!]$, where Γ_N is defined by equations (A.7)- (A.8) and Γ_G by equations (A.5)-(A.6).

We need to show that $(\Gamma_N)_V \cup (\Gamma_G)_V = (\Gamma_M)_V$ and $(\Gamma_N)_E \cup (\Gamma_G)_E = (\Gamma_M)_E$. The second statement follows directly from Equation (A.8).

For the first statement, it is trivial to prove that $(\Gamma_N)_V \cup (\Gamma_G)_V \subseteq (\Gamma_M)_V$; in fact, by Equation (A.7) we have $(\Gamma_N)_V \subseteq (\Gamma_M)_V$, while by Equation (A.5) it follows $(\Gamma_G)_V \subseteq (\Gamma_M)_V$.

It remains to show that $(\Gamma_M)_V \subseteq (\Gamma_N)_V \cup (\Gamma_G)_V$. To this end, suppose $\Gamma_M \vdash n$. Then either $n \in \text{nodes}(M)$ or $n \in \mathsf{Int}(\mathcal{M})$. In the first case we perform a case analysis on n. If $n = m$ then $\Gamma_G \vdash n$ by Equation (A.5); If $n \neq m$ then $n \in \text{nodes}(N)$, and therefore $\Gamma_N \vdash n$ by Equation (A.6).

Consider now the case in which $n \in \mathsf{Int}(\mathcal{M})$. Since $\mathcal{M}$ is a composable network, there exists a node $l \in \text{nodes}(M)$ such that $\Gamma_M \vdash n \leftrightarrows l$. In this case we perform a case analysis on l.
If $l = m$, then we have that $\Gamma_G \vdash n$ by Equation (A.5). If $l \neq m$, then $l \in \text{nodes}(N)$. Thus, by Equation (A.7), we obtain $\Gamma_N \vdash n$.

We have proved that whenever $\Gamma_M \vdash n$, then either $\Gamma_G \vdash N$ or $\Gamma_N \vdash n$; therefore, $(\Gamma_M)_V \subseteq (\Gamma_G)_V \cup (\Gamma_N)_V$.

A.3 Properties of Extensional Actions

Proof of Proposition 4.1.4 Let $\mathcal{M}, \mathcal{N}$ be networks, and suppose $\mathcal{M} = \Gamma \rhd M$ for some network connectivity Γ and $M \in \mathsf{tSys}$.

For the first statement, suppose $\mathcal{M} \twoheadrightarrow \mathcal{N}$. By Theorem 2.4.10 we have that either

1. $\mathcal{M} \xrightarrow{m.\tau} \mathcal{N}'$ for some node m and network $\mathcal{N}'$ such that $\mathcal{N}' \equiv \mathcal{N}$. By Definition 4.1.3(1) it follows that $\mathcal{M} \xmapsto{\tau} \mathcal{N}'$.

2. $\mathcal{M} \xrightarrow{m.c!v} \mathcal{N}'$ for some node m, channel c, value v and network $\mathcal{N}'$ such that $\mathcal{N}' \equiv \mathcal{N}$. We have two possible sub-cases:

   - $\{n \in \mathsf{Output}(\Gamma \rhd M) \mid \Gamma \vdash m \to n\} = \emptyset$; in this case we can apply Definition 4.1.3(1) to obtain $\mathcal{M} \xmapsto{\tau} \mathcal{N}'$
   - $\{n \in \mathsf{Output}(\Gamma \rhd M) \mid \Gamma \vdash m \to n\} \neq \emptyset$. Let the former set be denoted by η; in this case we have that $\mathcal{M} \xmapsto{c!v \rhd \eta}$ as a direct consequence of Definition 4.1.3(3)

Let us prove the second statement; suppose $\mathcal{M} \xmapsto{\tau} \mathcal{N}$; by Definition 4.1.3(1) there are two possible cases.

1. $\mathcal{M} \xrightarrow{m.\tau} \mathcal{N}$ for some node m; in this case the Harmony Theorem ensures that $\mathcal{M} \twoheadrightarrow \mathcal{N}$

2. $\mathcal{M} \xrightarrow{m.c!v} \mathcal{N}$ for some node m, channel c and value v such that $\{n \in \mathsf{Output}(\Gamma \rhd M) \mid \Gamma \vdash m \to n\} = \emptyset$. Since we have the broadcast transition $\mathcal{M} \xrightarrow{m.c!v} \mathcal{N}$, it follows again from the Harmony Theorem that $\mathcal{M} \twoheadrightarrow \mathcal{N}$

It remains to prove the last statement; let $\mathcal{M} \xmapsto{c!v \rhd \eta} \mathcal{N}$. In this case we have that $\mathcal{M} \xrightarrow{m.c!v} \mathcal{N}$ for some node m, channel c and value v such that $\{n \in \mathsf{Output}(\Gamma \rhd M) \mid \Gamma \vdash m \to n\} \neq \emptyset$. By an application of the Harmony Theorem to the transition $\mathcal{M} \xrightarrow{m.c!v} \mathcal{N}$ we obtain $\mathcal{M} \twoheadrightarrow \mathcal{N}$, as we wanted to prove.

Proof of Proposition 4.1.8 The two implications are proved separately.

For the if implication, let c a channel, v a value, k be a strictly positive integer and $\eta_1, \cdots, \eta_k$ be non-empty sets of nodes such that for any $i, j : i \neq j, 1 \leq i, j \leq k, \eta_i \cap \eta_j = \emptyset$. Further, let $\mathcal{M}, \mathcal{N}$ be two networks such that

$$\mathcal{M} \overset{\tau}{\models\!\!\Longrightarrow} \overset{c!v \rhd \eta_1}{\longmapsto} \overset{\tau}{\models\!\!\Longrightarrow} \cdots \overset{\tau}{\models\!\!\Longrightarrow} \overset{c!v \rhd \eta_k}{\longmapsto} \overset{\tau}{\models\!\!\Longrightarrow} \mathcal{N}$$

We show that $\mathcal{M} \overset{c!v \rhd \left(\bigcup_{j=1^k} \eta_j\right)}{\models\!\!\Longrightarrow} \mathcal{N}$ by performing a natural induction on k.

$k = 1$ In this case we have that $\mathcal{M} \overset{\tau}{\models\!\!\Longrightarrow} \overset{c!v \rhd \eta_1}{\longmapsto} \overset{\tau}{\models\!\!\Longrightarrow} \mathcal{N}$. By definition 4.1.5(3b) it follows that $\mathcal{M} \overset{c!v \rhd \eta}{\models\!\!\Longrightarrow} \mathcal{N}$.

$k > 1$ Suppose that the statement is true for any $j : j \geq 1, j < k$. In particular, for an arbitrary index $j < k$ there exists a network $\mathcal{M}'$ such that

$$\mathcal{M} \overset{\tau}{\models\!\!\Longrightarrow} c!v \rhd \eta_1 \overset{\tau}{\models\!\!\Longrightarrow} \cdots \overset{\tau}{\models\!\!\Longrightarrow} \overset{c!v \rhd \eta_j}{\longmapsto} \overset{\tau}{\models\!\!\Longrightarrow} \mathcal{M}'$$

$$\mathcal{M}_j \xmapsto{\tau} c!v \rhd \eta_{j+1} \overset{\tau}{\models\!\!\Longrightarrow} \cdots \overset{\tau}{\models\!\!\Longrightarrow} \overset{c!v \rhd \eta_k}{\longmapsto} \overset{\tau}{\models\!\!\Longrightarrow} \mathcal{N}$$

By inductive hypothesis it holds that $\mathcal{M} \overset{c!v \rhd \left(\bigcup_{i=1}^{j} \eta_i\right)}{\models\!\!\Longrightarrow} \mathcal{M}'$, $\mathcal{M}' \overset{c!v \rhd \left(\bigcup_{i=j+1}^{k} \eta_i\right)}{\models\!\!\Longrightarrow} \mathcal{N}$. We show that $\left(\bigcup_{i=1}^{j} \eta_i\right) \cap \left(\bigcup_{i=j+1}^{k} \eta_i\right) = \emptyset$, from which it follows from Definition 4.1.5(3b) that $\mathcal{M} \overset{c!v \rhd \left(\bigcup_{i=1}^{k} \eta_i\right)}{\models\!\!\Longrightarrow} \mathcal{N}$.

Let $m \in \left(\bigcup_{i=1}^{j} \eta_i\right), n \in \left(\bigcup_{i=j+1} \eta_i\right)$. Then there exist an index $i' \leq j$ such that $m \in \eta_{i'}$ and an index $i'' : i'' > j, i \leq k$ such that $n \in \eta_{i''}$. Since $i' \neq i''$, it follows that $\eta_{i'} \cap \eta_{i''} = \emptyset$, hence $m \neq n$. For we chose the nodes m, n arbitrarily, we obtain that $\left(\bigcup_{i=1}^{j} \eta_i\right) \cap \left(\bigcup_{i=j+1}^{k} \eta_i\right) = \emptyset$, as we wanted to prove.

Now we prove the only if implication. Suppose that $\mathcal{M} \overset{c!v \triangleright \eta}{\models\!\!\Longrightarrow} \mathcal{N}$ for some non-empty set of nodes η. We prove that

$$\mathcal{M} \overset{\tau}{\models\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta_1}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \cdots \overset{\tau}{\vdash\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta_k}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \mathcal{N}$$

for some index $k \geq 1$, and collection of pairwise disjoint, non-empty sets of nodes $\{\eta_i\}_{i=1}^{k}$ such that $\bigcup_{i=1}^{k} \eta_i = \eta$; this is done by performing an induction on the proof of the Transition $\mathcal{M} \overset{c!v \triangleright \eta}{\models\!\!\Longrightarrow} \mathcal{N}$.

Base Case $\mathcal{M} \overset{\tau}{\models\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \mathcal{N}$.

In this case it is sufficient to choose $k = 1$, $\eta_1 = \eta$; the statement follows from Definition 4.1.5(3b).

Inductive Step $\mathcal{M} \overset{c!v \triangleright \eta'}{\models\!\!\Longrightarrow} \mathcal{M}' \overset{c!v \triangleright \eta''}{\models\!\!\Longrightarrow} \mathcal{N}$ for some network $\mathcal{M}'$, non-empty set of nodes η', η'' such that $\eta' \cap \eta'' = \emptyset$, $\eta' \cup \eta'' = \eta$.

By Inductive hypothesis there exist two indexes k', k'' and two collections of pairwise disjoint, non-empty sets of nodes $\{\eta'_i\}_{i=1}^{k'}$, $\{\eta''_i\}_{i=1}^{k''}$ such that $\eta' = \bigcup_{i=1}^{k'} \eta'_i$, $\eta'' = \bigcup_{i=1}^{k''} \eta''_1$. Further,

$$\mathcal{M} \overset{\tau}{\models\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta'_1}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \cdots \overset{\tau}{\vdash\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta'_{k'}}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \mathcal{M}'$$
$$\mathcal{M}' \overset{\tau}{\models\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta''_1}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \cdots \overset{\tau}{\vdash\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta''_{k''}}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \mathcal{N}$$

Let $k = k' + k''$, for any $i \leq k', \eta_i = \eta'_i$ and for any $i : k' < i \leq k'', \eta_i = \eta''_{i-k'}$. Then we have that

$$\begin{aligned}
\bigcup_{i=1}^{k} \eta_i &= \left(\bigcup_{i=1} k' \eta_i\right) \cup \left(\bigcup_{i=1} k'' \eta_{i-k'}\right) \\
&= \left(\bigcup_{i=1}^{k'} \eta'\right) \cup \left(\bigcup_{i=1}^{k''} \eta''_i\right) \\
&= \eta' \cup \eta'' = \eta
\end{aligned}$$

Further we also have that

$$\mathcal{M} \overset{\tau}{\models\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta_1}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \cdots \overset{\tau}{\vdash\!\!\Longrightarrow\!\!\vdash} \overset{c!v \triangleright \eta_k}{\longrightarrow} \overset{\tau}{\vdash\!\!\Longrightarrow} \mathcal{N}$$

It remains to show that all the sets in the collection $\{\eta_i\}_{i=1,k}$ are disjoint. To this end, let j, j' be two indexes such that $1 \leq j', j'' \leq k$, $j' \neq j''$. If $j', j'' \leq k'$, then it holds that $\eta_{j'} = \eta'_{j'}$, $\eta_{j''} = \eta'_{j''}$, and $\eta'_{j'} \cap \eta'_{j''} = \emptyset$ by inductive hypothesis. Similarly, if $k' < j', j'' \leq k''$, then $\eta_{j'} = \eta''_{j'-k'}$, $\eta_{j''} = \eta''_{j''-k'}$, and since $j' - k' \neq j'' - k'$ it follows by inductive hypothesis that $\eta''_{j'-k'} \cap \eta''_{j''-k'} = \emptyset$.

The last case we need to check is the one in which $j' \leq k'$, $k' < j'' < k''$. Then $\eta_{j'} = \eta'_{j'}$, $\eta_{j''} = \eta''_{j''-k'}$. For any nodes $m \in \eta_{j'}, n \in \text{nodes}(\eta_{j''})$, we obtain that $m \in \bigcup_{i=1}^{k'} \eta'_i = \eta'$, $n \in \bigcup_{i=1}^{k''} \eta''_i = \eta''$. For $\eta' \cap \eta'' = \emptyset$ by hypothesis, it follows that $m \neq n$. Since we chose the nodes m, n arbitrarily in the sets $\eta_{j'}$, $\eta_{j''}$, respectively, it follows that $\eta_{j'} \cap \eta_{j''} = \emptyset$, as we wanted to prove.

Proof of Proposition 4.2.3 The two statements are proved separately. Let $\mathcal{M} = \Gamma_M \triangleright M, \mathcal{N} = \Gamma_N \triangleright N$ be networks. We need to show the following:

(i) $\text{sym}_{\mathcal{M}}(\mathcal{N})$ is a network. To prove this statement, it suffices to show that $\text{sym}_{\mathcal{M}}(\mathcal{N})$ satisfies all the requirements of composable networks, Definition 3.1.3, by showing that $N \in \mathsf{tSys}$ rather than $N \in \mathsf{sSys}$. In the proof below, recall that $\text{sym}_{\mathcal{M}}(\mathcal{N}) = \Gamma'_N \triangleright N$, where Γ'_N is defined according to the equations (4.1) and (4.2).

- $\Gamma'_N \triangleright N$ is well-formed. We already know that $N \in \mathsf{tSys}$, for $\Gamma_N \triangleright N$ is a network by hypothesis.

Suppose that $\Gamma'_N \vdash m \to n$; we need to show that $m \neq n$. By Equation 4.2 we have that either $\Gamma_N \vdash m \neq n$ or $\Gamma_M \vdash m \neq n$. Since $\Gamma_M \rhd M$ an $\Gamma_N \rhd N$ are composable networks, in both cases it holds that $m \neq n$.

Finally, let $n \in \mathrm{nodes}(N)$; we need to show that $\Gamma'_N \vdash n$. This is trivial; by Equation 4.1 it follows that $(\Gamma_N)_V \subseteq (\Gamma'_N)_V$. Since $\Gamma_N \rhd N$ is a composable network, if $n \in \mathrm{nodes}(N)$ then $\Gamma_N \vdash n$, and consequently $\Gamma'_N \vdash n$.

- Suppose that $\Gamma'_N \vdash m \to n$. We need to show that either $m \in \mathrm{nodes}(N)$ or $n \in \mathrm{nodes}(N)$. According to Equation 4.1 we have three possible cases:
  - $\Gamma_N \vdash m \to n$; since $\Gamma_N \rhd N$ is a composable network, it holds that either $m \in \mathrm{nodes}(N)$ or $n \in \mathrm{nodes}(N)$
  - $\Gamma_M \vdash m \to n$, with $m \in \mathrm{nodes}(N)$; in this case there is nothing to prove
  - $\Gamma_M \vdash m \to n$, with $n \in \mathrm{nodes}(N)$; again, in this case there is nothing to prove
- If $m \in \mathsf{Int}(\Gamma'_N \rhd N)$, then there exists $n \in \mathrm{nodes}(N)$ such that $\Gamma'_N \vdash m \rightleftarrows n$. Since $m \in \mathsf{Int}(\Gamma'_N \rhd N)$, we have that $m \notin \mathrm{nodes}(N)$, and $\Gamma'_N \vdash m$. By Equation 4.1 there are two possible scenarios:
  - $\Gamma_N \vdash m$. Then $m \in \mathsf{Int}(\Gamma_N \rhd N)$ (recall that $m \notin \mathrm{nodes}(N)$). For $\Gamma_N \rhd N$ is a composable network, there exist $n \in \mathrm{nodes}(N)$ such that $\Gamma_N \vdash m \rightleftarrows n$. By Equation 4.2 it follows that $(\Gamma_N)_E \subseteq (\Gamma'_N)_E$, so that we also have $\Gamma'_N \vdash m \rightleftarrows n$.
  - $\Gamma_M \vdash m \rightleftarrows n$ for some $n \in \mathrm{nodes}(N)$. By Equation 4.2 it follows immediately that $\Gamma'_N \vdash m \rightleftarrows n$.

(ii) $\mathrm{ext}_{\mathcal{M}}(\mathcal{N})$ is a composable network. Again, we need to check that this network satisfies the conditions satisfied by composable networks, where the constraint $N \in \mathsf{sSys}$ is replaced with $N \in \mathsf{tSys}$. Recall that $\mathrm{ext}_{\mathcal{M}}(\mathcal{N}) = \Gamma''_N \rhd N$, where Γ''_N is defined by equations (4.3) and (4.4)

- Γ''_N is well-formed. We already know that $N \in \mathsf{tSys}$, since $\Gamma_N \rhd N$ is a composable network.

  By Equation (4.4) it follows that $(\Gamma''_N)_E \subseteq (\Gamma_N)_E$. Thus, if $\Gamma''_N \vdash m \to n$, we have that $\Gamma_N \vdash m \to n$. Since $\Gamma_N \rhd N$ is a composable network, it follows that $m \neq n$.

  It remains to prove that $\mathrm{nodes}(N) \subseteq (\Gamma''_N)_V$. Recall that we are assuming that $\mathrm{nodes}(N) \cap \mathrm{nodes}(M) = \emptyset$. If $m \in \mathrm{nodes}(N)$, then $\Gamma_N \vdash m$. Again this is because $\Gamma \rhd N$ is a composable network by hypothesis. For $\mathrm{nodes}(N) \cap \mathrm{nodes}(M) = \emptyset$, we also have $m \in (\Gamma_N \setminus \mathrm{nodes}(M))$, the latter set being exactly $(\Gamma''_N)_V$ by Equation (4.3).
- Whenever $\Gamma''_N \vdash m \to n$, then either $m \in \mathrm{nodes}(N)$ or $n \in \mathrm{nodes}(N)$.

  Suppose that $\Gamma''_N \vdash m \to n$ for some nodes m, n. By Equation (4.4) it is immediate to notice that $(\Gamma''_N)_E \subseteq (\Gamma_N)_E$, so that $\Gamma_N \vdash m \to n$. Since $\Gamma_N \rhd N$ is a composable network, then either $m \in \mathrm{nodes}(N)$ or $n \in \mathrm{nodes}(N)$.
- If $m \in \mathsf{Int}(\Gamma''_N \rhd N)$, then there exists $n \in \mathrm{nodes}(N)$ such that $\Gamma''_N \vdash m \rightleftarrows n$.

  Let $m \in \mathsf{Int}(\Gamma''_N \rhd N)$. Then $m \notin \mathrm{nodes}(N)$ and $\Gamma''_N \vdash m$. By Equation (4.3) it follows that $(\Gamma''_N)_V \subseteq (\Gamma_N)_V$, hence $\Gamma_N \vdash N$. Now we have that $m \in \mathsf{Int}(\Gamma_N \rhd N)$, since $m \notin \mathrm{nodes}(N)$. Since $\Gamma_N \rhd N$ is a composable network, there exists a node $n \in \mathrm{nodes}(N)$ such that $\Gamma_N \vdash m \leftrightarrows n$. We prove that $\Gamma''_N \vdash m \to n$. For $\Gamma''_N \vdash m$, $\Gamma''_N \vdash n$ (the latter being an immediate consequence of $n \in \mathrm{nodes}(N)$), by Equation (4.4) it follows $\Gamma''_N \vdash m \to n$, as we wanted to prove.

Proof of Proposition 4.2.4 The two statements are proved separately.

(i) Let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$ be two composable networks such that $(\Gamma_N)_V \cap \mathrm{nodes}(M) = \emptyset$. Then $\mathrm{sym}_{\mathcal{M}}(\mathcal{N}) = \Gamma'_N \rhd N$, where Γ'_N is the connectivity graph defined according to equations (4.1) and (4.2). Further, let $\mathrm{ext}_{\mathcal{M}}(\mathrm{sym}_{\mathcal{M}}(\mathcal{N})) = \mathrm{ext}_{\mathcal{M}}(\Gamma'_N \rhd N) = \Gamma''_N \rhd N$, where Γ''_N is defined by equations (4.3) and (4.4).

To prove that $\Gamma_N \rhd N = \Gamma''_N \rhd N$ it is sufficient to show that $(\Gamma_N)_V = (\Gamma''_N)_V$, and $(\Gamma_N)_E = (\Gamma''_N)_E$. We proceed by proving these two statements separately.

- $(\Gamma_N)_V = (\Gamma''_N)_V$. We first prove that $(\Gamma_N)_V \subseteq (\Gamma''_N)_V$, then we show that $(\Gamma''_N)_V \subseteq (\Gamma_N)_V$.

  Let m be a node such that $\Gamma_N \vdash m$. We show that $\Gamma''_N \vdash m$, Since we are assuming that $(\Gamma_N)_V \cap \text{nodes}(M) = \emptyset$, it follows that $m \notin \text{nodes}(M)$. By Equation (4.1) it holds that $(\Gamma_N)_V \subseteq (\Gamma'_N)_V$, hence $\Gamma'_N \vdash m$. For $m \notin \text{nodes}(M)$, we also have that $m \in (\Gamma'_N) \setminus \text{nodes}(M)$; by Equation (4.3) the last set is exactly Γ''_N, thus $\Gamma''_N \vdash m$. We have proved that, for any node m, if $\Gamma_N \vdash m$ then $\Gamma''_N \vdash m$; that is, $(\Gamma_N)_V \subseteq (\Gamma''_N)_V$.

  Now suppose that m is a node such that $\Gamma''_N \vdash m$. We prove that $\Gamma_N \vdash m$, thus showing that $(\Gamma''_N)_V \subseteq (\Gamma_N)_V$. Since $\Gamma''_N \vdash m$, it follows by Equation 4.3 that $\Gamma'_N \vdash m$ and $m \notin \text{nodes}(M)$. By Equation (4.1) we obtain that either $\Gamma_N \vdash m$, in which case there is nothing to prove, or $\Gamma_M \vdash m \rightleftarrows n$ for some $n \in \text{nodes}(N)$. However, this last case is not possible.

  Suppose in fact that $\Gamma_M \vdash m \rightleftarrows n$ for some $n \in \text{nodes}(N)$. Since we are assuming that $(\Gamma_N) \cap \text{nodes}(M) = \emptyset$, and $n \in \text{nodes}(N)$, then we obtain than $n \notin \text{nodes}(M)$ (recall that $\text{nodes}(N) \subseteq (\Gamma_N)_V$, since by hypothesis $\Gamma_N \triangleright N$ is a composable network, hence well-formed). Recall that we have already proved that $m \notin \text{nodes}(M)$. Then, we would have that $\Gamma_M \vdash m \rightleftarrows n$, where $m, n \notin \text{nodes}(M)$. This last statement contradicts the assumption that $\Gamma_M \triangleright M$ is a composable network. In fact, it violates Requirement (2) in Definition 3.1.3

- $(\Gamma_N)_E = (\Gamma''_N)_E$. Again, we first show that $(\Gamma_N)_E \subseteq (\Gamma''_N)_E$, then we prove that $(\Gamma''_N)_E \subseteq (\Gamma_N)_E$.

  Let m, n be two nodes such that $\Gamma_N \vdash m \to n$. To prove that $(\Gamma_N)_E \subseteq (\Gamma''_N)_E$ it is sufficient to show that $\Gamma''_N \vdash m \to n$. Since $\Gamma_N \vdash m \to n$, it follows that $\Gamma_N \vdash m, n$. For $(\Gamma_N)_V \cap \text{nodes}(M) = \emptyset$, it also holds that $m, n \notin \text{nodes}(M)$. Now, by Equation (4.2) we obtain that $(\Gamma_N)_E \subseteq (\Gamma'_N)_E$, hence $\Gamma'_N \vdash m \to n$. The last statement induces that $\Gamma'_N \vdash m, n$. For $m, n \notin \text{nodes}(M)$, we have that $\Gamma''_N \vdash m, n$ as an immediate consequence of Equation (4.3). Finally, since $\Gamma'_N \vdash m \to n$, $\Gamma''_N \vdash m, n$, Equation (4.4) ensures that $\Gamma''_N \vdash m \to n$, as we wanted to prove.

  Suppose now that $\Gamma''_N \vdash m \to n$ for some nodes m, n. We show that $\Gamma_N \vdash m \to n$. Since $\Gamma''_N \vdash m \to n$, by Equation (4.4) it follows that $\Gamma'_N \vdash m \to n$; further $\Gamma''_N \vdash m, n$, hence by Equation 4.3 it follows that $m, n \notin \text{nodes}(M)$. Since $\Gamma'_N \vdash m \to n$, according to Equation 4.2 we have two possible scenarios.

  - $\Gamma_N \vdash m \to n$, in which case there is nothing to prove.
  - $\Gamma_M \vdash m \to n$ for some $n \in \text{nodes}(N)$; this case is not possible. In fact, we already proved that $m \notin \text{nodes}(M)$, $n \notin \text{nodes}(M)$. This contradicts the hypothesis that $\Gamma_M \triangleright M$ is a composable network.

(ii) Let $\mathcal{M} = \Gamma_M \triangleright M$, $\mathcal{N} = \Gamma_N \triangleright N$, and suppose that $P_s(\mathcal{M}, \mathcal{N}) = \text{true}$. Then we have that $\text{ext}_{\mathcal{M}}(\mathcal{N}) = \Gamma'_N \triangleright N$, where Γ'_N is defined by equations (4.3) and (4.4). Also, we have that $\text{sym}_{\mathcal{M}}(\text{ext}_{\mathcal{M}}(\mathcal{N})) = \text{sym}_{\mathcal{M}}(\Gamma'_N \triangleright N) = \Gamma''_N \triangleright N$, where Γ''_N is defined according to equations (4.1) and (4.2).

We want to prove that $\Gamma_N \triangleright N = \Gamma''_N \triangleright N$. To this end, it is sufficient to show that $(\Gamma_N)_V = (\Gamma''_N)_V$ and $(\Gamma_N)_E = (\Gamma''_N)_E$. These two statements are proved individually.

- $(\Gamma_N)_V = (\Gamma''_N)_V$. We first show that $(\Gamma_N)_V \subseteq (\Gamma''_N)_V$, then we prove that $(\Gamma''_N)_V \subseteq (\Gamma_N)_V$.

  Let m be a node such that $\Gamma_N \vdash m$. If we show that $\Gamma''_N \vdash m$, then we obtain as an immediate consequence that $(\Gamma_N)_V \subseteq (\Gamma''_N)_V$. For node m, we have two possible cases:

  - $m \notin \text{nodes}(M)$. Since we are assuming that $\Gamma_N \vdash m$, it follows immediately from Equation (4.3) that $\Gamma'_N \vdash m$. Further, by Equation (4.1) we have that $(\Gamma'_N)_V \subseteq (\Gamma''_N)_V$, so that $\Gamma''_N \vdash m$ also holds.
  - $m \in \text{nodes}(M)$. Recall that, by hypothesis, we have that $P_s(\Gamma_N \triangleright M, \Gamma_N \triangleright N) = \text{true}$. By definition of P_s, Definition 3.1.6, it follows that $\text{nodes}(M) \cap \text{nodes}(N) = \emptyset$. Therefore, $m \in \text{nodes}(M)$ implies that $m \notin \text{nodes}(N)$.

    On the other hand, we are assuming that $\Gamma_N \vdash m$. Thus, if $m \notin \text{nodes}(N)$, it has to be $n \in \mathsf{Int}(\Gamma_N \triangleright N)$. Now notice that, in this case, there exists a node $n \in \text{nodes}(N)$ such that $\Gamma_N \vdash m \to n$; this is because, by hypothesis, $\Gamma_N \triangleright N$ is a composable network, hence it satisfies Requirement

(2) in Definition (3.1.3). For we are assuming that $P_s(\Gamma_M \rhd M, \Gamma_N \rhd N) = \text{true}$, and $\Gamma_N m \to n$ with $m \in \text{nodes}(M)$, by Definition (3.1.6) it follows that $\Gamma_M m \to n$. Not it remains to notice that $n \in \text{nodes}(N)$, so that by Equation (4.1) we obtain that $\Gamma''_M m \to n$.

Let us prove now that $(\Gamma''_N)_V \subseteq (\Gamma_N)_V$. In this case it is sufficient to prove that, for any node m, $\Gamma''_N \vdash m$ implies $\Gamma_N \vdash m$.

Let then m be a node such that $\Gamma''_N \vdash m$. By Equation (4.1) we have two possible cases:

- $\Gamma'_N \vdash m$. For $(\Gamma_N)_V \subseteq (\Gamma'_N)_V$ as a simple consequence of Equation (4.3), it follows that $\Gamma_N \vdash m$.
- Otherwise there exists a node $n \in \text{nodes}(N)$ such that $\Gamma_M \vdash m \to n$. Since $P_s(\Gamma_M \rhd M, \Gamma_N \rhd N) = \text{true}$, we have that $\Gamma_N \vdash m \to n$.

- $(\Gamma_N)_E = (\Gamma''_N)_E$. Again, we first show that $(\Gamma_N)_E \subseteq (\Gamma''_N)_E$, then we prove that $(\Gamma''_N)_E \subseteq (\Gamma_N)_E$.

  For the first inclusion, we need to show that whenever $\Gamma_N \vdash m \to n$ for some nodes m, n, then it also holds $\Gamma''_N \vdash m \to n$. To this end, let m, n be two nodes such that $\Gamma_N \vdash m \to n$. Since $\Gamma_N \rhd N$ is a composable network by hypothesis, in this case we have that either $m \in \text{nodes}(N)$ or $n \in \text{nodes}(N)$. We consider only the case where $n \in \text{nodes}(N)$, as the proof for the other case is similar.

  Suppose then $n \in \text{nodes}(N)$. We perform a case analysis on node m:

  - $m \in \text{nodes}(M)$. In this case, we have that $\Gamma_M \vdash m \to n$; this is because we are assuming that $\Gamma_N \vdash m \to n$ and $P_s(\Gamma_M \rhd M, \Gamma_N \rhd N) = \text{true}$. Now it follows by Equation (4.2) that $\Gamma''_N \vdash m \to n$.
  - $m \notin \text{nodes}(M)$. Recall that we are assuming that $P_s(\Gamma_M \rhd M, \Gamma_N \rhd N) = \text{true}$, from which it follows $\text{nodes}(M) \cap \text{nodes}(N) = \emptyset$. Then, since $n \in \text{nodes}(N)$, we have $n \notin \text{nodes}(M)$.
    Thus, for nodes m, n we have $\Gamma_N \vdash m, n$ and $m, n \notin \text{nodes}(M)$; by Equation (4.3) we obtain that $\Gamma'_N \vdash m, n$, and by Equation (4.4) it follows that $\Gamma'_N \vdash m \to n$. Now it suffices to note that $(\Gamma'_N)_E \subseteq (\Gamma''_N)_E$ as a simple consequence of Equation (4.2) to infer that $\Gamma''_N \vdash m \to n$.

  Now we show that $(\Gamma''_N)_E \subseteq (\Gamma_N)_E$. To this end, let m, n be two nodes such that $\Gamma''_N \vdash m \to n$. We prove that $\Gamma_N \vdash m \to n$. By Equation (4.2) we have two possible cases:

  - $\Gamma'_N \vdash m \to n$. Then, since $(\Gamma'_N)_E \subseteq (\Gamma_N)_E$ by Equation (4.4), it follows that $\Gamma_N \vdash m \to n$, as we wanted to prove.
  - $\Gamma_M \vdash m \to n$, where either $m \in \text{nodes}(N)$ or $n \in \text{nodes}(N)$. We only consider the last case, as the proof for the first is similar.
    If $\Gamma_M \vdash m \to n$, and $n \in \text{nodes}(N)$, then it follows immediately that $\Gamma_N \vdash m \to n$. This is because, by hypothesis, we have that $P_s(\Gamma_M \rhd M, \Gamma_N \rhd N) = \text{true}$.

Proof of Lemma 4.2.9 The proof is performed by Rule induction on the proof of the transition $\Gamma \rhd M \xrightarrow{\lambda} M'$. We only consider the most interesting cases; in each of them, let Γ' be a connectivity graph which satisfies the hypothesis stated in the Lemma.

Suppose that the last rule applied in the proof of the transition $\Gamma \rhd M \xrightarrow{\lambda} M'$ is Rule (B-REC). Then $\lambda = n.c?v$ for some channel c, value v and node n. $M = m[\![P]\!]$ for some node m and processes P, P' such that $\Gamma \vdash m \leftarrow n$, $P \xrightarrow{c?v} P'$; finally, $M' = m[\![P']\!]$. Since $m \in \text{nodes}(M)$, and $\Gamma \vdash m \to n$, we have that $\Gamma' \vdash m \to n$. By a simple application of Rule (B-REC) we obtain $\Gamma' \rhd m[\![P]\!] \xrightarrow{n.c?v} M'[\![P]\!]$.

If the last rule applied in the proof of $\Gamma \rhd M \xrightarrow{\lambda} M'$ is Rule (B-DISC), then $\lambda = n.c?v$ for some node n, channel c and value v. Further, $M = M' = m[\![P]\!]$ for some node m and process P. Further we have that $\Gamma \vdash m \nleftarrow n$. For $\Gamma' \subseteq \Gamma$, then $\Gamma' \vdash m \nleftarrow n$; now, by an application of Rule (B-DISC), we obtain that $\Gamma' \rhd m[\![P]\!] = m[\![P]\!]$, as we wanted to prove.

Finally, suppose the last rule applied in the proof of $\Gamma \rhd M \xrightarrow{\lambda} M'$ is Rule (B-PROP). Then we have that $\lambda = n.c?v$ for some node n, channel c and value v; further, $M = M_1 \mid M_2$, for some system terms M_1, M_2, M'_1 and M'_2 such that $\Gamma \rhd M_1 \xrightarrow{n.c?v} M'_1$, $\Gamma \rhd M_2 \xrightarrow{n.c?v} M'_2$ and $M' = M'_1 \mid M'_2$.

Now let Γ' be a connectivity graph such that graphs such that $\Gamma' \subseteq \Gamma$, $\text{nodes}(M) \subseteq (\Gamma')_V$ and whenever $\Gamma \vdash m \to n$, with either $m \in \text{nodes}(M)$ or $n \in \text{nodes}(M)$, then $\Gamma' \vdash m \to n$. In this case it is trivial to show that

$\mathrm{nodes}(M_1) \subseteq \mathrm{nodes}(M) \subseteq (\Gamma_1)_V$, and if $\Gamma \vdash m \to n$, with wither $m \in \mathrm{nodes}(M_1)$ or $n \in \mathrm{nodes}(M_1)$, then either $m \in \mathrm{nodes}(M)$ or $n \in \mathrm{nodes}(N)$, hence $\Gamma' \vdash m \to n$. The same properties can be shown for the network $\Gamma' \rhd M_2$ as well.

Thus we can apply the inductive hypothesis to M_1, N_1, leading to $\Gamma' \rhd M_1 \xrightarrow{n.c?v} M_1'$, $\Gamma' \rhd M_2 \xrightarrow{n.c?v} M_2'$. Thus, with an application of Rule (B-PROP), we can infer the transition $\Gamma' \rhd M_1 \,|\, M_2 \xrightarrow{n.c?v} M_1' \,|\, M_2'$.

Proof of Proposition 4.2.12 Let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$ and $\mathcal{L} = \Gamma_L \rhd L$ be three composable networks. Assume also that $\mathcal{M} \,||\, \mathcal{N}$ is defined, and $\mathcal{M} \,||\, \mathcal{N} \overset{\tau}{\longmapsto} \Gamma_L \rhd L$.

Further, by Definition 4.1.3(1) then either $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.\tau} L$ for some node m, or $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.c!v} L$ for some node m, channel c and value v such that $\mathsf{Out}_m(\mathcal{M} \,||\, \mathcal{N}) = \emptyset$. In both cases we have that $\Gamma_L = (\Gamma_M \cup \Gamma_N)$.

- If $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.\tau} L$, then by Lemma 2.4.8 we have that $(M \,|\, N) \equiv m[\![\tau.P + Q]\!] \,|\, L_1$ for some processesP, Q and system term L_1; further $L \equiv m[\![P]\!] \,|\, L_1$. For $m \in \mathrm{nodes}(M \,|\, N)$, either $m \in \mathrm{nodes}(M)$ or $m \in \mathrm{nodes}(N)$.

  If $m \in \mathrm{nodes}(M)$, then we have that $M \equiv m[\![\tau.P + Q]\!] \,|\, M_1$ for some system term M_1. Also, we have that $M \,|\, N \equiv m[\![\tau.P + Q]\!] \,|\, M_1 \,|\, N$ and $M \,|\, N \equiv |[\![\tau.P + Q]\!]L_1$. Thus, it is easy to show that $L_1 \equiv M_1 \,|\, N$, hence $L \equiv m[\![P]\!] \,|\, M_1 \,|\, N$.

  Since $M \equiv |[\![\tau.P + Q]\!]M_1$, by Lemma 2.4.8 there exists M' such that $\Gamma_M \rhd M \xrightarrow{m.\tau} M'$ and $M' \equiv m[\![P]\!] \,|\, M_1$; further, by Definition 4.1.3 (1) we obtain that $\Gamma_M \rhd M \overset{\tau}{\longmapsto} M'$.

  Also, we have that $\Gamma_L = (\Gamma_M \cup \Gamma_N)$, and $L \equiv (m[\![P]\!] \,|\, M_1 \,|\, N) \equiv M' \,|\, N$. That is, $\mathcal{L} \equiv \mathcal{M}' \,||\, \mathcal{N}$, where $\mathcal{M}' = \Gamma_M \rhd M'$.

  The case $m \in \mathrm{nodes}(N)$ is similar to the previous one; in this case we obtain that $\mathcal{N} \overset{\tau}{\longmapsto} \mathcal{N}'$ for some $\mathcal{N}'$ such that $\mathcal{L} \equiv \mathcal{M} \,||\, \mathcal{N}'$.

- Suppose now that $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.c!v} L$ for some node m, channel c and value v such that $\mathsf{Out}_m(\mathcal{M} \,||\, \mathcal{N}) = \emptyset$.

  By Proposition 2.4.7 there exist a closed expression e, processes P, Q and system terms L_1, L_2 such that $[\![e]\!] = v$, $M \,|\, N \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, L_1$, $(\Gamma_M \cup \Gamma_N) \rhd L_1 \xrightarrow{m.c?v} L_2$ and $L \equiv m[\![P]\!] \,|\, L_2$.

  In this case we have that $m \in \mathrm{nodes}(M \,|\, N)$, hence either $m \in \mathrm{nodes}(M)$ or $n \in \mathrm{nodes}(N)$. We only consider the case in which $m \in \mathrm{nodes}(M)$, for the case $n \in \mathrm{nodes}(N)$ is symmetric to the former one.

  Let then $m \in \mathrm{nodes}(M)$. In this case we have that $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, M_1$ for some system term m_1; this leads to $M \,|\, N \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, M_1 \,|\, N$. Since we already known that $M \,|\, N \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, L_1$, and since $M \,|\, N$ is a well-formed system term, it follows that $L_1 \equiv M_1 \,|\, N$.

  Recall that $(\Gamma_M \cup \Gamma_N) \rhd L_1 \xrightarrow{m.c?v} L_2$. Since $L_1 \equiv M_1 \,|\, N$, by Proposition 2.4.6 it follows that $(\Gamma_M \cup \Gamma_N) \rhd (M_1 \,|\, N) \xrightarrow{m.c?v} L_2'$ for some L_2' such that $L_2' \equiv L_2$. By Lemma 2.4.5(i) we obtain that $(\Gamma_M \cup \Gamma_N) \rhd M_1 \xrightarrow{m.c?v} M_1'$, $(\Gamma_M \cup \Gamma_N) \rhd N \xrightarrow{m.c?v} N'$, for some M_1', N' such that $M_1' \,|\, N' = L_2'$. Since $L \equiv m[\![P]\!] \,|\, L_2$, $L_2 \equiv L_2'$ and $L_2' = M_1' \,|\, N'$, by performing the appropriate substitutions it follows that $L \equiv m[\![P]\!] \,|\, M_1' \,|\, N'$

  Let us focus on the transition $(\Gamma_M \cup \Gamma_N) \rhd M_1 \xrightarrow{m.c?v} M_1'$. We show that this implies $\Gamma_M \rhd M \xrightarrow{m.c?v} M_1'$. To this end, we need to show that $\Gamma_M \rhd M_1$ satisfies the hypothesis of Lemma 4.2.9, Strengthening, relative to the connectivity graph $\Gamma_M \cup \Gamma_N$. To this end, we need to show the following:

  - $\mathrm{nodes}(M_1) \subseteq (\Gamma_M)_V$. Recall that $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, M_1$; we obtain that $\mathrm{nodes}(M_1') \subseteq \mathrm{nodes}(M)$. Hence, since $\Gamma_M \rhd M$ is a composable network, we also have that $\mathrm{nodes}(M_1') \subseteq \mathrm{nodes}(M) \subseteq (\Gamma_M)_V$.
  - $\Gamma_M \subseteq (\Gamma_M \cup \Gamma_N)$. This statement is trivial.

– whenever $(\Gamma_M \cup \Gamma_N) \vdash n \to l$, where either $n \in \mathrm{nodes}(M_1)$ or $l \in \mathrm{nodes}(M_1)$, then $\Gamma_M \vdash n \to l$. Without loss of generality, let $n \in \mathrm{nodes}(M_1)$ and $(\Gamma_M \cup \Gamma_N) \vdash n \to l$. Since $\mathrm{nodes}(M_1) \subseteq \mathrm{nodes}(M)$, we also have $n \in \mathrm{nodes}(M)$.

  Since we are assuming that $(\Gamma_M \cup \Gamma_N) \vdash n \to l$, then either $\Gamma_M \vdash n \to l$, in which case there is nothing to prove, or $\Gamma_N \vdash n \to l$. In the latter case, note that $n \in \mathrm{nodes}(M)$, and $P_s(\mathcal{M}, \mathcal{N}) = \mathrm{true}$ by hypothesis. Thus it follows that $\Gamma_M \vdash n \to l$.

We have shown that we can apply Strengthening, Lemma 4.2.9, to the transition $(\Gamma_M \cup \Gamma_N) \rhd M_1 \xrightarrow{m.c?v} M_1'$ to obtain $\Gamma_M \rhd M_1 \xrightarrow{m.c?v}$. In a similar way, it is possible to prove that Strengthening can also be applied to the transition $(\Gamma_M \cup \Gamma_N) \rhd N \xrightarrow{m.c?v} N'$ and the connectivity graph Γ_N, leading to $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$.

Since $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, M_1$, $\Gamma_M \rhd M_1 \xrightarrow{m.c?v} M_1'$, $[\![e]\!] = v$, it is easy to derive the transition $\Gamma_M \rhd m[\![c!\langle e\rangle.P + Q]\!] \,|\, M_1 \xrightarrow{m.c!v} \Gamma_M \rhd m[\![P]\!] \,|\, M_1$; then, since $M \equiv m[\![c!\langle e\rangle.P + Q]\!] \,|\, M_1$, by Proposition 2.4.7 there exists a system term M' such that $\Gamma_M \rhd M \xrightarrow{m.c!v} M'$, and $M' \equiv m[\![P]\!] \,|\, M_1$. Finally, since we have already shown that $L \equiv m[\![P]\!] \,|\, M_1' \,|\, N'$, it is easy to note that $L \equiv M' \,|\, N'$.

So far we have proved the following:

(i) $\Gamma_M \rhd M \xrightarrow{m.c!v} M'$

(ii) $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$

(iii) $L \equiv M' \,|\, N'$

It remains to infer the extensional transitions which are induced by the transitions (i) and (ii) above. To accomplish this task, we need to perform a case analysis on the set $\mathsf{Out}_m(\mathcal{M})$.

– $\mathsf{Out}_m(\mathcal{M}) = \emptyset$. By Definition 4.1.3(1) we have that $\Gamma_M \rhd M \xrightarrow{m.c!v} M'$ implies $\Gamma_M \rhd M \xmapsto{\tau} M'$. Also, by Lemma 4.2.11 it follows that $\mathsf{In}_m(\mathcal{N}) = \emptyset$, so that $m \notin \mathsf{Input}(\mathcal{N})$. A simple structural induction on the proof of the derivation $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$ shows that, in this case, $N' = N$.

  In this case we have proved that, if $(\Gamma_M \rhd M) \parallel (\Gamma_N \rhd N) \xrightarrow{m.c!v} (\Gamma_L \rhd L)$, where $\mathsf{Out}_m(\Gamma_M \rhd M) = \emptyset$, then $\Gamma_M \rhd M \xmapsto{\tau} M'$, and $(\Gamma_L \rhd L) \equiv (\Gamma_M \rhd M') \parallel (\Gamma_N \rhd N)$.

– $\mathsf{Out}_m(\mathcal{M}) \neq \emptyset$. Let then $\eta = \mathsf{Out}_m(\mathcal{M})$. By Definition 4.1.3(3) it follows that $\Gamma_M \rhd M \xmapsto{c!v \rhd \eta} M'$.

  Recall that $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) = \emptyset$ by hypothesis; then it is straightforward to note that $\mathsf{Out}_m(\mathcal{M}) \subseteq \mathrm{nodes}(N)$; in fact, if there had been a node n such that $n \in \mathsf{Out}_m(\mathcal{M})$ and $n \notin \mathrm{nodes}(N)$, then it would have followed that $n \notin \mathrm{nodes}(M), n \notin \mathrm{nodes}(M)$ and $(\Gamma_M \cup \Gamma_N) \vdash m \to n$, leading to $n \in \mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N})$, which causes a contradiction.

  Since $\mathsf{Out}_m(\mathcal{M}) \subseteq \mathrm{nodes}(N)$, by Lemma 4.2.11 it follows that $\mathsf{Out}_m(\mathcal{M}) = \mathsf{In}_m(\mathcal{N})$. Since such sets are non-empty by hypothesis, it follows that $m \in \mathsf{Input}(\mathcal{N})$. For $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$, by Definition (4.1.3)(2) it follows that $\Gamma_N \rhd N \xmapsto{m.c?v} N'$.

  Therefore, in this case we have shown that $\Gamma_M \rhd M \xmapsto{c!v \rhd \eta} M'$ for some set of nodes $\eta \subseteq \mathrm{nodes}(N)$, $\Gamma_N \rhd N \xmapsto{m.c?v} N'$ for some node m such that $\mathsf{In}_m(\Gamma_N \rhd N) = \eta$, and $L \equiv M' \,|\, N'$.

Proof of Proposition 4.2.14 Let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$ and $\mathcal{L} = \Gamma_L \rhd L$ be three composable networks. Assume that $\mathcal{M} \parallel \mathcal{N}$ is defined, and $\mathcal{M} \parallel \mathcal{N} \xmapsto{m.c?v} \mathcal{L}$. Here it is immediate to note that $\Gamma_L = (\Gamma_M \cup \Gamma_N)$.

By Definition 4.1.3(2) it follows that $\mathcal{M} \parallel \mathcal{N} \xrightarrow{m.c?v} \mathcal{L}$, where $m \in \mathsf{Input}(\mathcal{M} \parallel \mathcal{N})$. That is, $(\Gamma_M \cup \Gamma_N) \vdash m \to n$ for some node $n \in \mathrm{nodes}(M \,|\, N)$.

By Lemma 2.4.5(i) it follows that there exist two system terms M', N' such that $(\Gamma_M \cup \Gamma_N) \rhd M \xrightarrow{m.c?v} M'$ and $(\Gamma_M \cup \Gamma_N) \rhd N \xrightarrow{m.c?v} N'$, with $L = M' \,|\, N'$. Further, it is not difficult that the conditions required to apply Strengthening, Lemma 4.2.9 to these transitions (relative to the connectivity graphs Γ_M, Γ_N, respectively) are met; therefore it follows that $\Gamma_M \rhd M \xrightarrow{m.c?v} M'$, $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$.

Note that, since $m \in \mathsf{Input}((\Gamma_M \rhd M) \parallel (\Gamma_N \rhd N))$, then there exists a node $n \in \mathrm{nodes}(M \,|\, N)$ such that either $\Gamma_M \vdash m \to n$ or $\Gamma_N \vdash m \to n$.

Note that we have $m \notin \text{nodes}(M), m \notin \text{nodes}(N)$; therefore, if $\Gamma_M \vdash m \to n$, then $n \in \text{nodes}(M)$, from which it follows by definition that $m \in \mathsf{Input}(\Gamma_M \rhd M)$. This is because $Gamma_M \rhd M$ is a composable network by hypothesis. Similarly, if $\Gamma_N \vdash m \to n$, we have that $n \in \text{nodes}(N)$, hence $n \in \mathsf{Input}(\Gamma_N \rhd N)$.

We have proved that, whenever $m \in \mathsf{Input}((\Gamma_M \rhd M) \,\|\, (\Gamma_N \rhd N))$ then either $m \in \mathsf{Input}(\Gamma_M \rhd M)$ or $m \in \mathsf{Input}(\Gamma_N \rhd N)$. Therefore, for node m we have three possible cases:

1. $m \in \mathsf{Input}(\Gamma_M \rhd M)$, but $n \notin \mathsf{Input}(\Gamma_N \rhd N)$. In this case it is possible to prove that $N = N'$, by performing a simple structural induction on the proof of the transition $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$.

   Further, since $\Gamma_M \rhd M \xrightarrow{m.c?v} M'$, and $m' \in \mathsf{Input}(\Gamma_M \rhd M)$, it follows from Definition 4.1.3(2) that $\Gamma_M \rhd M \overset{m.c?v}{\longmapsto} M'$.

   Let $\mathcal{M}' = \Gamma_M \rhd M'$. We have shown that, if $m \in \mathsf{Input}(\mathcal{M})$ and $n \in \mathsf{Input}(\mathcal{N})$, then $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{M}'$, and $\mathcal{L} = (\Gamma_M \cup \Gamma_N) \rhd M' \,|\, N$, the latter network being exactly $\mathcal{M}' \,\|\, \mathcal{N}$,

2. or $m \in \mathsf{Input}(\Gamma_N \rhd N)$, but $n \notin \mathsf{Input}(\Gamma_M \rhd M)$. This case is symmetric to the previous one. If we let $\mathcal{N}' = \Gamma_N \rhd N'$, it follows that $\mathcal{N} \overset{m.c?v}{\longmapsto} \mathcal{N}'$, and $\mathcal{L} = \mathcal{M} \,\|\, \mathcal{N}'$,

3. or $m \in \mathsf{Input}(\Gamma_M \rhd M), m \in \mathsf{Input}(\Gamma_N \rhd N)$. Since $\Gamma_M \rhd M \xrightarrow{m.c?v} M'$, $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$, it follows from Definition 4.1.3(2) that $\Gamma_M \rhd M \overset{m.c?v}{\longmapsto} M'$ and $\Gamma_N \rhd N \overset{m.c?v}{\longmapsto} N'$.

   Let $\mathcal{M}' = \Gamma_M \rhd M'$, $\mathcal{N}' = \Gamma_N \rhd N'$. We have shown that $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{M}'$, $\mathcal{N} \overset{m.c?v}{\longmapsto} \mathcal{N}'$, and $\Gamma_L \rhd L = (\Gamma_M \cup \Gamma_N) \rhd M' \,|\, N'$, the latter network being exactly $\mathcal{M}' \,\|\, \mathcal{N}'$.

Proof of Lemma 4.2.15 The proof is performed by Rule Induction on the proof of the derivation $\Gamma \rhd M \xrightarrow{\lambda} M'$. We only show some of the most interesting cases.

Suppose that the last Rule applied in the proof of $\Gamma \rhd M \xrightarrow{\lambda} M'$ is Rule (B-REC). Then $M = m[\![c?(x).P]\!]$ for some process P, channel c and node m, while $\lambda = n.c?v$ for some value v and node n such that $\Gamma_M \vdash n \to m$. Further, $M' = m[\![\{v/x\}P]\!]$. Now it is easy to show that $(\Gamma_M \cup \Gamma_N) \vdash n \to m$, so that we can apply Rule (B-REC) to infer $(\Gamma_M \cup \Gamma_N) \rhd m[\![c?(x).P]\!] \xrightarrow{n.c?v} m[\![\{v/x\}P]\!]$.

Now assume that the last Rule applied to derive the transition $\Gamma \rhd M \xrightarrow{\lambda} M'$ is Rule (B-DISC). Then $M = M' = m[\![P]\!]$ for some process P and node m, while $\lambda = n.c?v$ for some node n, channel c and value v such that $n \neq m$, $\Gamma_M \vdash n \nrightarrow m$. By assumption, we also have that $\Gamma_N \vdash n \nrightarrow m$. Thus, $(\Gamma_M \cup \Gamma_N) \vdash n \nrightarrow m$. Now we can apply Rule (B-DISC) to infer $(\Gamma_M \cup \Gamma_N) \rhd m[\![P]\!] \xrightarrow{n.c?v} m[\![P]\!]$.

The last case we consider is that in which Rule (B-SYNC − L) has been applied as the last rule in the proof of $\Gamma_M \rhd M \xrightarrow{\lambda} M'$. In this case we have that $\lambda = m.c!v$ for some node m, channel c and node v. Further, we have that $M = M_1 \,|\, M_2$ for some M_1, M_2, M_1' and M_2' such that $\Gamma_M \rhd M_1 \xrightarrow{m.c!v} M_1'$, $\Gamma_M \rhd M_2 \xrightarrow{m.c?v} M_2'$. By inductive hypothesis, we have that $(\Gamma_M \cup \Gamma_N) \rhd M_1 \xrightarrow{m.c!v} M_1'$, and $(\Gamma_M \cup \Gamma_N) \rhd M_2 \xrightarrow{m.c?v} M_2'$, where $M' = M_1' \,|\, M_2'$. Finally, we can apply Rule (B-SYNC − L) to derive $(\Gamma_M \cup \Gamma_N) \rhd M_1 \,|\, M_2 \xrightarrow{m.c!v} M_1' \,|\, M_2'$.

Proof of Proposition 4.2.18 We only prove statements (i) and (iii), for the proofs of statements (ii) and (iv) can be obtained symmetrically. To this end, let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$ be two composable networks, and suppose that $\mathcal{M} \,\|\, \mathcal{N}$ is defined.

First we prove that, if $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{M}'$, then $\mathcal{M} \,\|\, \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \,\|\, \mathcal{N}$. Suppose then $\Gamma_M \rhd M \overset{\tau}{\longmapsto} M'$, where $\mathcal{M}' = \Gamma_M \rhd M'$. By Definition 4.1.3(1) there are two possible cases.

(a) $\Gamma_M \rhd M \xrightarrow{m.\tau} M'$ for some node $m \in \text{nodes}(M)$. Note that, since $\mathcal{M} \,\|\, \mathcal{N}$ is defined, then $P_s(\mathcal{M}, \mathcal{N}) = \text{true}$. Hence we have that, whenever $\Gamma_N \vdash n \to l$, where either $n \in \text{nodes}(M)$ or $l \in \text{nodes}(M)$, then $\Gamma_M \vdash n \to l$. Therefore, we can apply Weakening, Lemma 4.2.15 to the transition $\Gamma_M \rhd M \xrightarrow{m.\tau} M'$ and the connectivity graph Γ_N, leading to $(\Gamma_M \cup \Gamma_N) \rhd M \xrightarrow{m.\tau} M'$.

   Now it is sufficient to apply rule (B-τ − PROP − L) to the transition above to infer $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.\tau} M'$. By Definition 4.1.3(1) it follows that $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \overset{\tau}{\longmapsto} (M' \,|\, N)$.

For $\mathcal{M}' = \Gamma_M \rhd M'$, we have that $(\Gamma_M \cup \Gamma_N) \rhd (M \mid N) = \mathcal{M} \parallel \mathcal{N}$, while $(\Gamma_M \cup \Gamma_N) \rhd (M' \mid N) = \mathcal{M}' \parallel \mathcal{N}$. Therefore we have $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}$.

(b) $\Gamma_M \rhd M \overset{m.c!v}{\longrightarrow} M'$ for some channel c, value v and node m such that $\mathsf{Out}_m(\mathcal{M}) = \emptyset$. Note that, by Lemma 4.2.16 this leads to $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) = \emptyset$.

In a way similar to the case above we can show that Weakening can be applied to the transition $\Gamma_M \rhd M \overset{m.c!v}{\longrightarrow} M'$ and the connectivity graph Γ_N, leading to the transition $(\Gamma_M \cup \Gamma_N) \rhd M \overset{m.c!v}{\longrightarrow} M'$. Further, since $\mathsf{Out}_m(\mathcal{M}) = \emptyset$ and $\mathcal{M} \parallel \mathcal{N}$ is defined, it follows from Lemma 4.2.11 that $\mathsf{In}_m(\mathcal{N}) = \emptyset$. In this case, it is possible to show that $\Gamma_N \rhd N \overset{m.c?v}{\longrightarrow} N$. It is easy to show that we can apply Weakening to the last transition and the connectivity graph Γ_M, leading to the transition $(\Gamma_M \cup \Gamma_N) \rhd N \overset{m.c?v}{\longrightarrow} N$.

We have shown that $(\Gamma_M \cup \Gamma_N) M \overset{m.c!v}{\longrightarrow} M'$, and $(\Gamma_M \cup \Gamma_N) \rhd N \overset{m.c?v}{\longrightarrow} N$. Then it is easy to derive $(\Gamma_M \cup \Gamma_N) \rhd (M \mid N) \overset{m.c!v}{\longrightarrow} (M' \mid N)$.

Since $\mathcal{M}' = \Gamma_M \rhd M'$, we obtain that $(\Gamma_M \cup \Gamma_N) \rhd (M' \mid N) = \mathcal{M}' \parallel \mathcal{N}$. Since $\mathcal{M} \parallel \mathcal{N} = (\Gamma_M \cup \Gamma_N) \rhd (M \mid N)$, the Transition $(\Gamma_M \cup \Gamma_N) \rhd (M \mid N) \overset{m.c!v}{\longrightarrow} M' \mid N$ can be rewritten as $\mathcal{M} \parallel \mathcal{N} \overset{m.c!v}{\longrightarrow} \mathcal{M}' \parallel \mathcal{N}$.

Now recall that $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) = \emptyset$, so that by Definition 4.1.3(1) we obtain then $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}$, as we wanted to prove.

Now suppose that $\mathcal{M} \overset{c!v \rhd \eta}{\longmapsto} \mathcal{M}'$ for some channel c, value v, network $\mathcal{M}'$ and set of nodes η for which $\eta \subseteq \mathrm{nodes}(N)$ holds.

By Definition 4.1.3(3) we have that $\Gamma_M \rhd M \overset{m.c!v}{\longrightarrow} M'$, where $\mathcal{M}' = \Gamma_M \rhd M'$, for some node m such that $\mathsf{Out}_m(\mathcal{M}) = \eta$. Since $\eta \subseteq \mathrm{nodes}(N)$, and $\mathcal{M} \parallel \mathcal{N}$ is defined by hypothesis, Lemma 4.2.11 ensures that $\mathsf{In}_m(\mathcal{N}) = \mathsf{Out}_m(\mathcal{M}) \cap \mathrm{nodes}(N) = \eta \cap \mathrm{nodes}(N) = \eta$.

It remains to prove that, if $\mathcal{N} \overset{m.c?v}{\longmapsto} \mathcal{N}'$, then $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}'$. Suppose then that $\Gamma_N \rhd N \overset{m.c?v}{\longrightarrow} N'$ for some system term N'. Here $\mathcal{N}' = \Gamma_N \rhd N'$.

It is easy to show that we can apply Weakening, Lemma 4.2.15, to the transitions $\Gamma_M \rhd M \overset{m.c!v}{\longrightarrow} M'$ (relative to the connectivity graph Γ_N) and $\Gamma_N \rhd N \overset{m.c?v}{\longrightarrow} N'$ (relative to the connectivity graph Γ_M). This leads to the transitions $(\Gamma_M \cup \Gamma_N) \rhd M \overset{m.c!v}{\longrightarrow} M'$ and $(\Gamma_M \cup \Gamma_N) \rhd N \overset{m.c?v}{\longrightarrow} N'$, from which $(\Gamma_M \cup \Gamma_M) \rhd (M \mid N) \overset{m.c!v}{\longrightarrow} (M' \mid N')$ can be easily derived through an application of Rule (B-SYNC – R).

Recall that $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{M}' = \Gamma_M \rhd M'$, $\mathcal{N} = \Gamma_N \rhd N$ and $\mathcal{N}' = \Gamma_M \rhd M'$; for we have that $P_s(\mathcal{M}, \mathcal{N}) = \mathrm{true}$ (recall that $\mathcal{M} \parallel \mathcal{N}$ is defined by hypothesis), it is trivial to show that $P_s(\mathcal{M}', \mathcal{N}') = \mathrm{true}$, hence $\mathcal{M}' \parallel \mathcal{N}'$ is defined. Thus the last transition can be rewritten as $\mathcal{M} \parallel \mathcal{N} \overset{m.c!v}{\longrightarrow} \mathcal{M}' \parallel \mathcal{N}'$.

It remains to note that, since $\mathsf{Out}_m(\mathcal{M}) \subseteq \mathrm{nodes}(\mathcal{N})$, then $\mathsf{Out}_m(\mathcal{M} \parallel \mathcal{N}) = \emptyset$. This is an immediate consequence of Corollary 4.2.17. By Definition 4.1.3(1) we obtain that the transition $\mathcal{M} \parallel \mathcal{N} \overset{m.c!v}{\longrightarrow} \mathcal{M}' \parallel \mathcal{N}'$ induces the extensional one $\mathcal{M} \parallel \mathcal{N} \overset{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{N}'$, as we wanted to prove.

Proof of Proposition 4.2.20 Let $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{N} = \Gamma_N \rhd N$, $\mathcal{M}' = \Gamma_M \rhd M'$ and $\mathcal{N}' = \Gamma_N \rhd N'$ be composable network. Suppose that $\mathcal{M} \parallel \mathcal{N}$ is defined.

We prove each of the statements in the Proposition individually.

(i) Suppose $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{M}'$ for some node m, channel c and value v such that $\Gamma_N \not\vdash m$.

Then we have that $\Gamma_M \rhd M \overset{m.c?v}{\longrightarrow} M'$ and $\Gamma_M \vdash m \to n$ for some node $n \in \mathrm{nodes}(M)$; further, it is easy to prove that $\Gamma_N \rhd N \overset{m.c?v}{\longrightarrow} N$. Now note that we can apply Weakening to these two transitions, relative to the connectivity graphs Γ_N and Γ_M, respectively.

By Definition 4.1.3(2) we have that $(\Gamma_M \cup \Gamma_N) \rhd M \overset{m.c?v}{\longrightarrow} M'$, $(\Gamma_M \cup \Gamma_N) \rhd N \overset{m.c?v}{\longrightarrow} N$, from which it follows $(\Gamma_M \cup \Gamma_N) \rhd (M \mid N) \overset{m.c?v}{\longrightarrow} (M' \mid N')$ by a simple application of Rule (B-PROP), defined in Figure 2.4.

For $\Gamma_M \vdash m \to n$, we have that $(\Gamma_M \cup \Gamma_N) \vdash m \to n$. Also, $n \in \mathrm{nodes}(M)$, from which it follows that $m \in \mathrm{nodes}(M \mid N)$. Finally, since $m \in \mathsf{Input}(\Gamma_M \rhd M)$, $m \notin \mathrm{nodes}(M)$. Since $\Gamma_N \not\vdash m$, we also have that $m \notin \mathrm{nodes}(M)$. Thus $(\Gamma_M \cup \Gamma_N) \vdash m \to n$, but $m \notin \mathrm{nodes}(M \mid N)$. It follows that $m \in \mathsf{Input}((\Gamma_M \cup \Gamma_N) \rhd (M \mid N))$.

We have proved that $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.c?v} (M' \,|\, N')$, and $m \in \mathsf{Input}((\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N))$. Therefore, by Definition 4.1.3(2) it follows that $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xmapsto{m.c?} (M' \,|\, N)$. Since $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) = \mathcal{M} \,\|\, \mathcal{N}$, and $(\Gamma_M \cup \Gamma_N) \rhd (M' \,|\, N) = \mathcal{M}' \,\|\, \mathcal{N}$, the last transition corresponds to $\mathcal{M} \,\|\, \mathcal{N} \xmapsto{m.c?v} \mathcal{M}' \,|\, \mathcal{N}$.

(ii) We have to show that if $\mathcal{N} \xmapsto{m.c?v} \mathcal{N}'$, and $\Gamma_M \not\vdash m$, then $\mathcal{M} \,\|\, \mathcal{N} \xmapsto{m.c?v} \mathcal{M} \,\|\, \mathcal{N}'$. The proof for this statement is symmetric to the previous one, and it is therefore omitted.

(iii) Suppose that $\mathcal{M} \xmapsto{m.c?v} \mathcal{M}'$, $\mathcal{N} \xmapsto{m.c?v} \mathcal{N}'$. By Definition 4.1.3(2) we have that $\Gamma_M \rhd M \xrightarrow{m.c?v} M'$, and $\Gamma_N \rhd N \xrightarrow{m.c?v} N'$. Further, $m \in \mathsf{Input}(\Gamma_M)$ and $n \in \mathsf{Input}(\Gamma_N)$.

Note that we can apply Weakening, Lemma 4.2.15 to these two transitions, leading to the transitions $(\Gamma_M \cup \Gamma_N) \rhd M \xrightarrow{m.c?v} M'$ and $(\Gamma_M \cup \Gamma_N) \rhd N \xrightarrow{m.c?v} N'$. A simple application of Rule (B-PROP) can be used to infer $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.c?v} (M' \,|\, N')$.

Therefore we have that $m \notin \mathrm{nodes}(M), m \notin \mathrm{nodes}(N)$, or equivalently $m \notin \mathrm{nodes}(M \,|\, N)$. Since, $m \in \mathsf{Input}(\Gamma_M \rhd M)$, $\Gamma_M \vdash m \to n$ for some node $n \in \mathrm{nodes}(M)$, hence $(\Gamma_M \cup \Gamma_N) \vdash m \to n$. Now it is straightforward to show that $m \in \mathsf{Input}((\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N))$.

Since $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xrightarrow{m.c?v} (M' \,|\, N')$ and $m \in \mathsf{Input}((\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N))$, Definition 4.1.3(2) ensures that $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) \xmapsto{m.c?v} (M' \,|\, N')$. For $(\Gamma_M \cup \Gamma_N) \rhd (M \,|\, N) = \mathcal{M} \,\|\, \mathcal{N}$, and $(\Gamma_M \cup \Gamma_N) \rhd (M' \,|\, N') = \mathcal{M}' \,\|\, \mathcal{N}'$, the last transition can be rewritten as $\mathcal{M} \,\|\, \mathcal{N} \xmapsto{m.c?v} \mathcal{M}' \,\|\, \mathcal{N}'$.

Proof of Proposition 4.2.23 Each of this statement is proved individually. We only show the proof for the most interesting of them.

In the following, we assume that $\mathcal{M} = \Gamma_M \rhd M$, $\mathcal{M}' = \Gamma_M \rhd M'$ are composable networks composable network, while $\mathcal{G} = \Gamma_G \rhd n[\![P]\!]$ and $\mathcal{G}' = \Gamma_G \rhd n[\![Q]\!]$ are generating networks. Further, we suppose that $\mathcal{M} \,\|\, \mathcal{G}$ is defined, from which it follows that $\mathcal{M}' \,\|\, \mathcal{G}$, $\mathcal{M} \,\|\, \mathcal{G}'$ and $\mathcal{M}' \,\|\, \mathcal{G}'$ are defined as well.

(a) **P**roof of Proposition 4.2.23(i).

Suppose that $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}'$, and $\mathcal{G} \overset{\tau}{\longmapsto} \mathcal{G}'$. By Lemma 4.2.21 we obtain that $\mathcal{M} \,\|\, \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \,\|\, \mathcal{G}$, while by Proposition 4.2.18(ii) it follows that $\mathcal{M}' \,\|\, \mathcal{G} \overset{\tau}{\longmapsto} \mathcal{M}' \,\|\, \mathcal{G}'$.

Therefore we have that

$$\mathcal{M} \,\|\, \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \,\|\, \mathcal{G} \overset{\tau}{\longmapsto} \mathcal{M}' \,\|\, \mathcal{G}'$$

from which $\mathcal{M} \,\|\, \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M} \,\|\, \mathcal{G}'$ follows.

(b) **P**roof of Proposition 4.2.23(ii).

Suppose $\mathcal{M} \overset{c!v \rhd \{n\}}{\Longmapsto} \mathcal{M}'$, and $\mathcal{G} \xmapsto{m.c?v} \mathcal{G}'$ for some node $m \in \mathsf{Input}(\mathcal{G})$. Note that, by Definition 4.1.5 (3b) either $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}_1 \xmapsto{c!v \rhd \{n\}} \mathcal{M}_2 \overset{\tau}{\Longmapsto} \mathcal{M}'$ for some networks $\mathcal{M}_1, \mathcal{M}_2$, or $\mathcal{M} \overset{c!v \rhd \eta_1}{\Longmapsto} \mathcal{M}'' \overset{c!v \rhd \eta_2}{\Longmapsto} \mathcal{M}'$ for some network $\mathcal{M}''$ and non-empty sets of nodes η_1, η_2 such that $\eta_1 \cup \eta_2 = \{n\}$, $\eta_1 \cap \eta_2 = \emptyset$.

It is straightforward to note that the last case cannot happen, for η_1 and η_2 should be disjoint sets containing at least an element; thus, for their union we obtain $|\eta_1 \cup \eta_2| \geq 2$, where $|\{n\}| = 1$. Therefore, the only possibility is that $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}_1 \xmapsto{c!v \rhd \{n\}} \mathcal{M}_2 \overset{\tau}{\Longmapsto} \mathcal{M}'$.

Lemma 4.2.21 ensures that $\mathcal{M} \,\|\, \mathcal{G} \overset{\mathcal{M}}{\Longmapsto}_1 \| \, \mathcal{G}$, and $\mathcal{M}_2 \,\|\, \mathcal{G}' \overset{\tau}{\Longmapsto} \mathcal{M}' \,\|\, \mathcal{G}'$.

Now note that, by Lemma 4.2.22, if $\mathcal{G} \xmapsto{m.c?v} \mathcal{G}'$, then for any node $m' \in \mathsf{Input}(\mathcal{G})$ it holds that $\mathcal{G} \xmapsto{m'.c?v} \mathcal{G}$.

Moreover, by Proposition 4.2.18(iii) there exists a node $m' \in \mathsf{Input}(\mathcal{G})$. For in this case we have that $\mathcal{G} \xmapsto{m'.c?v} \mathcal{G}'$, and $\mathcal{M}_1 \xmapsto{c!v \rhd \{n\}} \mathcal{M}_2$, it follows from Proposition 4.2.18(iii) that $\mathcal{M}_1 \,\|\, \mathcal{G} \overset{\tau}{\longmapsto} \mathcal{M}_2 \,\|\, \mathcal{G}'$.

We have proved that

$$\mathcal{M} \,\|\, \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}_1 \,\|\, \mathcal{G}\mathcal{M}_2 \,\|\, \mathcal{G}' \overset{\tau}{\Longmapsto} \mathcal{M}' \,\|\, \mathcal{G}'$$

hence, by definition, $\mathcal{M} \,\|\, \mathcal{G} \overset{\tau}{\Longmapsto} \mathcal{M}' \,\|\, \mathcal{G}'$.

(c) **P**roof of Proposition 4.2.23(iv)

Suppose that $\mathcal{M} \stackrel{c!v\triangleright\eta}{\Longmapsto} \mathcal{M}'$ for some channel c, value v and non-empty set of nodes η. We proceed by induction on the proof of the derivation above (see Remark 4.1.7).

Base case $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{M}_1 \stackrel{c!v\triangleright\eta}{\longmapsto} \mathcal{M}_2 \stackrel{\tau}{\Longmapsto} \mathcal{M}'$ for some networks $\mathcal{M}_1, \mathcal{M}_2$. This case is trivial; by Lemma 4.2.21 we have that $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longmapsto} \mathcal{M}_1 \parallel \mathcal{G}$, and $\mathcal{M}_2 \parallel \mathcal{G} \stackrel{\tau}{\Longmapsto} \mathcal{M}_2 \parallel \mathcal{G}$.
Further, by Proposition 4.2.19(i) it follows that $\mathcal{M}_1 \parallel \mathcal{G} \stackrel{c!v\triangleright\eta}{\longmapsto} \mathcal{M}_2 \parallel \mathcal{G}$; hence we have the sequence of extensional transitions

$$\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longmapsto} \mathcal{M}_1 \parallel \mathcal{G} \stackrel{c!v\triangleright\eta}{\longmapsto} \mathcal{M}_2 \parallel \mathcal{G} \stackrel{\tau}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}$$

Inductive Step There exist a network $\mathcal{M}''$ and two sets of nodes η_1, η_2 such that $\eta_1 \cap \eta_2 = \emptyset$, $\eta_1 \cup \eta_2 = \eta$, and $\mathcal{M} \stackrel{c!v\triangleright\eta_1}{\Longmapsto} \mathcal{M}''$ and $\mathcal{M}'' \stackrel{c!v\triangleright\eta_2}{\Longmapsto} \mathcal{M}'$.
Since $n \notin \eta$, it follows that $n \notin \eta_1, n \notin \eta_2$. It follows from the inductive hypothesis that $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v\triangleright\eta_1}{\Longmapsto} \mathcal{M}'' \parallel \mathcal{G}$ and $\mathcal{M}'' \parallel \mathcal{G} \stackrel{c!v\triangleright\eta_2}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}$.
By definition 4.1.5(3b) it follows that $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v\triangleright\eta}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}$.

(d) **P**roof of Proposition 4.2.23(vi)

Suppose that $\mathcal{M} \stackrel{c!v\triangleright\eta}{\Longmapsto} \mathcal{M}'$ for some set of nodes η such that $n \in \eta$. Also, suppose that $\mathcal{G} \stackrel{m.c?v}{\longmapsto} \mathcal{G}'$ for some node m. In this case note that $m \in \mathsf{Input}(\mathcal{G})$. We proceed by induction on the proof of the derivation $\mathcal{M} \stackrel{c!v\triangleright\eta}{\Longmapsto} \mathcal{M}'$.

Base Case $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{M}_1 \stackrel{c!v\triangleright\eta}{\longmapsto} \mathcal{M}_2 \stackrel{\tau}{\Longmapsto} \mathcal{M}'$ for some networks $\mathcal{M}_1, \mathcal{M}_2$.
It follows from Lemma 4.2.21 that $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longmapsto} \mathcal{M}_1 \parallel \mathcal{G}$, and $\mathcal{M}_2 \parallel \mathcal{G}' \stackrel{\tau}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}'$. Note that, since $m \in \eta$, we have that $\eta = (\eta \setminus \{n\} \cup \{n\})$. By Proposition 4.2.19(iii) there exists a node $m' \in \mathsf{Input}(\mathcal{G})$ such that, if $\mathcal{G} \stackrel{m'.c?v}{\longmapsto} \mathcal{G}''$ for some network $\mathcal{G}'$, then $\mathcal{M}_1 \parallel \mathcal{G} \stackrel{c!v\triangleright(\eta\setminus\{n\})}{\longmapsto} \mathcal{M}_2 \parallel \mathcal{G}'$. Further, for $\mathcal{M} \stackrel{m.c?v}{\longmapsto} \mathcal{M}'$, then by Lemma 4.2.22 we have that $\mathcal{G} \stackrel{m'.c?v}{\longmapsto} \mathcal{G}'$, hence $\mathcal{M}_1 \parallel \mathcal{G} \stackrel{c!v\triangleright(\eta\setminus\{n\})}{\Longmapsto} \mathcal{M}_2 \parallel \mathcal{G}'$.
Thus, we have shown that

$$\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longmapsto} \mathcal{M}_1 \parallel \mathcal{G} \stackrel{c!v\triangleright(\eta\setminus\{n\})}{\longmapsto} \mathcal{M}_2 \parallel \mathcal{G}' \stackrel{\tau}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}'$$

from which we obtain the weak extensional transition $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v\triangleright(\eta\setminus\{n\})}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}'$.

Inductive Step $\mathcal{M} \stackrel{c!v\triangleright\eta_1}{\Longmapsto} \mathcal{M}'' \stackrel{c!v\triangleright\eta_2}{\Longmapsto} \mathcal{M}'$ for some network $\mathcal{M}''$, non-empty sets of nodes $\eta_1 \cup \eta_2$ such that $\eta_1 \cap \eta_2 = \emptyset$, $\eta_1 \cup \eta_2 = \eta$. Since $m \in \eta$, and $\eta_1 \cap \eta_2 = \emptyset$, we obtain that either $n \in \eta_1$ or $n \in \eta_2$. Further, these two statements are mutually exclusive.
We only consider the case where $n \in \eta_1$, for the other case can be proved similarly. If $m \in \eta_1$, by inductive hypothesis we have that $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v\triangleright(\eta_1\setminus\{n\})}{\Longmapsto} \mathcal{M}'' \parallel \mathcal{G}$. Moreover, since $m \notin \eta_2$, Proposition 4.2.23 (iv) ensures that $\mathcal{M}'' \parallel \mathcal{G}' \stackrel{c!v\triangleright\eta_2}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}'$. Now we can combine these two transitions to obtain $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v\triangleright(\eta_1\setminus\{n\})\cup\eta_2}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}'$.
Since $n \notin \eta_2$, we also have that $(\eta_1 \setminus \{n\}) \cup \eta_2 = (\eta_1 \cup \eta_2) \setminus \{n\} = \eta \setminus \{n\}$; therefore $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v\triangleright(\eta\setminus\{n\})}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}$, as we wanted to prove.

A.4 Proofs of the Results Concerning the May-testing Preorder

Proof of Lemma 4.3.5 Let $\mathcal{M}, \mathcal{N}$ be composable networks, and let t be a trace. Suppose that $\mathcal{M} \stackrel{\tau}{\Longmapsto} \mathcal{N}$ and $t \in \mathrm{traces}_s(\mathcal{N})$.

By Definition 4.1.5(1) it holds that

$$\mathcal{M}_0 \overset{\tau}{\longmapsto} \mathcal{M}_1 \overset{\tau}{\longmapsto} \cdots \overset{\tau}{\longmapsto} \mathcal{M}_k$$

where $\mathcal{M}_0 = \mathcal{M}, \mathcal{M}_k = \mathcal{N}$ for some $k \geq 0$. We prove that $t \in \text{traces}_s(\mathcal{M})$ by performing a natural induction over k.

$k = 0$ This case is trivial, for $\mathcal{M} = \mathcal{M}_0 = \mathcal{M}_k = \mathcal{N}$, and $t \in \text{traces}_s(\mathcal{N})$ by hypothesis.

$k > 0$ Suppose that the statement is true for $k-1$. For $\mathcal{M}_{k-1} \overset{\tau}{\longmapsto} \mathcal{M}_k$, and $t \in \text{traces}_s(\mathcal{M}_k)$ by hypothesis (recall that $\mathcal{M}_k = \mathcal{N}$), Definition 4.3.4 ensures that $t \in \mathcal{M}_{k-1}$. Further, we have the sequence of transitions

$$\mathcal{M}_0 \overset{\tau}{\longmapsto} \mathcal{M}_1 \overset{\tau}{\longmapsto} \cdots \overset{\tau}{\longmapsto} \mathcal{M}_{k-1}$$

Since $t \in \text{traces}_s(\mathcal{M}_{k-1})$, it follows by inductive hypothesis that $t \in \text{traces}_s(\mathcal{M}_0)$. Now $\mathcal{M}_0 = \mathcal{M}$, hence $t \in \text{traces}_s(\mathcal{M})$.

Proof of Proposition 4.3.6 Let $\mathcal{M}$ be a composable network. We first show that $\text{traces}_s(\mathcal{M}) \subseteq \text{traces}(\mathcal{M})$, then we prove that $\text{traces}(\mathcal{M}) \subseteq \text{traces}_s(\mathcal{M})$.

For the first inclusion, it is sufficient to show that the set $\text{traces}(\mathcal{M})$ satisfies the requirements of Definition 4.3.4. In fact, for $\text{traces}_s(\mathcal{M})$ is the smallest set that satisfies such constraints, it follows that $\text{traces}_s(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$.

We need to show the following:

(i) $\varepsilon \in \text{traces}(\mathcal{M})$. This statement follows directly from Definition 4.3.1(i).

(ii) If $\mathcal{M}$ is a successful configuration, then $\omega \in \text{traces}(\mathcal{M})$.

(iii) Suppose that $\mathcal{M}$ is a successful configuration. Trivially $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}$, holds. Then, by Definition 4.3.1(ii) it holds that $\omega \in \text{traces}(\mathcal{M})$.

(iv) Suppose that $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{N}$, and let t be a trace such that $t \in \text{traces}(\mathcal{N})$. Then $t \in \text{traces}(\mathcal{M})$.

Let t be a trace in $\text{traces}(\mathcal{N})$. We show that $t \in \text{traces}(\mathcal{M})$ by performing an induction on the structure of t.

- $t = \varepsilon$. Then it is trivial to prove that $\varepsilon \in \text{traces}(\mathcal{M})$
- $t = \omega$. Then $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{N}'$ for some successful configuration $\mathcal{N}'$. Since $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{N}$, we also have $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{N}'$, hence $\omega \in \text{traces}(\mathcal{M})$ by Definition 4.3.1(ii).
- $t = m.c?v :: t'$. Then $\mathcal{N} \overset{m.c?v}{\Longmapsto} \mathcal{N}'$ for some network $\mathcal{N}'$ such that $t' \in \text{traces}(\mathcal{N}')$. For $\mathcal{M} \overset{\tau}{\longmapsto} \mathcal{N}$, it is not difficult to show that $\mathcal{M} \overset{m.c?v}{\Longmapsto} \mathcal{N}'$, hence $m.c?v :: t' \in \text{traces}(\mathcal{M})$.
- $t = c!v \triangleright \eta :: t'$. This case is similar to the previous ones.

(v) Suppose that $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{N}$, and let t be a trace such that $t \in \text{traces}(\mathcal{N})$. Then $t \in \text{traces}(\mathcal{M})$.

For $\mathcal{M} \overset{m.c?v}{\longmapsto} \mathcal{N}$, we also have $\mathcal{M} \overset{m.c?v}{\Longmapsto} \mathcal{N}$; hence, by Definition 4.3.1(iii) it follows that $m.c?v :: t \in \text{traces}(\mathcal{M})$.

(vi) If $\mathcal{M} \overset{c!v \triangleright \eta}{\longmapsto} \mathcal{N}$, and t is a trace in $\text{traces}(\mathcal{N})$, then $c!v \triangleright \eta :: t \in \text{traces}(\mathcal{M})$.

This case is analogous to the previous one, this time using Definition 4.3.1(iv)

(vii) If $\mathcal{M} \overset{c!v \triangleright \eta_1}{\longmapsto} \mathcal{N}$, and $c!v \triangleright \eta_2 :: t \in \text{traces}(\mathcal{N})$ for some trace t and set of nodes η_2 such that $\eta_1 \cap \eta_2 = \emptyset$, then $c!v \triangleright (\eta_1 \cup \eta_2) :: t \in \text{traces}(\mathcal{M})$.

By Definition 4.3.1(iv) it follows that there exists a network $\mathcal{N}'$ such that $\mathcal{N} \overset{c!v \triangleright \eta_2}{\Longmapsto} \mathcal{N}'$, and $t \in \text{traces}(\mathcal{N}')$. Since $\mathcal{M} \overset{c!v \triangleright \eta_1}{\longmapsto} \mathcal{N}$, it also holds that $\mathcal{M} \overset{c!v \triangleright \eta_1}{\Longmapsto} \mathcal{N}$. Now we have that $\mathcal{M} \overset{c!v \triangleright \eta_1}{\Longmapsto}$

$\mathcal{N} \stackrel{c!v\triangleright\eta_2}{\models\!\!\Longrightarrow} \mathcal{N}'$; for $\eta_1 \cap \eta_2 = \emptyset$ by hypothesis, Definition 4.1.5(3b) leads to $\mathcal{M} \stackrel{c!v\triangleright(\eta_1\cup\eta_2)}{\models\!\!\Longrightarrow} \mathcal{N}'$. Since $t \in \text{traces}(\mathcal{N}')$, by Definition 4.3.1 (iv) it follows that $c!v \triangleright (\eta_1 \cup \eta_2) :: t \in \text{traces}(\mathcal{M})$.

Now we show that, for any composable network $\mathcal{M}$, $\text{traces}(\mathcal{M}) \subseteq \text{traces}(\mathcal{N})$. In this case it is sufficient to show that the set $\text{traces}_s(\mathcal{M})$ satisfies the requirements of Definition 4.3.1.

(i) $\varepsilon \in \text{traces}_s(\mathcal{M})$. This follows directly from Definition 4.3.4(i).

(ii) If $\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{N}$, and $\mathcal{N}$ is a successful configuration, then $\omega \in \text{traces}_s(\mathcal{M})$.

If $\mathcal{N}$ is a successful configuration, then $\omega \in \text{traces}_s(\mathcal{N})$ by Definition 4.3.4 (ii). Further, since $\mathcal{M} \stackrel{\tau}{\models\!\!\Longrightarrow} \mathcal{N}$, Lemma 4.3.5 ensures that $t \in \text{traces}_s(\mathcal{M})$.

(iii) If $\mathcal{M} \stackrel{m.c?v}{\models\!\!\Longrightarrow} \mathcal{N}$, and $t \in \text{traces}_s(\mathcal{N})$, then $m.c.v :: t \in \text{traces}_s(\mathcal{M})$.

Suppose that $t \in \text{traces}_s(\mathcal{N})$. Since $\mathcal{M} \stackrel{m.c?v}{\models\!\!\Longrightarrow} \mathcal{N}$, by Definition 4.1.5(3) it follows that there exists two networks $\mathcal{M}_1, \mathcal{M}_2$ such that $\mathcal{M} \stackrel{\tau}{\models\!\!\Longrightarrow} \mathcal{M}_1 \stackrel{m.c?v}{\models\!\!\Longrightarrow} \mathcal{M}_2 \stackrel{\tau}{\models\!\!\Longrightarrow} \mathcal{N}$.

Since $t \in \text{traces}_s(\mathcal{N})$, by Lemma 4.3.5 we obtain that $t \in \text{traces}_s(\mathcal{M}_2)$. Further, by Definition 4.3.4(iv) it follows that $m.c.v :: t \in \text{traces}_s(\mathcal{M}_1)$. Finally, we can apply again Lemma 4.3.4, leading to $m.c.v :: t \in \text{traces}_s(\mathcal{M})$.

(iv) Let t be a trace in $\text{traces}_s(\mathcal{N})$. If $\mathcal{M} \stackrel{c!v\triangleright\eta}{\models\!\!\Longrightarrow} \mathcal{N}$ then $c!v \triangleright \eta :: t \in \text{traces}_s(\mathcal{M})$.

In this case, by Proposition 4.1.8 there exists an index $k \geq 1$ and a collection of non-empty sets of nodes $\{\eta_i\}_{i=1}^k$ such that

- for all $i, j : 1 \leq i, \leq k$, $i \neq j$ implies $\eta_i \cap \eta_j = \emptyset$
- $\bigcup_{i=1}^k \eta_i = \eta$
- $\mathcal{M} \stackrel{\tau}{\models\!\!\Longrightarrow\!\!\longmapsto} \stackrel{c!v\triangleright\eta_1}{\longmapsto\!\!\models\!\!\Longrightarrow} \stackrel{\tau}{\cdots} \stackrel{\tau}{\models\!\!\Longrightarrow\!\!\longmapsto} \stackrel{c!v\triangleright\eta_k}{\longmapsto\!\!\models\!\!\Longrightarrow} \stackrel{\tau}{\mathcal{N}}$

we show that $c!v \triangleright \eta :: t \in \text{traces}_s(\mathcal{M})$ by performing a natural induction over the index k.

$k = 1$ In this case $\mathcal{M} \stackrel{\tau}{\models\!\!\Longrightarrow} \mathcal{M}_1 \stackrel{c!v\triangleright\eta}{\longmapsto} \mathcal{M}_2 \stackrel{\tau}{\models\!\!\Longrightarrow} \mathcal{N}$ for some networks $\mathcal{M}_1, \mathcal{M}_2$.

By Lemma 4.3.5 it follows that $t \in \text{traces}_s(\mathcal{M}_2)$. By Definition v we have that $c!v \triangleright \eta :: t \in \text{traces}_s(\mathcal{M}_1)$. A final application of Lemma 4.3.5 gives us that $c!v \triangleright \eta :: t \in \text{traces}_s(\mathcal{M})$.

$k > 1$. Suppose that the statement is true for $k-1$. In this case, we have that $\mathcal{M} \stackrel{\tau}{\models\!\!\Longrightarrow} \mathcal{M}' \stackrel{c!v\triangleright\eta_1}{\longmapsto} \mathcal{M}'' \stackrel{\tau}{\models\!\!\Longrightarrow} \mathcal{M}_1$ for some network $\mathcal{M}', \mathcal{M}'', \mathcal{M}_1$ such that $\mathcal{M}_1 \stackrel{\tau}{\models\!\!\Longrightarrow\!\!\models\!\!\Longrightarrow\!\!\longmapsto} \stackrel{\tau}{\cdots} \stackrel{\tau}{\models\!\!\Longrightarrow\!\!\longmapsto} \stackrel{c!v\triangleright\eta_k}{\longmapsto\!\!\models\!\!\Longrightarrow} \stackrel{\tau}{\mathcal{N}}$.

Let $\eta' = \bigcup_{i=2}^k \eta_i$. By inductive hypothesis we have that $c!v \triangleright \eta' :: t \in \text{traces}_s(\mathcal{M}_1)$. Further, by Lemma 4.3.5 it follows that $t \in \text{traces}_s(\mathcal{M}'')$.

Now note that $\eta_1 \cap \eta' = \emptyset$. In fact, if $m \in \eta'$, then there exists an index $i > 1$ such that $m \in \eta_i$; however, $\eta_i \cap \eta_1 = \emptyset$, hence $m \notin \eta_1$. Further, $\eta_1 \cup \eta' = \eta_1 \cup \left(\bigcup_{i=2}^k \eta_i\right) = \bigcup_{i=1}^j \eta_i = \eta$.

We can therefore apply Definition 4.3.4(vi) to obtain that $c!v \triangleright \eta :: t \in \text{traces}_s(\mathcal{M}')$. A final application of Lemma 4.3.5 leads to $c!v \triangleright \eta :: t \in \text{traces}_s(\mathcal{M})$.

Proof of Proposition 4.3.10 We need to show that, for any $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$, $\text{zip}_{\mathcal{M}}^{\mathcal{G}}(t, t') \in \text{traces}(\mathcal{M} \,\|\, \mathcal{G})$, provided that the latter is defined.

The statement is proved by performing an induction on t, followed by an inner induction on t'. Throughout the proof, we let $\mathcal{M} = \Gamma_M \triangleright M$ and $\mathcal{G} = \Gamma_G \triangleright n[\![P]\!]$. In many of these cases it will be needed to perform a case analysis on the topology of the networks $\mathcal{M}, \mathcal{G}$. We only show some of the cases that need to be considered when performing the complete proof.

- $t = \omega$. In this case we have that $\mathcal{M} \stackrel{\tau}{\Longrightarrow} \mathcal{M}'$ for some successful configuration $\mathcal{M}'$. Lemma 4.2.21 ensures that $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}$. Now it is trivial to note that the latter is a successful configuration, since $\mathcal{M}'$ has at least one node in which the success process ω is enabled. Therefore $\omega \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$ by Definition 4.3.1 (ii). Now it remains to check that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(\omega, t') = \omega$ for any arbitrary trace $t' \in \text{traces}(\mathcal{G})$.

- $t = c!v \triangleright \{n\}'_t$, $t' = m.c?v :: t_2$. In this case, by Definition 4.3.1(iv) we have that $\mathcal{M} \stackrel{c!v \triangleright \{n\}}{\Longrightarrow} \mathcal{M}'$ for some $\mathcal{M}'$ for which $t_1 \in \text{traces}(\mathcal{M}')$ holds. By Definition 4.3.1(iii), it is also the case that $\mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{G}'$ for some $\mathcal{G}'$ such that $t_2 \in \text{traces}(\mathcal{G}')$.

  Since $t_1 \in \text{traces}(\mathcal{M}'), t_2 \in \text{traces}(\mathcal{N}')$, by inductive hypothesis it holds that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) \in \text{traces}(\mathcal{M}' \parallel \mathcal{G}')$[1].

  Since $\mathcal{M} \stackrel{c!v \triangleright \{n\}}{\Longrightarrow} \mathcal{M}'$, $\mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{G}'$, Proposition 4.2.23(ii) ensures that $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\longmapsto} \mathcal{M}' \parallel \mathcal{G}'$. Thus $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$. Now it remains to note that the last trace is equal to $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \{n\} :: t_1, m.c?v :: t_2)$, as stated in Definition 4.3.9(iv).

- $t = n.c?v :: t_1$, $t' = c!v \triangleright \eta_1 \cup \eta_2$, where $\eta_1 \cap \text{nodes}(\mathcal{M}) = \emptyset$ and $\eta_2 \subseteq \text{nodes}(\mathcal{M})$.

  As for in the previous case, it is easy to show that $\mathcal{M} \stackrel{n.c?v}{\Longrightarrow} \mathcal{M}'$ for some $\mathcal{M}'$ such that $t_1 \in \text{traces}(\mathcal{M}')$, while $\mathcal{G} \stackrel{c!v \triangleright \eta}{\Longrightarrow} \mathcal{G}'$ for some $\mathcal{G}'$ such that $t_2 \in \text{traces}(\mathcal{G}')$.

  By inductive hypothesis we have that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) \in \text{traces}(\mathcal{M}' \parallel \mathcal{G}')$; further, by Proposition 4.2.23(vii) we have that $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v \triangleright \eta_1}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$.

  Therefore, by Definition 4.3.1(iv) it follows that $c!v \triangleright \eta_1 :: \text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$. This trace is exactly $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(n.c?v :: t_1, c!v \triangleright \eta :: t_2)$, as it can be seen from Definition 4.3.9(ix)

- $t' = m.c?v :: t_2$ and $\Gamma_M \not\vdash m$.

  In this case we have that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon, m.c?v :: t_2) = m.c?v :: \text{zip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon, t_2)$. Since $m.c?v :: t_2 \in \text{traces}(\mathcal{G})$, it follows that $\mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{G}'$ for some $\mathcal{G}'$ such that $t_2 \in \text{traces}(\mathcal{G}')$. We can apply Proposition 4.2.23(ix) to the transition $\mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{G}'$ to infer that $\mathcal{M} \parallel \mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{M} \parallel \mathcal{G}'$.

  For $\varepsilon \in \text{traces}(\mathcal{M}), t_2 \in \text{traces}(\mathcal{G}')$, by inductive hypothesis it holds that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon, t_2) \in \text{traces}(\mathcal{M} \parallel \mathcal{G}')$.

  Since $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon, t_2) \in \text{traces}(\mathcal{M} \parallel \mathcal{G}')$, and $\mathcal{M} \parallel \mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{M} \parallel \mathcal{G}'$, it follows that $m.c?v :: \text{zip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon, t_2) \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$. But the last trace is exactly $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon, m.c?v :: t_2)$, as stated in Definition 4.3.9(xi).

- $t = m.c?v :: t_1$, $t' = m.c?v :: t_2$. This is the last case we analyse. In this case we have that $\mathcal{M} \stackrel{m.c?v}{\Longrightarrow} \mathcal{M}'$, $\mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{G}'$, where $t_1 \in \text{traces}(\mathcal{M}'), t_2 \in \text{traces}(\mathcal{G}')$.

  It follows from the inductive hypothesis that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) \in \text{traces}(\mathcal{M}' \parallel \mathcal{G})$, and by Proposition 4.2.23 (x) we can infer the transition $\mathcal{M} \parallel \mathcal{G} \stackrel{m.c?v}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$. Therefore, $m.c?v :: \text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$.

  Then we have that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(m.c?v :: t_1, m.c?v :: t_2) = m.c?v :: \text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) \in \text{traces}(\mathcal{M} \parallel \mathcal{G})$. The equality in this case is obtained by using Definition 4.3.9(xii)

Proof of Lemma 4.3.12 If $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$ for some networks $\mathcal{M}', \mathcal{G}'$, then there exists two sequences $\mathcal{M}_0, \cdots, \mathcal{M}_k$ and $\mathcal{G}_0, \cdots, \mathcal{G}_k$ such that $\mathcal{M} = \mathcal{M}_0, \mathcal{M}' = \mathcal{M}_k, \mathcal{G} = \mathcal{G}_0, \mathcal{G}' = \mathcal{G}_k$ and

$$\mathcal{M}_0 \parallel \mathcal{G}_0 \stackrel{\tau}{\longmapsto} \cdots \stackrel{\tau}{\longmapsto} \mathcal{M}_k \parallel \mathcal{G}_k$$

Let $\langle t_1 , t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, and suppose that $t_1 \in \text{traces}(\mathcal{M}_k)$, $t_2 \in \text{traces}(\mathcal{G}_k)$.

[1] To be formal, the inductive hypothesis states that $\text{zip}^{\mathcal{G}'}_{\mathcal{M}'}(t_1, t_2) \in \text{traces}(\mathcal{M}' \parallel \mathcal{G}')$; however, since the topology of $\mathcal{M}$ is the same of that of $\mathcal{M}'$, and the topology of $\mathcal{G}$ is the same of that of $\mathcal{G}'$, it is easy to note that $\text{zip}^{\mathcal{G}}_{\mathcal{M}}(t_1, t_2) = \text{zip}^{\mathcal{G}'}_{\mathcal{M}'}(t_1, t_2)$ for any traces t_1, t_2.

We perform a natural induction on k to prove that there exists a pair $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t \in \text{traces}(\mathcal{M}_0), t' \in \text{traces}(\mathcal{G}_0)$.

- $k = 0$.

  In this case the statement is trivial. Choose $t = t_1, t_2 = t'$. Since we have that $\mathcal{M}_0 = \mathcal{M}_k, \mathcal{G}_0 = \mathcal{G}_k, \langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{N}}_{\mathcal{M}}(s)$, $t_1 \in \text{traces}(\mathcal{M}_k)$ and $t_2 \in \text{traces}(\mathcal{G}_k)$, it also follows that $t \in \text{traces}(\mathcal{M}_0)$ and $t' \in \text{traces}(\mathcal{M}_0)$.

- $k > 0$.

  Suppose that the statement is true for $k - 1$.

  Since $\mathcal{M}_1 \parallel \mathcal{G}_1 \stackrel{\tau}{\longmapsto} \cdots \stackrel{\tau}{\longmapsto} \mathcal{M}_k \parallel \mathcal{G}_k$, by inductive hypothesis we have that there exists a pair $\langle t'_1, t'_2, \in \rangle \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t'_1 \in \text{traces}(\mathcal{M}'), t'_2 \in \text{traces}(\mathcal{G}_1)$.

  We also have that $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\longmapsto} \mathcal{M}_1 \parallel \mathcal{G}_1$. By Proposition 4.2.12 this leads to four possible cases.

  1. $\mathcal{M}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1$, and $\mathcal{G}_0 = \mathcal{G}_1$. Let $t = t'_1, t' = t'_2$. We show that $t \in \text{traces}(\mathcal{M}_0), t' \in \text{traces}(\mathcal{G}_0)$.

     The last statement is trivial to check, while for the first recall that the sets $\text{traces}(\mathcal{M}_0)$ and $\text{traces}_s(\mathcal{M}_0)$ are equivalent by Proposition 4.3.6. Since $\mathcal{M}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1$, and $t'_1 \in \text{traces}_s(\mathcal{M}_1)$, it follows that $t_1 \in \text{traces}_s(\mathcal{M}_0)$, hence $t'_1 \in \text{traces}(\mathcal{M}_0)$. This is a direct consequence of Definition iii. Thus we have shown that the pair $\langle t, t' \rangle = \langle t'_1, t'_2 \rangle$ is in the set $\text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, and $t \in \text{traces}(\mathcal{M}_0), t' \in \text{traces}(\mathcal{G}_0)$.

  2. $\mathcal{G}_0 \stackrel{\tau}{\longmapsto} \mathcal{G}_1$ and $\mathcal{M}_0 = \mathcal{M}_1$. This case is symmetric to the previous one.

  3. $\mathcal{M}_0 \stackrel{c!v \triangleright \{n\}}{\longmapsto} \mathcal{M}_1$ and $\mathcal{G}_0 \stackrel{m.c?v}{\longmapsto} \mathcal{M}_1$.

     By Definition 4.3.4(iv) it follows that $c!v \triangleright \{n\} :: t'_1 \in \text{traces}_s(\mathcal{M}_0)$, while by Definition 4.3.4(vi) we obtain that $m.c?v :: t'_2 \in \text{traces}_s(\mathcal{G}_0)$.

     Since we have the equivalences $\text{traces}_s(\mathcal{M}_0) = \text{traces}(\mathcal{M}_0), \text{traces}_s(\mathcal{G}_0) = \text{traces}(\mathcal{G}_0)$, it follows that $c!v \triangleright \{n\} :: t'_1 \in \text{traces}(\mathcal{M}_0)$ $c!v \triangleright \{n\} :: t'_2 \in \text{traces}(\mathcal{G}_0)$.

     For it holds by inductive hypothesis that $\langle t'_1, t'_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, Definition 4.3.11 (xii) leads to $\langle c!v \triangleright \{n\} :: t'_1, m.c?v :: t'_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$.

  4. $\mathcal{M}_0 \stackrel{n.c?v}{\longmapsto} \mathcal{M}_1, \mathcal{G}_0 \stackrel{c!v \triangleright \eta}{\longmapsto} \mathcal{G}_1$, where $\eta \subseteq \text{nodes}(\mathcal{M})$. The proof for this case is symmetric to the previous case, this time using Definition 4.3.11 (xiii) instead of Definition 4.3.11 (xii)

Proof of Proposition 4.3.13 Let s be a trace in $\text{traces}(\mathcal{M} \parallel \mathcal{G})$. We show, by induction on s, that there exists a pair $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$.

Throughout the proof we will use several times the equivalence $\text{traces}(\mathcal{M}) = \text{traces}_s(\mathcal{M})$ and $\text{traces}(\mathcal{G}) = \text{traces}_s(\mathcal{G})$, which hold by Proposition 4.3.6

- $s = \varepsilon$.

  This case is trivial, for $\langle \varepsilon, \varepsilon \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(\varepsilon)$ by Definition 4.3.11(i). Further, we have that $\varepsilon \in \text{traces}_s(\mathcal{M}), \varepsilon \in \text{traces}_s(\mathcal{G})$.

- $s = \omega$.

  In this case we have that $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longmapsto} \mathcal{M}' \parallel \mathcal{G}'$ for some $\mathcal{M}', \mathcal{G}'$ such that $\mathcal{M}' \parallel \mathcal{G}'$ is successful. It is straightforward to note that in this case either $\mathcal{M}'$, or $\mathcal{G}'$ is successful. We only consider the first case, for the second one is analogous.

  Suppose that $\mathcal{M}'$ is a successful configuration. Then we have that $\omega \in \text{traces}(\mathcal{M}')$. Also, we have that $\varepsilon \in \text{traces}(\mathcal{G}')$. Now note that the pair $\langle \omega, \varepsilon \rangle$ is in the set $\text{unzip}^{\mathcal{G}}_{\mathcal{M}}(\omega)$. Therefore, we can apply Lemma 4.3.12 to show that there exists a pair $\langle t, t' \rangle$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$ and $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(\omega)$.

- $t = c!v \triangleright \eta :: s'$.

  In this case we have that $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v \triangleright \eta}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$, for some $\mathcal{M}'$ and $\mathcal{G}'$ such that $s' \in \text{traces}(\mathcal{M}' \parallel \mathcal{G}')$. Thus, by inductive hypothesis, there exists a pair $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s')$ such that $t_1 \in \text{traces}(\mathcal{M}'), t_2 \in \text{traces}(\mathcal{G}')$.

  We show that there exists a pair $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s')$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$ by performing an inner induction on the proof of the weak Transition $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v \triangleright \eta}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$.

  - $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longrightarrow} \mathcal{M}_1 \parallel \mathcal{G}_1 \stackrel{c!v \triangleright \eta}{\longmapsto} \mathcal{M}_2 \parallel \mathcal{G}_2 \stackrel{\tau}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$.

    Since there exists a pair $\langle t_1, t_2 \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t_1 \in \text{traces}(\mathcal{M}'), t_2 \in \text{traces}(\mathcal{G}')$, then by Lemma 4.3.12 it follows that there also exists a pair $\langle t_1'', t_2'' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$ such that $t_1'' \in \text{traces}(\mathcal{M}_2), t_2'' \in \text{traces}(\mathcal{G}_2)$.

    For $\mathcal{M}_1 \parallel \mathcal{G}_1 \stackrel{c!v \triangleright \eta}{\longmapsto} \mathcal{M}_2 \parallel \mathcal{G}_2$, by Proposition 4.2.13 there are four possible; we only provide the details for the first of them, as the other three can be handled similarly.

    In this case we have that $\mathcal{M}_1 \stackrel{c!v \triangleright \eta}{\longmapsto} \mathcal{M}_2$, and $\mathcal{G}_1 = \mathcal{G}_2$. Therefore, we have that $t_2'' \in \text{traces}(\mathcal{G}_1)$, and $c!v \triangleright \eta :: t_1'' \in \text{traces}(\mathcal{M}_1)$. Since $\langle t_1'', t_2'' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, by Definition 4.3.11(vi) it follows that $\langle c!v \triangleright \eta :: t_1'', t_2'' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s)$.

    Now it remains to apply again Lemma 4.3.12 to the transition $\mathcal{M} \parallel \mathcal{G} \stackrel{\tau}{\Longrightarrow} \mathcal{M}_1 \parallel \mathcal{G}_1$ to infer that there exist two traces $t \in \text{traces}(\mathcal{M})$ and $t' \in \text{traces}(\mathcal{G})$ such that $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s)$.

  - $\mathcal{M} \parallel \mathcal{G} \stackrel{c!v \triangleright \eta_1}{\Longrightarrow} \mathcal{M}'' \parallel \mathcal{G}'' \stackrel{c!v \triangleright \eta_2}{\Longrightarrow} \mathcal{M}' \parallel \mathcal{G}'$, where $\eta_1 \cup \eta_2 = \eta$ and $\eta_1 \cap \eta_2 = \emptyset$.

    If pair $\langle t_1, t_2, \in \rangle \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(s)$, and $t_1 \in \text{traces}(\mathcal{M}'), t_2 \in \text{traces}(\mathcal{G}')$, then by inductive hypothesis there exists a pair $\langle t_1'', t_2'' \rangle$ such that $t_1'' \in \text{traces}(\mathcal{M}''), t_2'' \in \text{traces}(\mathcal{G}'')$ and $\langle t_1'', t_2'', \in \rangle \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta_2 :: s)$.

    By another application of the inductive hypothesis it follows that there exists a pair $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta_1 :: c!v \triangleright \eta_2 :: s)$ such that $t \in \text{traces}(\mathcal{M}), t' \in \text{traces}(\mathcal{G})$. Since $\eta_1 \cap \eta_2 = \eta$, $\eta_1 \cap \eta_2 = \emptyset$, it suffices to apply Definition 4.3.11 (xi) to obtain that $\langle t, t' \rangle \in \text{unzip}^{\mathcal{G}}_{\mathcal{M}}(c!v \triangleright \eta :: s)$.

- $s = m.c?v :: s'$.

  This case can be handled in a way similar as above, this time using Proposition 4.2.14; therefore we will not provide any details for it.

Proof of Proposition 4.3.16 Let $\mathcal{M}, \mathcal{N}, \mathcal{G}, \mathcal{H}, \mathcal{K}$ be networks as defined in the statement of the Proposition. Further, let $\mathcal{G} = \Gamma_G \triangleright n[\![P]\!]$ for some node n, process P and connectivity graph Γ_G.

We first show that, for any set of nodes η, η' such that $\eta \cap \text{nodes}(\mathcal{M}) = \emptyset$, $\eta' \subseteq \text{nodes}(\mathcal{M})$ and $\mathsf{Input}(\mathcal{H}) = \eta \cup \eta'$, then $\eta \cap \text{nodes}(\mathcal{N}) = \emptyset$. Further, there exists a set $\eta'' \subseteq \text{nodes}(\mathcal{N})$ such that $\mathsf{Input}(\mathcal{K}) = \eta \cup \eta''$.

Recall that $\mathcal{M} \natural \mathcal{G}$ is defined, and that $\mathcal{H} = \text{sym}_{\mathcal{M}}(\mathcal{G})$. Let $\eta = \mathsf{Input}(\mathcal{G})$. It is trivial to show that $\eta \cap \text{nodes}(\mathcal{M}) = \emptyset$, and that $\eta \cap \text{nodes}(\mathcal{N}) = \emptyset$. Further, by Definition 4.2.2, Equation (4.1), it follows that for any $m \in \eta, m \in \mathsf{Input}(\mathcal{H})$ (recall that $\text{nodes}(\mathcal{G}) = \text{nodes}(\mathcal{H})$). Similarly, we can show that $\eta \subseteq \mathsf{Input}(\mathcal{K})$.

Let now $\eta' = \mathsf{Input}(\mathcal{H}) \setminus \eta$. We show that, for any $m \in \eta'$, $m \in \text{nodes}(\mathcal{M})$. Since $m \notin \eta$, and $m \in \mathsf{Input}(\mathcal{H})$, it follows from Equation 4.2 that $\Gamma_G \vdash m \rightarrow n$, and $m \in \text{nodes}(\mathcal{M})$. Thus we have that $\eta' \subseteq \text{nodes}(\mathcal{M})$; since $\eta' = \mathsf{Input}(\mathcal{H}) \setminus \eta$, it follows that $\mathsf{Input}(\mathcal{H}) = \eta \cup \eta'$, as we wanted to prove.

Let $\eta'' = \mathsf{Input}(\mathcal{K}) \setminus \eta$; in a way analogous to the one above, we can show that $\eta'' \subseteq \text{nodes}(\mathcal{K})$, and $\mathsf{Int}(\mathcal{K}) = \eta \cup \eta''$.

Note also that, if $\eta' = \emptyset$, then $\eta'' = \emptyset$. In fact, if $m \in \eta''$, then $n \in \mathsf{Out}_m(\mathcal{N})$. This is because $\mathcal{N} \parallel \mathcal{K}$ is defined, hence $P_s \mathcal{N}, \mathcal{K} = \text{true}$ (recall that P_s is defined formally in Definition 3.1.6), and $m \in \text{nodes}(\mathcal{N})$. For $\mathsf{Input}(\mathcal{N}) = \mathsf{Input}(\mathcal{M})$, there exists a node $m' \in \text{nodes}(\mathcal{M})$ such that $n \in \mathsf{Out}_m(\mathcal{N})$. Since $P_s \mathcal{M}, \mathcal{H} = \text{true}$, then $m' \in \mathsf{Input}(\mathcal{H})$.

With an analogous proof we can show that if $\mathsf{Output}(\mathcal{H}) = \eta \cup \eta'$, where $\eta \cap \text{nodes}(\mathcal{M}) = \emptyset$, $\eta' \subseteq \text{nodes}(\mathcal{M})$, then $\eta \cap \text{nodes}(\mathcal{N}) = \emptyset$ and $\mathsf{Output}(\mathcal{K}) = \eta \cup \eta''$ for some $\eta'' \subseteq \text{nodes}(\mathcal{N})$. Further, if $\eta' = \emptyset$, then $\eta'' = \emptyset$.

Thus, for the networks $\mathcal{H},\mathcal{K}$ we have that if $\mathsf{Input}(\mathcal{H}) = \emptyset$ then $\mathsf{Input}(\mathcal{K}) = \emptyset$; the same applies for $\mathsf{Output}(\mathcal{H})$ and $\mathsf{Output}(\mathcal{K})$. Note that this statement allows us to infer that if $m \in \mathsf{Int}(\mathcal{H})$, and $m \notin \mathrm{nodes}(\mathcal{M})$, then $m \in \mathsf{Int}(\mathcal{K})$.

It remains to prove the main statement. This can be done by proceeding by induction on s, followed by an inner induction on t and t'. The proof for all these cases can be obtained by simple calculations. We just provide the details for some cases.

- $s = \omega, t = c!\{n\} \triangleright :: t_1, t' = m.c?v :: t_2$.

  In this case we have that $\langle\, t_1\,, t_2\,\rangle \in \mathrm{unzip}^{\mathcal{H}}_{\mathcal{M}}(\omega)$. By inductive hypothesis, it follows that there exists a trace $t'' \in \mathrm{switch}_{\mathcal{K}}(t_2)$ such that $\mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(t_1, t'') = \omega$.

  By Definition 4.3.14(iv) we have that $m'.c?v :: t'' \in \mathrm{switch}_{\mathcal{K}}(m.c?v :: t_2)$, where $m' \in \mathsf{Input}(\mathcal{K})$.

  At this point we have that

$$\begin{aligned} \mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(c!v \triangleright \{n\} :: t_1, m'.c?v :: t'') &= \\ \mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(t_1, t'') & \\ &= \omega \end{aligned}$$

- $s = c!v \triangleright \eta :: s'$, $t = n.c?v :: t_1$, $t' = c!v \triangleright (\eta \cup \eta') :: t_2$, where $\emptyset \subset \eta' \subseteq \mathcal{M}$. Since $c!v \triangleright (\eta \cup \eta') :: t_2 \in \mathrm{traces}(\mathcal{H})$ by hypothesis, we have that $\mathsf{Output}(\mathcal{H}) \neq \emptyset$. Therefore, $\mathsf{Output}(\mathcal{K}) \neq \emptyset$. Further, we have already shown that $\mathsf{Output}(\mathcal{K}) = \eta \cup \eta''$, where $\emptyset \subset \eta'' \subseteq \mathrm{nodes}(\mathcal{N})$.

  In this case we have that, if $\langle\, t\,, t'\,\rangle \in \mathrm{unzip}^{\mathcal{H}}_{\mathcal{M}}(s)$, then $\langle\, t_1\,, t_2\,\rangle \in \mathrm{unzip}^{\mathcal{H}}_{\mathcal{M}}(s')$, so that by inductive hypothesis there exists a trace $t'' \in \mathrm{switch}_{\mathcal{K}}(t_2)$ such that $\mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(t_1, t'') = s'$.

  Note also that, if we let $\eta_1 = \mathsf{Input}(\mathcal{K})$, then $c!v \triangleright \eta_1 :: t'' \in \mathrm{switch}_{\mathcal{K}}(c!v \triangleright (\eta \cup \eta') :: t_2)$. In this case we know that $\eta_1 = \eta \cup \eta''$.

  By calculations, it follows that

$$\begin{aligned} \mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(n.c?v :: t_1, c!v \triangleright (\eta \cup \eta'') :: t'') &= c!v \triangleright \eta :: \mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(t_1, t'') \\ &= c!v \triangleright \eta :: s' \\ &= s \end{aligned}$$

- $s = m.c?v :: s'$, $t = m.c?v :: t_1$ and $t' = m.c?v :: t_2$.

  In this case, if $\langle\, t, t'\,\rangle \in \mathrm{unzip}^{\mathcal{H}}_{\mathcal{M}}(s)$, then $\langle\, t_1\,, t_2\,\rangle \in \mathrm{unzip}^{\mathcal{H}}_{\mathcal{M}}(s')$. Also, since $m.c?v :: t_1 \in \mathrm{traces}(\mathcal{M})$, we have that $m \in \mathsf{Int}(\mathcal{M})$, hence $m \notin \mathrm{nodes}(\mathcal{N})$.

  By inductive hypothesis, it holds that there exists a trace $t'' \in \mathrm{switch}_{\mathcal{K}}(t_2)$ such that $\mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(t_1, t'') = s$.

  Also, we have that $m \in \mathsf{Input}(\mathcal{H})$ and $m \notin \mathrm{nodes}(\mathcal{M})$, which ensures that $m \in \mathsf{Input}(\mathcal{K})$. Thus we have that $m.c?v :: t'' \in \mathrm{switch}_{\mathcal{K}}(m.c?v :: t_2)$. By calculations we obtain that

$$\begin{aligned} \mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(m.c?v :: t_1, m.c?v :: t'') &= m.c?v :: \mathrm{zip}^{\mathcal{K}}_{\mathcal{N}}(t_1, t'') && \text{(A.9)} \\ &= m.c?v :: s' && \text{(A.10)} \\ &= s && \text{(A.11)} \end{aligned}$$

Proof of Lemma 4.3.21 First note that every reduction of the computation listed in the hypothesis correspond to a strong τ transition. In fact, recall that $\mathsf{Int}(\mathcal{M}_0) = \{e_1, \cdots, e_k\}$ by hypothesis, $\mathsf{Int}(\mathcal{T}) = \emptyset$ and $\{e_1, \cdots, e_k\} \subseteq \mathrm{nodes}(\mathcal{T}_0)$. It follows from Lemma 3.2.1 that $\mathsf{Int}(\mathcal{M}_0 \between \mathcal{T}_0) = \emptyset$. Now, by Proposition 4.1.4 we have that whenever $\mathcal{M}_j \between \mathcal{T}_j \twoheadrightarrow \mathcal{M}_{j+1} \between \mathcal{T}_{j+1}$, for some index j such that $0 \leq j \leq k$, then either $\mathcal{M}_j \between \mathcal{T}_j \xrightarrow{\tau} \equiv \mathcal{M}_{j+1} \between \mathcal{T}_{j+1}$ or $\mathcal{M}_j \between \mathcal{T}_j \xrightarrow{c!v \triangleright \eta} \equiv \mathcal{M}_{j+1} \between \mathcal{T}_{j+1}$ for some channel c, value v and non-empty set of nodes η. However, the

last case is not possible. In fact, in this case we would have that $\eta \subseteq \mathsf{Int}(\mathcal{M} \between \mathcal{T}) = \emptyset$, contradicting the fact that $\eta \neq \emptyset$. Therefore $\mathcal{M}_j \between \mathcal{T}_j \stackrel{\tau}{\longmapsto} \equiv \mathcal{M}_{j+1} \between \mathcal{T}_{j+1}$.

Since $\mathcal{M}_0$ is a composable network it cannot contain the success process ω at the code of some of its nodes. Further, for any network $\mathcal{T}' \in \mathrm{Tests}_t$, only the code running at node cn can contain the process ω. It is straightforward to note that in this case we have that $\mathcal{T}_n = \Gamma_T \rhd \prod_{i=1}^{k} \mathbf{0}[\![e_i]\!] \,|\, \omega[\![cn]\!]$.

Further, note that this is possible only if there exists two indexes $h, 0 \leq h < j$ and $i, 0 \leq i \leq k$ such that $\mathcal{T}_h = \Gamma_T \rhd \prod_{i=1}^{k} e_i[\![cc!\langle \mathbf{CHECK} \rangle]\!] \,|\, cc[\![cc?(x_1, \cdots x_k).\omega]\!]$. Therefore, for any index h it follows that the code running at node e_i in $\mathcal{T}_h$ cannot be the empty process $\mathbf{0}$.

Since the code running at node cn in the network $\mathcal{T}_0$ is deterministic, we also have that there exists a least index j such that $\mathcal{T}_j = \Gamma_T \rhd \prod_{i=1}^{k} e_i[\![P_i^j]\!] \,|\, cn[\![P_t]\!]$. Consider now the sequence of extensional transitions $\mathcal{M}_0 \between \mathcal{T}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_j \between \mathcal{T}_t$. Note that, for any index $i : 0 \leq i \leq j$, the code running at node e_i in network $\mathcal{T}_j$ cannot be the deadlocked process $\mathbf{0}$.

We show, by performing a case analysis on $\mathcal{T}_0$, that $\mathcal{T}_j = \mathcal{T}_t$, and that $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_j$. First, suppose that $\mathcal{T}_0 = \mathcal{T}^t_{\mathrm{next}}$, and consider the sequences of τ-extensional transitions $\mathcal{M}_0 \between \mathcal{T}_0 \stackrel{\tau}{\longmapsto} \cdots \stackrel{\tau}{\longmapsto} \mathcal{M}_j \between \mathcal{T}_j$. As a consequence of Proposition 4.2.5 we can rewrite this sequence of transitions as $\mathcal{M}_0 \parallel \mathcal{T}'_0 \stackrel{\tau}{\longmapsto} \cdots \stackrel{\tau}{\longmapsto} \mathcal{M}_j \parallel \mathcal{T}'_j$, where $\mathcal{T}'_i = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_i)$ for any $i = 0, \cdots, j$[2].

Recall that $\mathcal{T}'_0 = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}^{\mathcal{M}}_{\mathrm{next}})$; We show that $\mathcal{T}'_j = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}'_t)$ and that $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_j$ by induction on j. Note that we cannot have the case $j = 0$, for we have already shown that $\mathcal{T}'_j \equiv \Gamma'_T \rhd \prod_{i=1}^{k} e_i[\![P_i^j]\!] \,|\, cn[\![P_t]\!]$, while $\mathcal{T}'_0 = \Gamma'_T \rhd \prod_{i=1}^{k} [\![\,]\!] | [\![cc?(x).P_t^i + \mathbf{LOCK}]\!] cn[\![cc!\langle \mathbf{PROCEED} \rangle.P_t]\!]$.

- $j = 1$. Then $\mathcal{M}_0 \parallel \mathcal{T}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1 \parallel \mathcal{T}_1$, where $\mathcal{T}'_1 \equiv \Gamma'_T \rhd \prod_{i=1}^{k} e_i[\![P_i^j]\!] \,|\, cn[\![P_t]\!]$.

  Since $\mathcal{T}'_0 = \Gamma'_T \rhd \mathcal{T}'_0 = \Gamma'_T \rhd \prod_{i=1}^{k} [\![\,]\!] | [\![cc?(x).P_t^i + \mathbf{LOCK}]\!] cn[\![cc!\langle \mathbf{PROCEED} \rangle.P_t]\!]$ and $\mathcal{T}'_1 \equiv \Gamma'_T \rhd \prod_{i=1}^{k} e_i[\![P_i^j]\!] \,|\, cn[\![P_t]\!]$, in this case we have that a broadcast along channel cc has been performed by $\mathcal{T}'_0$, and $\mathcal{T}'_1 \equiv \Gamma'_T \rhd \prod_{i=1}^{k} | [\![P_t^i]\!] e_i[\![.P_t^i]\!] \,|\, cn[\![P_t]\!] = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_t)$.

  In other words, we have that $\mathcal{M}_0 \parallel \mathcal{T}'_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1 \parallel \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_t)$, and $\mathcal{T}'_0 \stackrel{\tau}{\longmapsto} \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_t)$. This corresponds to case (ii) of Proposition 4.2.12, from which it follows that $\mathcal{M}_0 = \mathcal{M}_1$; hence $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_1$.

- $j > 1$. Suppose that the statement is true for $j - 1$, and consider again the transition $\mathcal{M}_0 \parallel \mathcal{T}'_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1 \parallel \mathcal{T}'_1$. According to proposition 4.2.12 there are four possible cases; however, we show that only one can actually happen.

  - $\mathcal{M}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1$, and $\mathcal{T}'_1 = \mathcal{T}'_0 = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}^t_{\mathrm{next}})$. This is the only possible case.
  - $\mathcal{T}'_0 \stackrel{\tau}{\longmapsto} \mathcal{T}'_1$; this is possible only if network $\mathcal{T}'_0$ broadcasts a message along channel cc; however, we have already seen that $\mathcal{T}_1 = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_t)$, hence $\mathcal{M}_1 \parallel \mathcal{T}'_1 = \mathcal{M}_1 \between \mathcal{T}_t$. Since we have that $j > 1$, this contradicts the hypothesis that j is the least index such that $\mathcal{T}_j = \mathcal{T}_t$; therefore, this case is not possible.
  - $\mathcal{M}_0 \stackrel{d!v \rhd \eta'}{\longmapsto} \mathcal{M}_1$ for some non-empty set of nodes η', and $\mathcal{T}'_0 \stackrel{m.c?v}{\longmapsto} \mathcal{T}_1$ for some $m \in \mathsf{Input}(\mathcal{T}'_0)$. Note that in this case we have that $d \neq cc$, since cc does not appear in the code of $\mathcal{M}_0$ by hypothesis. Recall that $\mathsf{Int}(\mathcal{M}) = \{e_1, \cdots, e_k\}$. Thus, there exists at least an index i such that $e_i \in \eta'$. That is, $\Gamma_M \vdash m' \to e_i$ for some node e_i, and $m' = m$. Now, note that in $\mathcal{T}'_0$ node e_i reacts to a broadcast from node m by making its code evolve in the deadlocked process $\mathbf{0}$. However, we have already proved that for any index $i' = 0, \cdots, j$, none of the nodes e_i, where $i = 1, \cdots, k$ can run the code $\mathbf{0}$. Therefore, this case is not possible.
  - The last possible case is that in which $\mathcal{T}'_0 \stackrel{c!v \rhd \eta'}{\longmapsto} \mathcal{T}'_1$ for some channel c, value v and non-empty set of nodes η. Note that this requires that the network $\mathcal{T}'_0$ performs a broadcast. This is possible only if the node that performs the broadcast is cn; that is, $\eta' = \mathsf{Out}_{cn}(\mathcal{T}'_0)$.

[2] In practice, we have that $\mathcal{T}'_i = \mathrm{sym}_{\mathcal{M}_i}(\mathcal{T}_i) = \Gamma'_T \rhd T_i$ for any $i = 0, \cdots, j$. However, this is the same as $\mathrm{sym}_{\mathcal{M}}(\mathcal{T}_i)$, for $\mathcal{M}$ has the same topological structure of $\mathcal{M}_i$.

Since we are assuming that $cn \notin (\Gamma_V)$, there exists no node m in Γ_M such that $\Gamma_M \vdash m \to cn$. Therefore, we have that $\Gamma'_T \vdash m \to cn$ for no $m \in (\Gamma_M)_v$. Since $\mathsf{Out}_{cn}(\mathcal{T}'_0) = \emptyset$, it follows that also $\mathsf{Out}_{cn}(\mathcal{T}'_0) = \emptyset$, which contradicts the requirement that η' is non-empty. Thus this case is not possible.

Therefore, the only possibility in this case is that $\mathcal{M}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1$ and $\mathcal{T}'_0 = \mathcal{T}'_1 = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}^t_{\mathrm{next}})$. Then we can apply the inductive hypothesis to infer that $\mathcal{T}'_j = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_t)$ and $\mathcal{M}_1 \stackrel{\tau}{\Longmapsto} \mathcal{M}_j$, from which it follows that $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_j$.

The next case we need to consider is that in which $\mathcal{T}_0 = \mathcal{T}^t_\eta$ for some set η such that $|\eta| = k-1$. In a way similar as the one above, we can prove that there exists an index $j' > 0$ such that $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_{j'}$ and $\mathcal{T}_{j'} = \mathcal{T}^t_{\mathrm{next}}$; then, from the discussion above, we obtain that there exists an index $j > j'$ such that $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_j$ and $\mathcal{T}_j = \mathcal{T}_t$.

We can iterate this procedure, first by looking at a network of the form $\mathcal{T}^t_\eta$ where $|\eta| = k-2$, to show that an index j'' such that $\mathcal{T}''_j = \mathcal{T}^t_{\eta'}$, where $|\eta'| = k-1$ can be reached. At the end of this iteration we find that the statements also holds if $\mathcal{T}_0 = \mathcal{T}^t_{\mathrm{check}}$, which concludes the proof.

Proof of Lemma 4.3.23 Suppose that $\mathcal{M}_0 \between \mathcal{T}_0 \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{M}_n \between \mathcal{T}_n$, where $\mathcal{T}_n$ is a successful configuration. We have already proved in Lemma 4.3.21 that each transition corresponds to a $\stackrel{\tau}{\longmapsto}$ transition; further, we have that $\mathcal{T}_n \equiv \Gamma_n \rhd \prod_{i=1}^k e_i[\![\mathbf{0}]\!] \mid cn[\![\omega]\!]$ and, in order for the computation above to be successful, the code at some node e_i, $i = 1, \cdots, k$ cannot be $\mathbf{0}$ throughout the computation. That is, for any $h : 0 \le h < n$ and $i : 1 \le i \le k$, we have that $\mathcal{T}_h \not\equiv \mathcal{T}_h \rhd e_j[\![\mathbf{0}]\!] \mid \prod_{j=1\, j\neq i}^k e_j[\![P^j_h]\!] \mid cn[\![P_h]\!]$.

Further, since the code running at node cn is deterministic, there exists a least index $j \le n$ such that $\mathcal{T}_j \equiv \prod_{i=1}^k e_i[\![P^j_i]\!] \mid cn[\![P^{t'}_{\mathrm{check}}.P^{t'}_{\mathrm{next}}.P_t]\!]$.

We prove, by well-founded induction over the relation $\prec$, that if $\mathcal{T}_0 \in \mathrm{Dtest}_t$ then $\mathcal{T}_j = \mathcal{T}^{t'}_{\mathrm{check}}$, and $\mathcal{M}_0 \stackrel{c!v \rhd \eta^\times}{\Longmapsto} \mathcal{M}_j$.

- The base case is that in which $\eta^\times = \emptyset, \eta^\rightarrow = \emptyset$ and $\eta^\downarrow = \eta$. We have already noticed that $\mathcal{T}_0 = \mathcal{T}^{t'}_{\emptyset,\emptyset,\eta} = \mathcal{T}_{\mathrm{check}}$, so that $j = 0$. Further, we have that $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_0$, so that there is nothing to prove in this case.
- Let now $\eta^\times, \eta^\rightarrow, \eta^\downarrow$ be an arbitrary partition of η, and let $\mathcal{T}_0 = \mathcal{T}^{t'}_{\eta^\times,\eta^\rightarrow,\eta^\downarrow}$. Since we are assuming that the computation $\mathcal{M}_0 \between \mathcal{T}_0 \stackrel{\tau}{\longmapsto} \cdots \stackrel{\tau}{\longmapsto} \mathcal{M}_n \between \mathcal{T}_n$ is successful, and $\mathcal{T}_0$ is not a successful configuration, there exists a least index $j' \le j$ such that $\mathcal{T}_{j'}$ is not $\mathcal{T}_0$.

  We perform an inner induction on j' to show that $\mathcal{T}_{j'} = \mathcal{T}^{t'}_{\eta^\times_1,\eta^\rightarrow_1,\eta^\downarrow_1}$ for some sets $\eta^\times_1, \eta^\rightarrow_1, \eta^\downarrow_1 \in \mathrm{Dtests}_t$ such that $\mathcal{T}_0 \prec \mathcal{T}_{j'}$. Further, if $\eta' = \eta^\times \setminus \eta^\times_1$ is non-empty, then $\mathcal{M}_0 \stackrel{c!v \rhd \eta'}{\Longmapsto} \mathcal{M}_{j'}$, otherwise $\mathcal{M}_0 \stackrel{\tau}{\Longmapsto} \mathcal{M}_{j'}$.

  - $j' = 0$. This case is vacuous, for we require that $\mathcal{T}_{j'} \neq \mathcal{T}_0$.
  - $j' > 0$. Consider the transition $\mathcal{M}_0 \between \mathcal{T}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1 \between \mathcal{T}_1$. We can rewrite this transition as $\mathcal{M}_0 \parallel \mathcal{T}'_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1 \parallel \mathcal{T}'_1$, where $\mathcal{T}'_0 = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_0)$ and $\mathcal{T}'_1 = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}_1) = \Gamma'_T \rhd T_1$, by applying Proposition 4.2.5. By Proposition 4.2.12 there are four possible cases.
    - $\mathcal{M}_0 \stackrel{\tau}{\longmapsto} \mathcal{M}_1$, and $\mathcal{T}'_0 = \mathcal{T}'_1$. In this case the result follows by the inductive hypothesis.
    - $\mathcal{M}_0 \stackrel{d!w \rhd \eta''}{\longmapsto} \mathcal{M}_1$. Further, there exists a node $m \in \mathsf{Input}(\mathcal{T}'_0)$ such that $\mathsf{In}_m(\mathcal{T}'_0) = \eta''$, and $\mathcal{T}'_0 \stackrel{m.c?v}{\longmapsto} \mathcal{T}'_1$. In this case $j' = 1$.

      Now note that d has to be the channel c, for otherwise one the code running in the nodes in η'' would evolve in the deadlocked process $\mathbf{0}$; similarly, we have that $w = v$. Further, note that $\eta'' \subseteq \eta'$, for otherwise the nodes in $\eta'' \setminus \eta^\times$ would deadlock too. Let $\eta^\times_1 = \eta^\times \setminus \eta''$, $\eta^\rightarrow_1 = \eta^\rightarrow \cup \eta^\times_1, \eta^\downarrow_1 = \eta_\downarrow$.

We can check that in this case we have that $\mathcal{T}'_1$ is the network

$$\Gamma'_T \rhd \prod_{i:e_i\in\eta^\times} e_i[\![P_t]\!] \mid \prod_{i:e_i\in\eta^\rightarrow} e_i[\![cc!\langle\mathbf{DETECTED}\rangle.P'_{\text{next}}.P'_{\text{check}}.P'_{t'}]\!] \mid$$
$$\prod_{i:e_i\in\eta^\rightarrow} e_i[\![P'_{\text{next}}.P'_{\text{check}}.P'_{t'}]\!] \mid cn[\![cc?\big(x_1,\cdots,x_{|\eta^\times\cup\eta^\rightarrow|}\big).P_{\text{next}}.P_{\text{check}}.P_{t'}]\!]$$

which is exactly $\text{sym}_{\mathcal{M}}(\eta_1^\times)\mathcal{T}^{t'}_{\eta_1^G o,\eta_1^\downarrow}$. Further, since $\eta_1^\times = \eta^\times \setminus \eta''$, and $\eta'' \subseteq \eta^\times$, it follows that $\eta'' = \eta^\times \setminus \eta_1^\times$. Since $\mathcal{M}_0 \overset{c!v\rhd\eta''}{\longmapsto} \mathcal{M}_1$, we also have that $\mathcal{M}_0 \overset{c!v\rhd\eta''}{\Longmapsto} \mathcal{M}_1$.

* $\mathcal{T}'_0 \overset{\tau}{\Longmapsto} \mathcal{T}'_1$, and $\mathcal{M}_0 = \mathcal{M}_1$. This case can happen only if one of the nodes e_i, $i = 1,\cdots k$, has broadcast the acknowledgement value along channel cc; further, $\mathsf{Out}_{e_i}(\mathcal{T}'_0) = \emptyset$. Since node e_i can broadcast the acknowledgement value to node cn, then it has to be contained in the set $\eta^\rightarrow$. Further, if we choose $\eta_1^\times = \eta^\times$, $\eta_1^\rightarrow = \eta^\rightarrow \setminus \{e_i\}$, and $\eta_1^\downarrow = \eta^\rightarrow \setminus \{e_i\}$, then $\mathcal{T}'_1 = \text{sym}_{\mathcal{M}}(\mathcal{T}^{t'}_{\eta_1^\times,\eta_1^\rightarrow,\eta_1^\downarrow})$. Since $\mathcal{M}_0 = \mathcal{M}_1$, then we have that $\mathcal{M}_0 \overset{\tau}{\Longmapsto} \mathcal{M}_1$.
* $\mathcal{T}'_0 \overset{cc!\delta_d\rhd\eta_M}{\longmapsto} \mathcal{T}'_1$ for some non-empty set of nodes η_M. Since we are assuming that no node in the network $\mathcal{M}_0$ can receive values along the channel cc, then we have that $\mathcal{M}_1 = \mathcal{M}_0$. The rest of the proof for this case is analogous to the previous one.

Recall that we want to prove that if $\mathcal{T}_0 \in \text{Dtest}_t$ then $\mathcal{T}_j = \mathcal{T}^{t'}_{\text{check}}$, and $\mathcal{M}_0 \overset{c!v\rhd\eta^\times}{\Longmapsto} \mathcal{M}_j$. Here $\mathcal{T}_j$ is the least configuration for the testing component in the successful computation we are considering for which node cn is running the process $P_{\text{check}}.P_{\text{next}}.P_{t'}$.

So far we have showed that for $\mathcal{T}_{j'} = \mathcal{T}^{t'}_{\eta_1^\times,\eta_1^\rightarrow,\eta_1^\downarrow}$ for some sets $\eta_1^\times,\eta_1^\rightarrow,\eta_1^\downarrow \in \text{Dtests}_t$ such that $\mathcal{T}_0 \prec \mathcal{T}_{j'}$. Further, if $\eta' = \eta^\times \setminus \eta_1^\times$ is non-empty, then $\mathcal{M}_0 \overset{c!v\rhd\eta'}{\Longmapsto} \mathcal{M}_{j'}$, otherwise $\mathcal{M}_0 \overset{\tau}{\Longmapsto} \mathcal{M}_{j'}$. Then the configuration $\mathcal{M}_{j'} \nparallel \mathcal{T}^{t'}_{\eta_1^\times,\eta_1^\rightarrow,\eta_1^\downarrow}$ reaches, after a finite number of steps, the configuration $\mathcal{T}_j \mid \mathcal{T}_j$; by well-founded induction it holds that $\mathcal{T}_j = \mathcal{T}^{t'}_{\text{check}}$, and $\mathcal{M}_{j'} \overset{c!v\rhd\eta_1^\times}{\Longmapsto} \mathcal{M}_j$.

If $\eta^\times \setminus \eta_1^\times = \emptyset$, then $\eta^\times = \eta_1^\times$. This is because $\eta^\times \subseteq \eta_1^\times$. In this case we have proved that $\mathcal{M}_0 \overset{\tau}{\Longmapsto} \mathcal{M}_{j'} \overset{c!v\rhd\eta_1^\times}{\Longmapsto} \mathcal{M}_j$, which is equivalent to $\mathcal{M}_0 \overset{c!v\rhd\eta^\times}{\Longmapsto} \mathcal{M}_j$. Otherwise, $\mathcal{M}_0 \overset{c!v\rhd\eta^\times\setminus\eta_1^\times}{\Longmapsto} \mathcal{M}_{j'} \overset{c!v\rhd\eta^\times}{\Longmapsto} \mathcal{M}_j$; the sets The sets $\eta^\times \setminus \eta_1^\times$ and $\eta^\times$ are obviously disjoint, and their union correspond to $\eta^\times$. Therefore, by Definition 4.1.5(3b) we obtain that $\mathcal{M}_0 \overset{c!v\rhd\eta^\times}{\Longmapsto} \mathcal{M}_j$, as we wished to prove.

A.5 Proofs of the Results Concerning the Must-testing Preorder

Proof of Lemma 4.4.16 We only prove the first statement in the case $\mu = m.c?v$. The proofs for any other case, as well as the proof of the second statement, are in fact similar in style.

Let $\mathcal{M}$ be a finitary network and μ be an extensional action; we show that if $\mathcal{M}$ is finitary, with $\sup\{\mathcal{M} \overset{m.c?v}{\Longmapsto} \mathcal{M}'\} = \infty$, then $\mathcal{M}$ is not strongly convergent, thus contradicting the hypothesis.

To this end, we build a directed tree T as the smallest tree which satisfies the following conditions:

- Vertices of T are finite sequences of extensional transitions, with the root of T being the sequence of length 0 coinciding with $\mathcal{M}$
- If $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}''$ and $\mathcal{M}'' \overset{\mu}{\longmapsto} \mathcal{M}'$, where μ is either τ or $m.c?v$, then there exists an edge from $\mathcal{M} \overset{\tau}{\Longmapsto} \mathcal{M}''$ to $\mathcal{M} \overset{\mu}{\Longmapsto} \mathcal{M}'$
- if $\mathcal{M} \overset{m.c?v}{\Longmapsto} \mathcal{M}''$ and $\mathcal{M}'' \overset{\tau}{\longmapsto} \mathcal{M}'$, then there exists a vertex from $\mathcal{M} \overset{m.c?v}{\Longmapsto} \mathcal{M}''$ to $\mathcal{M} \overset{m.c?v}{\Longmapsto} \mathcal{M}'$

Note that $\mathcal{M} \stackrel{\mu}{\Longmapsto} \mathcal{M}'$ appears in T only if we can derive the corresponding weak extensional transition for $\mathcal{M}$; since no other vertices appear in T, it follows that T has a single connected component. Also, since $\mathcal{M}$ is assumed to be a finitary network, each vertex in T has finite degree, for the number of outgoing edges for a vertex of the form $\mathcal{M} \stackrel{\mu}{\Longmapsto} \mathcal{M}'$ is less or equal to the number of transitions which $\mathcal{M}'$ can perform.

Finally, the tree T has an infinite number of vertices. By assumption the quantity $\sup\{\mathcal{M} \stackrel{m.c?v}{\Longmapsto} \mathcal{M}'\}$; hence, there exists an index $k \geq 0$ such that for any $n \geq k$, there exists a transition of the form $\mathcal{M} \stackrel{m.c?v}{\Longmapsto} \mathcal{M}''$ of length n such that $\mathcal{M}'' \stackrel{\tau}{\longmapsto} \mathcal{M}'$, leading to length$(\mathcal{M} \stackrel{m.c?v}{\Longmapsto} \mathcal{M}') = n + 1$. That is, we are able to associate to any number $n \geq k$ a sequence of transitions of length n; since such a sequence of transitions corresponds to a vertex in T, it follows that T has an infinite number of vertices.

Therefore we can apply Corollary 4.4.15 to prove that the directed tree T has an infinite path rooted in the network $\mathcal{M}$; this corresponds to an infinite sequence of transitions of the form

$$\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{M}_1 \stackrel{\tau}{\longmapsto} \cdots \stackrel{\tau}{\longmapsto} \mathcal{M}_{k-1} \stackrel{m.c?v}{\longmapsto} \mathcal{M}_k \stackrel{\tau}{\longmapsto} \cdots \stackrel{\tau}{\longmapsto} \mathcal{M}_{k+h} \stackrel{\tau}{\longmapsto} \cdots$$

Thus we have that $\mathcal{M}_k \uparrow$ and, since $\mathcal{M} \stackrel{m.c?v}{\Longmapsto} \mathcal{M}_k$, it follows that $\mathcal{M} \Uparrow_{m.c?v}$. By definition it follows that $\mathcal{M}$ is not strongly convergent.

Outline of the Proof of Lemma 4.4.17 Consider the network $\mathcal{M} \between \mathcal{T}$; we show that, for any sequence of extensional transitions of the form

$$(\mathcal{M} \between \mathcal{T}) \stackrel{\mu_1}{\longmapsto} (\mathcal{M}^1 \between \mathcal{T}^1) \stackrel{\mu_2}{\longmapsto} \cdots \stackrel{\mu_k}{\longmapsto} (\mathcal{M}^k \between \mathcal{T}^k) \tag{A.12}$$

of length $n_{\mathcal{M}}$ there exists an index $j \in J$ such that $\mathcal{N}_j \between \mathcal{T}$ is equipped with the sequence of weak extensional transitions

$$(\mathcal{N}_j \between \mathcal{T}) \stackrel{\mu_1}{\Longmapsto} (\mathcal{N}_j^1 \between \mathcal{T}_1) \stackrel{\mu_2}{\Longmapsto} \cdots \stackrel{\mu_n}{\Longmapsto} (\mathcal{N}_j^k \between \mathcal{T}^k) \tag{A.13}$$

where the length $n_{\mathcal{N}}$ of the above computation fragment is greater or equal to $n_{\mathcal{M}}$.

This statement is proved by reasoning by induction on k.

$k = 0$ In this case the sequence of extensional transitions described in Equation (A.12) consists of the only network $(\mathcal{M} \between \mathcal{T})$. In this case, for any index $j \in J$, we have the empty sequence of weak extensional transitions consisting of the only network $(\mathcal{N}_j \between \mathcal{T})$; hence, in this case there is nothing to prove.

$k > 0$ Suppose that the statement is true for $k - 1$; we have that

$$(\mathcal{M} \between \mathcal{T}) \stackrel{\mu_1}{\longmapsto} (\mathcal{M}^1 \between \mathcal{T}^1)$$

By Proposition 4.2.5 we can rewrite the transition above as $(\mathcal{M} \parallel \mathcal{T}_{\mathcal{M}}) \stackrel{\mu_1}{\longmapsto} (\mathcal{M}^1 \parallel \mathcal{T}^1_{\mathcal{M}})$, where $\mathcal{T}_{\mathcal{M}} = \mathrm{sym}_{\mathcal{M}}(\mathcal{T})$, $\mathcal{T}^1_{\mathcal{M}} = \mathrm{sym}_{\mathcal{M}}(\mathcal{T}^1)$ [3]

The rest of the proof is carried out by performing a case analysis on the action μ_1; here we only show some of the details for the case $\mu_1 = \tau$. By Proposition 4.2.12 there are four possible sub-cases:

(i) $\mathcal{M} \stackrel{\tau}{\longmapsto} \mathcal{M}^1$ and $\mathcal{T}^1 = \mathcal{T}$,

(ii) $\mathcal{T} \stackrel{\tau}{\longmapsto} \mathcal{T}^1$ and $\mathcal{M}^1 = \mathcal{M}$,

(iii) $\mathcal{M} \stackrel{c!v \triangleright \{n\}}{\longmapsto} \mathcal{M}'$ and $\mathcal{T} \stackrel{m.c?v}{\longmapsto} \mathcal{T}^1$ or

(iv) $\mathcal{M} \stackrel{n.c?v}{\longmapsto} \mathcal{M}^1$ and $\mathcal{T} \stackrel{c!v \triangleright \eta}{\Longmapsto} \mathcal{T}^1$, with $\eta \subseteq \mathrm{nodes}(\mathcal{N})$.

[3] In practice, we have that $\mathcal{T}^1_{\mathcal{M}} = \mathrm{sym}_{\mathcal{M}^1}(\mathcal{T}^1)$; however, this network coincides with $\mathrm{sym}_{\mathcal{M}}(\mathcal{T}^1)$, for the topological structure of $\mathcal{M}$ and $\mathcal{M}^1$ is the same, and the definition of $\mathrm{sym}_{\mathcal{M}}(\cdot)$ depends only on the topological structure of $\mathcal{M}$.

We only provide the details for Case (iii). Since $\mathrm{DTraces}(\mathcal{M}) \subseteq \bigcup_{j\in J} \mathrm{DTraces}(\mathcal{N}_j)$, and $\mathcal{M} \stackrel{c!v\triangleright\{n\}}{\longmapsto} \mathcal{M}^1$, it follows that there exists an index $j \in J$ such that $\mathcal{N}_j \stackrel{c!v\triangleright\{n\}}{\Longmapsto} \mathcal{N}'$ for some network $\mathcal{N}'$; in particular, the set $\mathscr{N} = \{\mathcal{N}' \mid \mathcal{N}_j \stackrel{c!v\triangleright\{n\}}{\Longmapsto} \mathcal{N}' \text{ for some } j \in J\}$ is non-empty.

Further, the set $\mathscr{N}$ is finite; in fact, if it were infinite, there would exist an index $j \in J$ such that the set $\mathscr{N}_j = \{\mathcal{N}' \mid \mathcal{N}_j \stackrel{c!v\triangleright\{n\}}{\Longmapsto} \mathcal{N}'\}$ would be infinite. For such a set, the quantity $k = \sup\{\mathcal{N}_j \stackrel{c!v\triangleright\{n\}}{\Longmapsto} \mathcal{N}' \mid \mathcal{N}' \in \mathscr{N}_j\}$ would be defined and finite, for otherwise it would follow that $\mathcal{N}_j$ is not strongly convergent as a direct consequence of Lemma 4.4.16. Now it is possible to prove, by natural induction on k and by using the assumption that $\mathcal{N}_j$ is finitary, that the set of networks $\mathscr{N}_j'$ which can be reached by the latter via a sequence of at most k (arbitrary) strong extensional transition is finite, hence so is the set set $\mathscr{N}_j$, for it is trivial to note that $\mathscr{N}_j \subseteq \mathscr{N}_j'$.

Thus we have shown that the set $\mathscr{N}$ is non-empty and finite. Now we show that it also holds that $\mathrm{DTraces}(\mathcal{M})^1 \subseteq \bigcup_{\mathcal{N}'\in\mathscr{N}} \mathrm{DTraces}(\mathcal{N}')$; this enables us to apply the inductive hypothesis to the sequence of extensional transitions

$$(\mathcal{M}^1 \between \mathcal{T}^1) \stackrel{\mu_2}{\longmapsto} \cdots \stackrel{\mu_k}{\Longmapsto} (\mathcal{M}^k \between \mathcal{T}^k)$$

and the set of networks $\mathscr{N}$. Let $t \in \mathrm{DTraces}(\mathcal{M}^1)$; since $\mathcal{M} \stackrel{c!v\triangleright\{n\}}{\longmapsto} \mathcal{M}^1$, we have that $c!v \triangleright \{n\} :: t \in \mathrm{DTraces}(\mathcal{M})$, and by hypothesis there exists an index $j \in J$ such that $c!v \triangleright \{n\} :: t \in \mathrm{DTraces}(\mathcal{N}_j)$; this is possible only if there exists a network $\mathcal{N}'$ such that $t \in \mathrm{DTraces}(\mathcal{N}')$ and $\mathcal{N}_j \stackrel{c!v\triangleright\{n\}}{\Longmapsto} \mathcal{N}'$. It follows then that $\mathcal{N}' \in \mathscr{N}$, hence $t \in \bigcup_{\mathcal{N}'\in\mathscr{N}} \mathrm{DTraces}(\mathcal{N}')$.

We can now apply the inductive hypothesis to infer that there is a network $\mathcal{N}' \in \mathscr{N}$ such that

$$(\mathcal{N}' \between \mathcal{T}^1) \stackrel{\mu_2}{\Longmapsto} \cdots \stackrel{\mu_k}{\Longmapsto} (\mathcal{N}'_{k-1} \between \mathcal{T}^k)$$

Further, we have already shown that there exists an index $j \in J$ such that $\mathcal{N}_j \stackrel{c!v\triangleright\{n\}}{\Longmapsto} \mathcal{N}'$. It remains to show that $(\mathcal{N}_j \between \mathcal{T}) \stackrel{\tau}{\Longmapsto} (\mathcal{N}' \between \mathcal{T}^1)$. To this end, we first rewrite $(\mathcal{N}_j \between \mathcal{T})$ as $(\mathcal{N}_j \parallel \mathcal{T}_{\mathcal{N}})$, where $\mathcal{T}_{\mathcal{N}} = \mathrm{sym}_{\mathcal{N}}(\mathcal{T})$. Since $\mathcal{T} \stackrel{m.c?v}{\longmapsto} \mathcal{T}^1$, we can employ Lemma 4.2.22 to derive that $\mathcal{T}_{\mathcal{N}} \stackrel{m.c?v}{\longmapsto} \mathcal{T}^1_{\mathcal{N}}$, where $\mathcal{T}^1_{\mathcal{N}} = \mathrm{sym}_{\mathcal{N}}(\mathcal{T}^1)$. It follows from Proposition 4.2.23 (ii) that $(\mathcal{N}_j \parallel \mathcal{T}_{\mathcal{N}}) \stackrel{\tau}{\Longmapsto} (\mathcal{N}' \parallel \mathcal{T}^1_{\mathcal{N}})$. Finally, we can rewrite the latter network as $\mathcal{N}' \between \mathcal{T}^1$; this is a direct consequence of propositions 4.2.4 and 4.2.3.

Appendix B

Proofs of the Propositions in Part II

B.1 Decomposition and composition results

To prove propositions 7.2.10 and 7.2.11, we first need to prove the following statements for actions which can be derived in the intensional semantics:

Proposition B.1.1 (Weakening)**.** *Let $\Gamma_1 \rhd M$ be a network, and let Γ_2 such that whenever $\Gamma_2 \vdash m \leftrightarrow n$ with $n \in nodes(M)$ then $n \notin nodes(M)$. Then*

$$\Gamma_1 \cup M \xrightarrow{\alpha} \Delta \text{ if and only if } (\Gamma_1 \cup \Gamma_2) \rhd M \xrightarrow{\alpha} \Delta$$

where α ranges over the actions $m.\tau, c.m!v, c.m?v$.

Proof. The two implications are proved separately; the only if case is proved by structural induction on the proof of the derivation $\Gamma_1 \rhd M \xrightarrow{\alpha} \Delta$, while the if implication is proved by structural induction on the proof of the derivation $(\Gamma_1 \cup \Gamma_2) \xrightarrow{\alpha} \Delta$; in both cases, the conditions required for the structure of Γ_2 are vital. □

Proposition B.1.2 (Node identification)**.** *Let $\Gamma_M \rhd M$ be a network such that*

1. *$\Gamma_M \rhd M \xrightarrow{m.\tau} \Delta$; then*
   - *$M \equiv m[\![s]\!] \mid M'$,*
   - *$s \xrightarrow{\tau} P$, for some P,*
   - *$\Delta = m[\![\Delta']\!] \mid \overline{M'}$, with $\Delta' = [\![P]\!]$.*
2. *$\Gamma_M \rhd M \xrightarrow{c.m!v} \Delta$, then*
   - *$M \equiv m[\![s]\!] \mid M'$,*
   - *$s \xrightarrow{c!v} P$ for some P,*
   - *$\Gamma_M \rhd M' \xrightarrow{c.m?v} \Theta$ for some Θ,*
   - *$\Delta = m[\![\Delta']\!] \mid \Theta$, with $\Delta' = [\![P]\!]$.*

Proof. Both cases are proved by structural induction on the proof of the derivation $\Gamma_M \rhd M \xrightarrow{\alpha} \Delta$, using Proposition B.1.1 and with α ranging over $m.\tau, c.m!v$. □

Proof of Proposition 7.2.10 We only prove the first statements; details for the other statements are similar.

Suppose $(\Gamma_M \rhd M) \parallel (\Gamma_n \rhd n[\![s]\!]) \xrightarrow{\tau} \Delta$; First we rewrite the network in the left hand side of the transition as $(\Gamma_M \cup \Gamma_n) \rhd M \mid n[\![s]\!]$. By definition of extensional actions, there are two possible cases:

1. $(\Gamma_M \cup \Gamma_n) \rhd Mn[\![s]\!] \xrightarrow{m.\tau} \Delta$; in this case we can apply Proposition B.1.2 (1) to derive $Mn[\![s]\!] \equiv m[\![s']\!] \mid M'$. Here again we have two possible cases:

- $m = n$; then, again by Proposition B.1.2 (1) $M \equiv M' \mathrm{m}\ s \xrightarrow{n.\tau} P_n$, with $[\![P_n]\!] = \Delta_n$ and $\Delta = n[\![\Delta]\!] \,|\, \overline{M}$. Now we can derive $\Gamma_M \rhd n[\![s]\!] \xrightarrow{n.\tau} n[\![\Delta_n]\!]$, and therefore $\Gamma_M \rhd n[\![s]\!] \xrightarrow{\tau} \Gamma_M \rhd n[\![\Delta_n]\!]$.

- $m \neq n$; in this case $m \in \mathrm{nodes}(M)$. By Proposition B.1.2 (1) it holds $M \equiv m[\![s']\!] \,|\, M' \,|\, n[\![s]\!]$, $s' \xrightarrow{\tau} P'$ and $\Delta = m[\![\Delta_m]\!] \,|\, \overline{M'} \,|\, \overline{n[\![s]\!]}$. Let now $\Delta_M = m[\![\Delta_m]\!] \,|\, \overline{M'}$. It is straightforward to show that $\Gamma_M \cup \Gamma_n \rhd M \xrightarrow{m.\tau} \Delta_M$. Note also that, whenever $\Gamma_n \vdash l \leftrightarrow k$ for some $l \in \mathrm{nodes}(M)$, then $k = n$, and $n \notin \mathrm{nodes}(M)$. Thus, we can apply Proposition B.1.1 to derive $\Gamma_M \rhd M \xrightarrow{m.\tau} \Delta_M$, and by definition of extensional actions we obtain $\Gamma_M \rhd M \xrightarrow{\tau} \Delta_M$.

2. $\Gamma_M \cup \Gamma_n \rhd M \,|\, n[\![s]\!] \xrightarrow{c.m!v} \Delta$, with $\{l \mid (\Gamma_M \cup \Gamma_n) \vdash m \leftrightarrow l\} \subseteq \mathrm{nodes}(M \,|\, n[\![s]\!])$. Denote the set in the left hand side of the inclusion above as η_m. Here, by Proposition B.1.2 (2) there are two different possible cases

   - $m = n$. First, note that, as $\eta_n \subseteq \mathrm{nodes}(M)$, whenever $\Gamma_n \vdash m \leftrightarrow n$ then $m \in \mathrm{nodes}(M)$. Now we apply Proposition B.1.2 (2) to obtain $s \xrightarrow{c.n!v} P_n$ and $(\Gamma_1 \cup \Gamma_2) \rhd M \xrightarrow{c.n?v} \Delta_M$, with $\Delta = \Delta_M \,|\, n[\![\Delta_n]\!]$ for $\Delta_n = [\![P_n]\!]$. It is now easy to derive $\Gamma_n \rhd n[\![s]\!] \xrightarrow{c.n!v} \Gamma_n \rhd n[\![s]\!]$; by definition of extentional actions, we obtain $\Gamma_n \rhd n[\![s]\!] \xrightarrow{c!v \rhd \eta} \Gamma_n \rhd s[\![\Delta_n]\!]$, where $\eta = \mathsf{Int}(\Gamma_n \rhd n[\![s]\!]) = \{m \mid \Gamma_n \vdash m \leftrightarrow n\}$. However, we have already noticed that every node in this set is a node in $\mathrm{nodes}(M)$, equivalently $\eta \subseteq \mathrm{nodes}(M)$. It remains to prove $\Gamma_M \rhd M \xrightarrow{c.n?v} \Gamma_M \rhd \Delta_M$. We have already shown that $(\Gamma_1 \cup \Gamma_2) \rhd M \xrightarrow{c.n?v} \Delta_M$, and by Proposition B.1.1 we obtain $\Gamma_M \rhd M \xrightarrow{c.n?v} \Delta_M$. By definition of weak extensional action, it remains to show that $n \in \mathsf{Int}(\Gamma_M \rhd M)$. However, this is ensured, since $\mathsf{Int}(\Gamma_n \rhd n[\![s]\!]) = \eta \subseteq \mathrm{nodes}(M)$ $n \in \mathsf{Int}(\Gamma_M \rhd M)$. Since η is non-empty, there exists at least a node $m \in \mathrm{nodes}(M)$ such that $\Gamma_n \vdash m \leftrightarrow n$, and by the definition of $||$ we obtain $\Gamma_M \vdash mm \leftrightarrow n$.

   - $n \neq m$; this case is similar to the one above, and it is therefore skipped.

□

Proof of Proposition 7.2.11 We prove only (i) and (ii); the other cases ore similar. For (i), Suppose $(\Gamma_M \rhd \Delta) \xRightarrow{\tau} (\Gamma_M \rhd \Delta_M), \Gamma_n \rhd n[\![\Theta]\!] \xRightarrow{\tau} \Gamma_n \rhd n[\![\Theta_n]\!]$, and $(\Gamma_M \rhd \Delta) \,||\, (\Gamma_n \rhd n[\![\Theta]\!])$ is well defined

Now, note that for any node $m \in \mathrm{nodes}(M)$ we have that $\Gamma_n n \leftrightarrow m$ implies $m \neq n$. Thus, whenever for a distribution Δ_0 such that $\mathrm{nodes}(\Delta_0) \subseteq \mathrm{nodes}(\Delta)$ we can derive a (strong) action $\Gamma_M \rhd \Delta_0 \xrightarrow{\tau} \Gamma_M \rhd \Delta_1$, we can apply Propositions B.1.1 and B.1.2 to obtain $(\Gamma_M \cup \Gamma_n)\Delta_0 \,|\, s[\![\Theta]\!] \xrightarrow{\tau} (\Gamma_M \cup \Gamma_n) \rhd \Delta_1 \,|\, n[\![\Theta]\!]$. It is now easy to show that we can infer the hyper-derivation $(\Gamma_M \cup \Gamma_n) \rhd \Delta \xRightarrow{\tau} (\Gamma_M \cup \Gamma_n) \rhd \Delta_M \,|\, n[\![\Theta]\!]$. A similar argument can be used to show that $(\Gamma_M \cup \Gamma_n) \rhd \Delta_M \,|\, n[\![\Theta]\!] \xRightarrow{\tau} (\Gamma_M \cup \Gamma_n) \rhd \Delta_M \,|\, n[\![\Theta_n]\!]$; the result follows now from the transitivity of $\Longrightarrow$, Theorem 6.1.5 (1).

For (ii), suppose $(\Gamma_M \rhd \Delta) \xRightarrow{c!v \rhd \eta} (\Gamma_M \rhd \Delta_M)$, with $n \notin \eta$. Then, we can rewrite Δ as

$$\Delta = \sum_{i \in I} p_i \cdot M_i$$

such that, for all$i \in I$, $\Gamma_M \rhd M_i \xRightarrow{c!v \rhd \eta} \Gamma_M \rhd \Delta_i$ and $\Delta_M = \sum_{iinI} p_i \cdot \Delta_i$. It is sufficient to show that, for every $i \in I$, $\Gamma_M \rhd M_i \,||\, \Gamma_n \rhd n[\![\Theta]\!] \xRightarrow{c!v \rhd \eta} \Gamma_M \rhd \Delta_i \,||\, \Gamma_n \rhd n[\![\Theta]\!]$, thus proving

$$(\Gamma_M \rhd \Delta \,||\, \Gamma_n \rhd n[\![\Theta]\!]) \xRightarrow{c!v \rhd \eta} (\Gamma_M \rhd \Delta_M \,||\, \Gamma_n \rhd n[\![\Theta]\!])$$

The proof for this statement can be done by performing an induction on the proof of the derivation $\Gamma_M \rhd M_i \xRightarrow{c!v \rhd \eta} \Gamma_M \rhd \Delta_i$; for the base case, we have

$$\Gamma_M \rhd M_i \xRightarrow{\tau} \xrightarrow{c!v \rhd \eta} \xRightarrow{\tau} \Gamma_M \rhd \Delta_i$$

By (i) and by Propositions B.1.1 and B.1.2[1], it is easy to derive

$$\Gamma_M \rhd M_i \,||\, \Gamma_n \rhd n[\![\Theta]\!] \stackrel{\tau}{\Longrightarrow} \stackrel{c!v \rhd \eta}{\longrightarrow} \stackrel{\tau}{\Longrightarrow} \Gamma_M \rhd \Delta_i \,||\, \Gamma_n \rhd n[\![\Theta]\!]$$

For the inductive case, $\Gamma_M \rhd M_i \stackrel{c!v \rhd \eta_1}{\Longrightarrow} \stackrel{c!v \rhd \eta_2}{\Longrightarrow} \Gamma_M \rhd \Delta_i$, with $\eta = \eta_1 \cup \eta_2$ in this case it is sufficient to note that $n \notin \eta_1, n \notin \eta_2$ to apply the inductive hypothesis and obtain the result.

[1] Note that it is first necessary to convert an extensional action in an intensional one, in order to apply these theorems

Appendix C

Proofs of the Propositions in Part III

C.1 Properties of the Calculus

Proof of Theorem 9.2.4 We show that if $\Gamma \rhd W \twoheadrightarrow \Gamma' \rhd W'$ and $\Gamma \rhd W$ is well-defined, then so is $\Gamma' \rhd W'$. The proof requires a structural induction on the proof of the reduction above; we only show the most interesting cases, which corresponds to rules (R-NoColl) and (R-Time).

- Rule (R-NoColl): in this case we have that $\Gamma' = \mathrm{upd}^{v}_{c}(\Gamma)$ for some channel c and value v. Suppose that $W' \equiv d[x].P|W'_1$ for some channel $d \neq c$, process P and network W'_1. It follows by definition that $\mathrm{rcv}(d[x].P, c) = \mathsf{false}$. Then it is the case that $W \equiv d[x].P|W_1$; since $\Gamma \rhd W$ is well-formed, it follows that $\Gamma \vdash d : \mathrm{exp}$, and since $(\Gamma')(d) = (\mathrm{upd}^{v}_{c}(\Gamma))(d)$ we also have that $\Gamma' \vdash d : \mathrm{exp}$.

  If $W' \equiv c[x].P|W'_1$ it holds that $\mathrm{upd}^{v}_{c}(\Gamma) \vdash c : \mathrm{exp}$, hence there is nothing to prove.

- Rule (R-Time): in this case, we have that whenever $\Gamma' \rhd W' \equiv \nu l.c[x].P|W'$ then $l = \emptyset$, $\Gamma \vdash_t c : n$ for some $n > 1$ and $\Gamma' = \Gamma \ominus 1$; by definition of $\ominus$, it holds that $\Gamma' \vdash_t c : n-1$, and since $n - 1 > 0$ it follows $\Gamma \vdash c : \mathrm{exp}$.

Proof of Theorem 9.3.1 Let $\Gamma \rhd W, \Gamma_1 \rhd W_1, \Gamma_2 \rhd W_2$ be three configurations such that $\Gamma \rhd W \twoheadrightarrow_{\sigma} \Gamma_1 \rhd W_1$ and $\Gamma \rhd W \twoheadrightarrow_{\sigma} \Gamma_2 \rhd W_2$.

We show that $\Gamma_1 \rhd W_1 \equiv \Gamma_2 \rhd W_2$ by performing a structural induction on the proof of the reduction $\Gamma \rhd W \twoheadrightarrow_{\sigma} \Gamma_1 \rhd W_1$.

- The last rule applied in the proof of the derivation is Rule (R-Time). In this case there exists five index sets J, R, K, L, S and three sets of channels $\{c_{rj}\}_{r\in R, j\in J}$, $\{c_k\}_{k\in K}$, $\{c_l\}_{l\in L}$, a set of closed values $\{v_l\}_{l\in L}$ and collection of processes $\{P_{rj}\}_{r\in R, j\in J}$, $\{Q_{rj}\}_{r\in R, j\in J}$, $\{P_{sj}\}_{s\in S, j\in J}$, $\{P_k\}_{k\in K}$, $\{P_l\}_{l\in L}$ such that

  1. $\Gamma \vdash c_{rj}$: free for all $r \in R, j \in J$,
  2. For any $k \in K$, $\Gamma(c_k) = (n, v)$ for some value v and $n > 1$,
  3. For all $l \in L$, $\Gamma \vdash l$ deliver v_l
  4. $\Gamma_1 = \Gamma \ominus 1$
  5. $W \equiv$
     $prod_{j\in J}\left(\sum_{r\in R} \lfloor c_{rj}?(P_{rj}).Q_{rj} \rfloor + \sum_{s\in S} \sigma.P_{sj}\right) | \prod_{k\in K} c_k[x].P_k | \prod l \in L c_l[x]..P_l$
  6. $W_1 = \prod_{j\in J}\left(\sum_{r\in R} Q_{rj} + \sum_{s\in S} \sigma.P_{sj}\right) | \prod_{k\in K} c[x].P_k \prod |\{^{v_l}/_x\}P_l$

  In order to prove that $\Gamma_1 \rhd W_1 \equiv \Gamma_2 \rhd W_2$ we need to perform an inner structural induction on the proof of $\Gamma \rhd W \twoheadrightarrow_{\sigma} \Gamma_2 \rhd W_2$. If the last rule applied is (R-Time), then it is trivial to prove that $\Gamma_1 = \Gamma_2, W_1 = W_2$,

  The only other possible case is that in which the last Rule applied in the proof of the reduction $\Gamma \rhd W \twoheadrightarrow_{\sigma} \Gamma_2 \rhd W_2$ is Rule (R-Struct); note in fact that $\Gamma \rhd W$ has no restricted channels, so that it would not be

possible to apply Rule (R-RSTRTMD) in the proof of the derivation above. In this case we have that there exists a network W_2' such that $W_2 \equiv W_2'$ and $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma_2 \rhd W_2'$; by inductive hypothesis it holds that $\Gamma_1 \rhd W_1 \equiv \Gamma_2 \rhd W_2'$, hence $\Gamma_1 \rhd W_1 \equiv \Gamma_2 \rhd W_2$.

- The last rule applied in the proof of the reduction $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma_1 \rhd W_1$ is Rule (R-RSTSTMD). In this case we have that $W = \nu c : (n, v)W'$, $W_1 = \nu c : (n_1, v_1).W_1'$ and $\Gamma[c \mapsto (n, v)] \rhd W \twoheadrightarrow_\sigma \Gamma_1' \rhd W_1'$, where $\Gamma_1' = \Gamma_1[c \mapsto (n_1, v_1)]$.

  It is easy now to note that the last rule applied in the proof of the derivation $\Gamma \rhd W \twoheadrightarrow_\sigma \Gamma_2 \rhd W_2$ is Rule (R-RSTRTMD) and and $W_2 = \nu c : (n_1, v_1).W_2'$ for some network W_2' such that $\Gamma[c \mapsto (n, v)] \rhd W \twoheadrightarrow_\sigma \Gamma_2' \rhd W_2'$, where $\Gamma_2' = \Gamma_1[c \mapsto (n_1, v_1)]$. By inductive hypothesis it follows that $W_1' \equiv W_2'$, hence $W_1 = \nu c : (n, v).W_1' \equiv \nu c : (n, v).W_2' = W_2$.

  It remains to show that $\Gamma_1 = \Gamma_2$; this is trivial, for we have $\Gamma_1 = \Gamma \ominus 1 = \Gamma_2$.

- The last case left to analyse is similar to that of Rule (R-STRUCT); this is similar to the other two cases, and it is therefore omitted.

Proof of Theorem 9.3.2 Let $\Gamma \rhd W, \Gamma_1 \rhd W_1$ be two configurations such that $\Gamma \rhd W \twoheadrightarrow_u \Gamma_1 \rhd W_1$; we show, by structural induction on the proof of the reduction above, that $\Gamma \rhd W$ is structurally congruent to a configuration for which none of the rules (R-TIME), (B-RSTRTMD) can be applied. We only provide some of the details needed to perform the proof.

- The last rule applied in the proof of the reduction above is Rule (R-INTERNAL); in this case we have that $W \equiv \tau.P + Q | W'$; now it suffices to note that Rule (B-TIME) cannot be applied to $\Gamma \rhd W$, for in order for the latter to be applied there should be no parallel component structurally congruent to process of the form $\tau.P + Q$ in W. Rule (B-()*rstrtmd*) also cannot be applied, for it is required that $W \equiv \nu c : (n, v).W''$, which is not the case.

- In the case the last rule applied is either (B-NOCOLL) or (B-COLL) it suffices to note that $W \equiv c\,!\langle v \rangle.P + Q | W'$, then proceeding as in the previous case.

- If the last rule applied is Rule (B-LATEWAKEUP) then $W \equiv \lfloor c?(x).P \rfloor Q + R | W$ for some channel c such that $\Gamma \vdash c : \mathrm{exp}$. In this case it suffices to note that the configuration above has no restricted channel, so that Rule (R-RSTRTMD) cannot be applied to the configuration $\Gamma \rhd W$; further, rule (R-TIME) cannot be applied either to the latter, for the premises of such a rule require that whenever $W \equiv \lfloor c?(x).P \rfloor Q + R | W$ then $\Gamma \vdash c : \mathrm{free}$.

Proof of Lemma 9.4.1(3) Suppose that $\Gamma \rhd W \xrightarrow{\tau} \Gamma' \rhd W'$; we need to show that $\Gamma = \Gamma'$; to this end, we perform a rule induction on the proof of the transition above. We only show some of the details.

- The last rule applied is Rule (B-RCVLATE); then $\Gamma \vdash c : \mathrm{exp}$ and $\Gamma \rhd W = \Gamma \rhd \lfloor c?(x).P \rfloor Q$ and $\Gamma' \rhd W' = \Gamma \rhd c[x].\{^{\mathsf{err}}/_x\}P$. The latter equation establishes that $\Gamma = \Gamma'$.

- The last rule applied is Rule (B-τ); in this case $\Gamma \rhd W = \Gamma \rhd \tau.P$ and $\Gamma' \rhd W' = \Gamma \rhd P$, hence $\Gamma = \Gamma'$.

- The last rule applied is Rule (B-TAUPAR). We have that $\Gamma \rhd W = \Gamma \rhd W_1 | W_2$, $\Gamma \rhd W_1 \xrightarrow{\tau} \Gamma' \rhd W_1'$ and $\Gamma' \rhd W' = \Gamma' \rhd W_1' | W_2$; we can apply the inductive hypothesis to the transition $\Gamma \rhd W_1 \xrightarrow{\tau} \Gamma' \rhd W_1'$ to infer that $\Gamma = \Gamma'$.

- The last rule applied is Rule (B-RESI); In this case we have that $W = \nu n : (c, v).W'$, $\Gamma[c \mapsto (n, v)] \rhd W \xrightarrow{c!v} \Gamma'' \rhd W''$ and $\Gamma' \rhd W' = \Gamma''[c \mapsto \Gamma(c)] \rhd W''$. By Proposition 9.4.1(1) it holds that $\Gamma'' = \mathrm{upd}_c^v(\Gamma[c \mapsto (n, v)])$; therefore we have that $\Gamma' = (\Gamma[c \mapsto (n, v)])[c \mapsto \Gamma(c)] = \Gamma[c \mapsto \Gamma(c)] = \Gamma$.

Proof of Proposition 9.5.1 Let W be a term such that $c \notin \mathrm{fn}(W)$, and suppose $\Gamma \rhd W \xrightarrow{\lambda} \Gamma' \rhd W'$, where $\lambda \neq \sigma, c?v$. We show that $\Gamma[c \mapsto (n,v)] \rhd W \xrightarrow{\lambda} \Gamma'[c \mapsto (n,v)] \rhd W'$ by performing a rule induction on the proof of the former transition. We only show some of the possible cases, and we leave all the details to the interested reader.

- The last rule applied is (B-SND); then $W = !\langle d\rangle.wP$ and $\lambda = d!w$ for some channel $d \neq c$ (recall that we are assuming that $c \notin \mathrm{fn}(W)$) and value w. It follows that $\Gamma' = \mathrm{upd}_d^w(\Gamma)$ and $W' = d^{\delta_v}.P$. Now we can apply Rule (B-SND) to the configuration $\Gamma[c \mapsto (n,v)] \rhd W$ to infer the transition $\Gamma[c \mapsto (n,v)] \rhd d\,!\langle w\rangle.P \xrightarrow{d!w} \mathrm{upd}_d^w(\Gamma[c \mapsto (n,v)]) \rhd d^{\delta_v}.P$.

  It remains to note that $\mathrm{upd}_d^w(\Gamma[c \mapsto (n,v)]) = \mathrm{upd}_d^w(\Gamma)[c \mapsto (n,v)] = \Gamma'[c \mapsto (n,v)]$.

- The last rule applied is (B-EXPTHEN). In this case $W = \langle d\rangle P, Q$ for some channel c such that $\Gamma \vdash d : \mathrm{exp}$, $\lambda = \tau$ and $\Gamma' \rhd W' = \Gamma \rhd \sigma.P$.

  For $c \notin \mathrm{fn}(W)$ it follows that $d \neq c$, hence $\Gamma[c \mapsto (n,v)] \vdash d\mathrm{exp}$. By an application of Rule (B-EXPTHEN) it follows that $\Gamma[c \mapsto (n,v)] \rhd \langle d\rangle P, Q \xrightarrow{\tau} \Gamma[c \mapsto (n,v)] \rhd \sigma.P$; the result follows by noting that $\Gamma = \Gamma'$, hence $\Gamma[c \mapsto (n,v)] = \Gamma'[c \mapsto (n,v)]$.

- The last rule applied is Rule (B-RESI). In this case it follows that $W = \nu d : (n',v').W_1$, $\lambda = \tau$ and $\Gamma[d \mapsto (n',v')] \rhd W_1 \xrightarrow{d!w} \Gamma'' \rhd W_1'$ for some Γ'', W_1' such that $\Gamma' = \Gamma''[d \mapsto \Gamma(d)]$ and $W_1 = \nu d : \Gamma''(d).W_1'$. We have two possible cases:

  1. $d = c$; consider the configuration $\Gamma[d \mapsto (n,v)] \rhd W_1$. It is easy to show that $(\Gamma[d \mapsto (n,v)])[d \mapsto (n',v')] = \Gamma[d \mapsto (n',v')]$. By applying Rule (B-RESI) to the transition $\Gamma[d \mapsto (n',v')] \rhd W_1 \xrightarrow{d!w} \Gamma'' \rhd W_1'$ we obtain that $(\Gamma[d \mapsto (n,v)]) \rhd \nu d : (n',v').W_1 \xrightarrow{\tau} \Gamma''[d \mapsto (\Gamma[d \mapsto (n,v)])(d)] \rhd \nu d : \Gamma''(d).W_1'$.
     Now it remains to note that

     $$\Gamma''[d \mapsto (\Gamma[d \mapsto (n,v)])(d)] = \Gamma''[d \mapsto (n,v)]$$

     and, since $\Gamma' = \Gamma''[d \mapsto \Gamma(d)]$, we have also

     $$\Gamma'[d \mapsto (n,v)] = (\Gamma''[d \mapsto \Gamma(d)])[d \mapsto (n,v)] = \Gamma'[d \mapsto (n,v)]$$

     Therefore we have that

     $$\Gamma''[d \mapsto (\Gamma[d \mapsto (n,v)])(d)] = \Gamma'[d \mapsto (n,v)]$$

     so that the transition $(\Gamma[d \mapsto (n,v)]) \rhd \nu d : (n',v').W_1 \xrightarrow{\tau} \Gamma''[d \mapsto (\Gamma[d \mapsto (n,v)])(d)] \rhd \nu d : \Gamma''(d).W_1'$ can be rewritten as $(\Gamma[d \mapsto (n,v)]) \rhd \nu d : (n',v').W_1 \xrightarrow{\tau} \Gamma'[d \mapsto (n,v)] \rhd \nu d : \Gamma''(d).W_1'$, as we wanted to prove.

  2. $d \neq c$; in this case we have that $c \notin \mathrm{fn}(W_1)$, hence by inductive hypothesis $(\Gamma[d \mapsto (n',v')])[c \mapsto (n,v)] \rhd \xrightarrow{d!w} \Gamma''[c \mapsto (n,v)] \rhd W_1'$. Since $d \neq c$, we can rewrite the channel environment $(\Gamma[d \mapsto (n',v')])[c \mapsto (n,v)]$ as $(\Gamma[c \mapsto (n,v)])[d \mapsto (n',v')]$ in the transition above, leading to

     $$(\Gamma[c \mapsto (n,v)])[d \mapsto (n',v')] \rhd W_1 \xrightarrow{d!w} \Gamma''[c \mapsto (n,v)] \rhd W_1'$$

     By Applying Rule (B-RESI) to the last transition we obtain that

     $$\Gamma[c \mapsto (n,v)] \rhd \nu d : (n',v') \xrightarrow{\tau} (\Gamma''[c \mapsto (n,v)])[d \mapsto (\Gamma[c \mapsto (n,v)])(d)] \rhd \nu d : (\Gamma''[c \mapsto (n,v)])(d).W_1'$$

By performing the required map, we find that $(\Gamma''[c \mapsto (n,v)])(d) = \Gamma''(d)$, while

$$\begin{aligned}(\Gamma''[c \mapsto (n,v)])[d \mapsto (\Gamma[c \mapsto (n,v)])(d)] &= (\Gamma''[c \mapsto (n,v)][d \mapsto \Gamma(d)]) \\ &= \Gamma''[d \mapsto \Gamma(d)][c \mapsto (n,v)] \\ &= \Gamma'[c \mapsto (n,v)]\end{aligned}$$

when the last equation can be proved by recalling that $\Gamma' = \Gamma''[d \mapsto \Gamma(d)]$.

By performing the appropriate substitusions we can rewrite the transition we have derived as $\Gamma[c \mapsto (n,v)] \rhd \nu d : (n',v')W_1 \xrightarrow{\tau} \Gamma'[c \mapsto (n,v)] \rhd \nu d : \Gamma''(d).W_1'$, which is exactly what we wanted to prove.

Proof of Proposition 9.5.5 The proof is carried out by Rule induction on the proof of the derivation $\Gamma \rhd W \xrightarrow{c!} \Gamma' \rhd W'$. We provide the details only for the case in which Rule (B-SYNC) has been applied, and $\Gamma(c)_{\mathbf{1}} = 0$.

In this case we have that $W = S|R$ for some S, R and value w such that $\Gamma \rhd S \xrightarrow{c!} \Gamma \rhd S'$ and $\Gamma \rhd S' \xrightarrow{c?w} \Gamma \rhd R'$, with $W' = S'|R'$. Thus we can apply the inductive hypothesis to the first derivation, and we assume that $\Gamma \rhd S \equiv \Gamma \rhd \nu l'.(c\,!\langle v\rangle.P + Q|S_1)$ for some l', P, Q, S_1 and S_1' such that the following is true:

- c does not appear in l'
- $\Gamma \rhd \nu l'.S_1 \xrightarrow{c?v} \Gamma' \rhd \nu l'.S_1'$
- $\Gamma' \rhd S' \equiv \Gamma' \rhd \nu l'.(\sigma^{\delta_v}.P|S_1')$.

Here note that the value v being received by $\Gamma \rhd S_1$ is not necessarily the same received by $\Gamma \rhd R$.

Without loss of generality, we can also assume that none of the variable in $\mathrm{ch}(l')$ appear free in R, as it is always possible to perform an α-conversion.

First, we show that $\Gamma \rhd R \xrightarrow{c?v} \Gamma' \rhd R'$. This is a direct consequence of Propositions 9.4.2 and 9.4.1(3). In fact, by applying Proposition 9.4.1(2) to the transition $\Gamma \rhd \nu l'.S \xrightarrow{c?v} \Gamma' \rhd \nu l'.S'$ we obtain $\Gamma' = \mathrm{upd}_c^v(\Gamma)$. Now, by Proposition 9.4.2 applied to the transition $\Gamma \rhd R \xrightarrow{c?w} \Gamma' \rhd R'$ there exists a channel environment Γ_v such that $\Gamma \rhd R \xrightarrow{c?v} \Gamma_v \rhd R'$. Finally, we can apply again Proposition 9.4.1(2) to the latter to obtain $\Gamma_v = \mathrm{upd}_c^v(\Gamma) = \Gamma'$, hence $\Gamma \rhd R \xrightarrow{c?v} \Gamma' \rhd R'$.

We are now ready to prove the statements; we need to show that $\Gamma \rhd W = \Gamma \rhd (S|R)$ is congruent to a configuration whose form is $\Gamma \rhd \nu l.(c\,!\langle v'\rangle.P' + Q'|W_1)$ for some l, v', P, Q, W_1, W_1' such that conditions (1-3) of the Proposition are satisfied. To this end, let $l = l'$, $P' = P$, $Q' = Q$, $W_1 = S_1|R$, $W_1' = S_1'|R'$.

We now show that $\Gamma \rhd W \equiv \Gamma \rhd \nu l.(c\,!\langle v\rangle.PQ|(S_1|R))$; but this follows directly from the assumptions $W = (S|R)$, $S \equiv \Gamma \rhd \nu l.(c\,!\langle v\rangle.PQ|S_1)$ and the constraints satisfied by the congruence relation $\equiv$, as none of the values in l appear free in R.

Also, from the inductive hypothesis, it follows that c does not appear in l, so that Requirement (1) is met.

Let us turn our attention to Requirement (2); we need to show that $\Gamma \rhd \nu l.(S_1|R) \xrightarrow{c?v} \Gamma' \rhd \nu l.(S_1'|R')$. This can be proved as follows: As $\Gamma \rhd R \xrightarrow{c?v} \Gamma' \rhd R'$, and none of the names in $\mathrm{ch}(c)$ appear in R, a repeated application of Proposition 9.5.1 leads to the derivation $\Gamma[l] \rhd R \xrightarrow{c?v} \Gamma'[l] \rhd R'$. Further, by Corollary 9.5.4, we can remove the channel restrictions from the transition $\Gamma \rhd \nu l.S_1 \xrightarrow{c?v} \Gamma' \rhd \nu l.S_1'$, thus showing $\Gamma[l] \rhd S_1 \xrightarrow{c?v} \Gamma'[l] \rhd S_1'$. Now it is sufficient to apply Rule (B-RCVPAR) to infer $\Gamma[l] \rhd S_1|R \xrightarrow{c?v} \Gamma'[l] \rhd S_1'|R'$, and by a repeated application of Rule (B-RCV − UNRSTR) (recall that c does not appear in $\mathrm{ch}(l)$), we obtain that Requirement (2) is met.

Finally, we need to show that Requirement (3) is met. That is, $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l.(\sigma^{\delta_v}.P|(S_1'|R'))$. Note that in this case we have $\Gamma' \rhd W' \equiv \Gamma' \rhd \nu l.(\sigma^{\delta_v}.P|S_1')|R'$. As none of the free names in R appear free in $\mathrm{ch}(l)$, the same holds for R', as $\Gamma' \rhd R'$ is a derivative of $\Gamma \rhd R$; again, the above congruence follows from the definition of the congruence relation, and Requirement (3) is met.

C.2 Barbed Equivalence and Weak Bisimulation

Proof of Theorem 10.1.4 Let $\Gamma \rhd W$ be a configuration such that $\Gamma \rhd W \Downarrow_c$. By definition we have that $\Gamma \rhd W \rightarrow^* \Gamma' \rhd W' \downarrow_c$ or, more specifically, $\Gamma \rhd W \rightarrow^n \Gamma' \rhd W' \downarrow_c$ for some $n \geq 0$. Recall that $\rightarrow^n$ denotes a sequence of n reductions. We show that $\Gamma \rhd W|T \rightarrow^* \Gamma' \rhd W'$ by induction on n. Note that, since we are assuming that eureka, fail are fresh channels it holds that $\Gamma \vdash \text{eureka} : \text{free}$, $\Gamma \vdash \text{fail} : \text{free}$.

- Base case: $n = 0$. In this case $\Gamma \rhd W = \Gamma' \rhd W'$, hence $\Gamma \rhd W \downarrow_c$, which implies $\Gamma \vdash c : \text{exp}$. In this case it is easy to show that

$$\Gamma \rhd W|T \rightarrow_u \Gamma \rhd W|T_1 \rightarrow_\sigma^* \Gamma_1 \rhd W_1|T_2 \rightarrow_u \Gamma_2 \rhd W_1|T_3$$

  where

$$\begin{array}{rcl} T_1 & = & \sigma.\text{eureka}!\langle \text{ok} \rangle \\ T_2 & = & \text{eureka}!\langle \text{ok} \rangle \\ T_3 & = & \sigma.\mathsf{nil} \\ \Gamma \rhd W & \rightarrow_\sigma^* & \Gamma_1 \rhd W_1 \\ \Gamma_2 & \vdash & \text{fail} : \text{free} \\ \Gamma_2 & \vdash & \text{eureka} : \text{exp} \end{array}$$

  Note that we can infer the weak timed reduction $\Gamma \rhd W \rightarrow_\sigma^* \Gamma_1 \rhd W_1$ for we are assuming that $\Gamma \rhd W$ is well-timed. We have proved that $\Gamma \rhd W|T \rightarrow^* \Gamma_2 \rhd W_1|T_3$ with $\Gamma_2 \vdash \text{fail} : \text{free}, \Gamma_2 \vdash \text{eureka} : \text{exp}$.

  Since $\text{fail} \notin \text{fn}(T_3)$ and $\text{fail} \notin \text{fn}(W)$ (from which $\text{fail} \notin \text{fn}(W_2)$ follows) it is straightforward to note that $\Gamma_2 \rhd W_1|T_3 \not\Downarrow_{\text{fail}}$. Further, since $\Gamma_2 \vdash \text{eureka} : \text{exp}$ we also have $\Gamma_2 \rhd W_1|T_3 \downarrow_{\text{eureka}}$, as we wanted to prove.

- Suppose now $n > 0$, and assume the statements holds for $n - 1$ If $\Gamma \vdash c : \text{exp}$ we can proceed as in the previous case, otherwise we have that there exists a configuration $\Gamma_1 \rhd W_1$ such that $\Gamma_1 \rhd W_1$ for some $\Gamma_1 \rhd W_1$ and $\Gamma_1 \rhd W_1 \rightarrow^{n-1} \Gamma' \rhd W'$. By inductive hypothesis it holds that $\Gamma_1 \rhd W_1|T \rightarrow^* C$ for some configuration C such that $C \downarrow_{\text{eureka}}, C \not\Downarrow_{\text{fail}}$.

  We have to consider two different cases, according to the nature of the reduction $\Gamma \rhd W \rightarrow \Gamma_1 \rhd W_1$.

  - $\Gamma \rhd W \rightarrow_u \Gamma_1 \rhd W_1$. In this case it is easy to show that $\Gamma \rhd W|T \rightarrow_u \Gamma_1 \rhd W_1|T$, hence $\Gamma \rhd W|T \rightarrow^* C$; at this point there is nothing left to prove.

  - $\Gamma \rhd W \rightarrow_\sigma \Gamma_1 \rhd W_1$; in this case we can build the sequence of transitions

$$\Gamma \rhd W|T \rightarrow_u \Gamma \rhd W|T_4 \rightarrow_\sigma \Gamma_1 \rhd W_1|T_5 \rightarrow_u \Gamma_1 \rhd W_1|T$$

    where $T_4 = \sigma.(\text{fail}!\langle \text{no} \rangle + \tau.T)$ and $T_5 = \text{fail}!\langle \text{no} \rangle + \tau.T$. We have proved that $\Gamma \rhd W|T \rightarrow_\sigma^* \Gamma_1 \rhd W_1|T$, hence $\Gamma \rhd W|T \rightarrow^* C$. Since $C \downarrow_{\text{eureka}}, C \not\Downarrow_{\text{fail}}$ there is nothing left to prove.

Now suppose that $\Gamma \rhd W|T \rightarrow^* C$ for some configuration C such that $C \downarrow_{\text{eureka}}$, $C \not\Downarrow_{\text{fail}}$. In order to prove that $\Gamma \rhd W \Downarrow_c$ it is convenient to introduce the notation $T_6 = \text{fail}!\langle \text{no} \rangle$, $T_7 = \mathsf{nil}$.

First, note that whenever $\Gamma \rhd W|T \rightarrowtriangle^* C'$ for some configuration C' then the latter is structurally equivalent to $\Gamma \rhd W'|T'$ for some channel environment Γ' and system term T' such that either

$$\begin{array}{lcllll}
T' & = & T & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free} \\
T' & = & T_1 & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free} \\
T' & = & T_2 & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free} \\
T' & = & T_3 & \text{and} & \Gamma' \vdash \text{eureka} : \text{exp} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free} \\
T' & = & T_3 & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{exp} \\
T' & = & T_4 & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free} \\
T' & = & T_5 & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free} \\
T' & = & T_6 & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free} \\
T' & = & T_7 & \text{and} & \Gamma' \vdash \text{eureka} : \text{free} & \text{and} \quad \Gamma' \vdash \mathit{fail} : \text{free}
\end{array}$$

Note that the only possibility for $\Gamma' \rhd W'|T'$ to have a strong barb on channel eureka is the fourth case. Further, recall that $T_3 = \sigma.\mathsf{nil}$; since $\text{fail} \notin \text{fn}(T_3)$ and $\text{fail} \notin \text{fn}(W')$, it follows that $\Gamma' \rhd W'|T_3 \not\Downarrow_{\text{fail}}$.

We also note that the weak reduction $\Gamma \rhd W|T \rightarrowtriangle^* \Gamma' \rhd W'$ can be obtained only via a sequence of (both weak and strong) reductions of the form

$$\begin{array}{c}
\Gamma \rhd W|T \rightarrowtriangle_u^* \Gamma_1 \rhd W_1|T \rightarrowtriangle^* \Gamma_1' \rhd W_1'|T \rightarrowtriangle_u \Gamma_1'' \rhd W_1|T_1 \rightarrowtriangle_u^* \Gamma_2 \rhd W_2|T_1 \rightarrowtriangle_\sigma \\
\rightarrowtriangle_\sigma \Gamma_2' \rhd W_2|T_2 \rightarrowtriangle_u^* \Gamma_3 \rhd W_3|T_2 \rightarrowtriangle_u \Gamma_3' \rhd W_3|T_3 \rightarrowtriangle^* \Gamma_3'' \rhd W_3'|T_7
\end{array}$$

where Γ_1' is a channel environment such that $\Gamma_1' \vdash c : \text{exp}$ (note that we can reach the configuration T_1 from T only if the exposure check on channel c returns a positive outcome). Since the only value T can broadcast is along channel $\text{fail}!\langle \text{no} \rangle$, which does not appear free in W, it is possible to infer the weak reduction $\Gamma \rhd W \rightarrowtriangle_u^* \Gamma_1' \rhd W_1'$ from $\Gamma \rhd W|T \rightarrowtriangle^* \Gamma_1 \rhd W_1'|T$. Not it is trivial to note that $\Gamma_1' \rhd W_1' \downarrow c$, hence $\Gamma \rhd W \Downarrow c$.

Bibliography

[1] Aceto, L., Ingólfsdóttir, A., Larsen, K. G., and Srba, J. Reactive systems: Modelling, specification and verification, cambridge university press (2007) ISBN 9780521875462. *J. Log. Algebr. Program 78*, 1 (2008), 52.

[2] Attiya, H., and Welch, J. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.

[3] Baier, C., and Katoen, J.-P. *Principles of model checking*. MIT Press, 2008.

[4] Banerjee, S., Bhattacharjee, B., and Kommareddy, C. Scalable application layer multicast. In *SIGCOMM* (2002), ACM, pp. 205–217.

[5] Blizard, W. D. Multiset theory. *Notre Dame Journal of Formal Logic 30*, 1 (Winter 1989), 36–66.

[6] Bracha, and Toueg. Asynchronous consensus and broadcast protocols. *JACM: Journal of the ACM 32* (1985).

[7] Cenciarelli, P., Gorla, D., and Salvo, I. Depletable channels: Dynamics and behaviour. In *FCT* (2009), M. Kutylowski, W. Charatonik, and M. Gebala, Eds., vol. 5699 of *Lecture Notes in Computer Science*, Springer, pp. 50–61.

[8] Cerone, A., and Hennessy, M. Modelling probabilistic wireless networks (extended abstract). In *Proceedings of the 14th Annual International Conference on Formal Methods for Open Object-Based Distributed System FMOODS 2012, and the 32th Annual International Conference on Formal Techniques for Networked and Distributed Systems FORTE 2012, Stockholm, Sweden, June 13-16 2012* (2012), H. Giese and G. Rosu, Eds., Springer.

[9] Cerone, A., and Hennessy, M. A simple probabilistic broadcast language. Tech. Rep. TCD-CS-2012-02, Trinity College Dublin, 2012.

[10] Cerone, A., Hennessy, M., and Merro, M. Modelling collisions in broadcast calculi. Technical Report, Trinity College Dublin, Universita' di Verona, In Preparation.

[11] Cheriton, D. R., and Deering, S. E. Multicast routing in datagram internetworks and extended LANs. *ACM Trans. on Computing Sys. 8*, 2 (May 1990), 85.

[12] Choi, J. I., Jain, M., Srinivasan, K., Levis, P., and Katti, S. Achieving single channel, full duplex wireless communication. In *MOBICOM* (2010), N. H. Vaidya, S. Banerjee, and D. Katabi, Eds., ACM, pp. 1–12.

[13] Cleaveland, Dayar, Smolka, and Yuen. Testing preorders for probabilistic processes. *INFCTRL: Information and Computation (formerly Information and Control) 154* (1999).

[14] Curran, E., and Dowling, J. Sample: Statistical network link modelling in an on-demand probabilistic routing protocol for ad hoc networks. In *WONS* (2005), IEEE Computer Society, pp. 200–205.

[15] Davey, B. A., and Priestley, H. A. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[16] Davide Sangiorgi. *An Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011.

[17] de Nicola, R., and Hennessy, M. Testing equivalences for processes. *Theor. Comput. Sci 34* (1984), 83–133.

[18] Deng, Y., and Hennessy, M. On the semantics of markov automata. In *ICALP (2)* (2011), L. Aceto, M. Henzinger, and J. Sgall, Eds., vol. 6756 of *Lecture Notes in Computer Science*, Springer, pp. 307–318.

[19] Deng, Y., van Glabbeek, R., Hennessy, M., and Morgan, C. Testing finitary probabilistic processes. In *Proceedings of the 20th International Conference on Concurrency Theory* (2009), vol. 5710 of *Lecture Notes in Computer Science*, Springer, pp. 274–288. Full-version available from http://www.scss.tcd.ie/Matthew.Hennessy/onlinepubs.html.

[20] Deng, Y., van Glabbeek, R., Hennessy, M., and Morgan, C. Testing finitary probabilistic processes (extended abstract). In *Proceedings of the 20th International Conference on Concurrency Theory* (2009), vol. 5710 of *Lecture Notes in Computer Science*, Springer, pp. 274–288.

[21] Deng, Y., and van Glabbeek, R. J. Characterising probabilistic processes logically. *CoRR abs/1007.5188* (2010).

[22] Deng, Y., van Glabbeek, R. J., Morgan, C., and Zhang, C. Scalar outcomes suffice for finitary probabilistic testing. In *ESOP* (2007), R. D. Nicola, Ed., vol. 4421 of *Lecture Notes in Computer Science*, Springer, pp. 363–378.

[23] Ene, C., and Muntean, T. A broadcast-based calculus for communicating systems. In *IPDPS* (2001), IEEE Computer Society, p. 149.

[24] Ene, C., and Muntean, T. Testing theories for broadcasting processes. *Sci. Ann. Cuza Univ 11* (2002), 214–230.

[25] Francis, P. Yoid: Extending the internet multicast architecture. 2000.

[26] Ghassemi, F., Fokkink, W., and Movaghar, A. Restricted broadcast process theory. In *SEFM* (2008), A. Cerone and S. Gruner, Eds., IEEE Computer Society, pp. 345–354.

[27] Ghassemi, F., Fokkink, W., and Movaghar, A. Equational reasoning on mobile ad hoc networks. *Fundam. Inform 105*, 4 (2010), 375–415.

[28] Ghassemi, F., Fokkink, W., and Movaghar, A. Verification of mobile ad hoc networks: An algebraic approach. *Theor. Comput. Sci 412*, 28 (2011), 3262–3282.

[29] Ghassemi, F., Talebi, M., Movaghar, A., and Fokkink, W. Stochastic restricted broadcast process theory. In *EPEW* (2011), N. Thomas, Ed., vol. 6977 of *Lecture Notes in Computer Science*, Springer, pp. 72–86.

[30] Godskesen, J. C. A calculus for mobile ad hoc networks. In *COORDINATION* (2007), A. L. Murphy and J. Vitek, Eds., vol. 4467 of *Lecture Notes in Computer Science*, Springer, pp. 132–150.

[31] Halonen, T., Melero, J., and Garcia, J. R. *GSM, GPRS and EDGE Performance: Evolution Toward 3G/UMTS*. Halsted Press, New York, NY, USA, 2002.

[32] Hennessy, and Rathke. Bisimulations for a calculus of broadcasting systems. *TCS: Theoretical Computer Science 200* (1998).

[33] Hennessy, M. *Algebraic Theory of Processes*. MIT Press, 1988.

[34] Hennessy, M. *A distributed Pi-calculus*. Cambridge University Press, 2007.

[35] Hennessy, M. Exploring probabilistic bisimulation (part i). Tech. Rep. TCD-CS-2012-01, Trinity College Dublin, 2012.

[36] Hennessy, M., and Regan, T. A process algebra for timed systems. *Information and Computation 117*, 2 (Mar. 1995), 221–239.

[37] H.P. Barendregt. *The λ-calculus*. North-Holland, 1984.

[38] Jonsson, and Yi. Testing preorders for probabilistic processes can be characterized by simulations. *TCS: Theoretical Computer Science 282* (2002).

[39] Jonsson, B., Ho-Stuart, C., and Yi, W. Testing and refinement for nondeterministic and probabilistic processes. *Lecture Notes in Computer Science 863* (1994), 418–??

[40] Jurdak, R., Lopes, C. V., and Baldi, P. A survey, classification and comparative analysis of medium access control protocols for ad hoc networks. *IEEE Communications Surveys and Tutorials 6*, 1-4 (2004), 2–16.

[41] Kaynar, D. K., Lynch, N. A., Segala, R., and Vaandrager, F. W. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan & Claypool Publishers, 2006.

[42] Kleene, S. C. λ-definability and recursiveness. *Duke Mathematical Journal 2* (1936), 340–353.

[43] Lanese, I., and Sangiorgi, D. An operational semantics for a calculus for wireless systems. *Theor. Comput. Sci 411*, 19 (2010), 1928–1948.

[44] Lanotte, R., and Merro, M. Semantic analysis of gossip protocols for wireless sensor networks. In *CONCUR* (2011), J.-P. Katoen and B. König, Eds., vol. 6901 of *Lecture Notes in Computer Science*, Springer, pp. 156–170.

[45] Liu, J. C., Rao, S. G., Li, B., and Zhang, H. Opportunities and challenges of peer-to-peer internet video broadcast. *Proceedings of IEEE 96*, 1 (Jan. 2008), 11–24.

[46] Merro, M. An observational theory for mobile ad hoc networks (full version). *Inf. Comput 207*, 2 (2009), 194–208.

[47] Merro, M., and Sibilio, E. A timed calculus for wireless systems. In *FSEN* (2009), F. Arbab and M. Sirjani, Eds., vol. 5961 of *Lecture Notes in Computer Science*, Springer, pp. 228–243.

[48] Mezzetti, N., and Sangiorgi, D. Towards a calculus for wireless systems. *Electr. Notes Theor. Comput. Sci 158* (2006), 331–353.

[49] Milner, and Sangiorgi. Barbed bisimulation. In *ICALP: Annual International Colloquium on Automata, Languages and Programming* (1992).

[50] Milner, R. A calculus of communicating systems. *LNCS 92* (1980).

[51] Milner, R. *Communication and Concurrency*. Prentice-Hall, 1989.

[52] Nanz, S., and Hankin, C. Static analysis of routing protocols for ad-hoc networks, Mar. 25 2004.

[53] Nanz, S., and Hankin, C. A framework for security analysis of mobile wireless networks. *TCS: Theoretical Computer Science 367* (2006).

[54] Nikoletseas, S., and Spirakis, P. G. Probabilistic data propagation in wireless sensor networks. In *Theoretical Aspects of Distributed Computing in Sensor Networks*, S. Nikoletseas and J. D. Rolim, Eds., Monographs in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2011, pp. 353–380.

[55] Prasad. A calculus of broadcasting systems. *SCIPROG: Science of Computer Programming 25* (1995).

[56] Rahamatkar, S., Agarwal, A., and Kumar, N. Analysis and comparative study of clock synchronization schemes in wireless sensor networks. *Analysys 2*, 3 (2010), 536–541.

[57] Rappaport, T. S. *Wireless communications - principles and practice*. Prentice Hall, 1996.

[58] Sangiorgi, D., and Walker, D. *The Pi-Calculus — A Theory of Mobile Processes*. Cambridge University Press, 2001.

[59] Sasson, Y., Cavin, D., and Schiper, A. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2003)* (Mar. 2003).

[60] Segala, R. Testing probabilistic automata. In *CONCUR'96, Concurrency Theory (7th CONCUR'96)*, U. Montanari and V. Sassone, Eds., vol. 1119 of *Lecture Notes in Computer Science (LNCS)*. Springer-Verlag (Berlin), Pisa, Italy, Aug. 1996, pp. 299–314.

[61] Segala, R., and Lynch, N. A. Probabilistic simulations for probabilistic processes. *Nord. J. Comput 2*, 2 (1995), 250–273.

[62] Singh, A., Ramakrishnan, C. R., and Smolka, S. A. A process calculus for mobile ad hoc networks. *Sci. Comput. Program 75*, 6 (2010), 440–469.

[63] Song, L., and Godskesen, J. Probabilistic mobility models for mobile and wireless networks. In *Theoretical Computer Science*, C. Calude and V. Sassone, Eds., vol. 323 of *IFIP Advances in Information and Communication Technology*. Springer Boston, 2010, pp. 86–100.

[64] Tanenbaum, A. S. *Distributed operating systems*. Prentice Hall, London, 1995.

[65] Tanenbaum, A. S. *Computer Networks, 4th ed.* Prentice-Hall International, Inc., 2003.

[66] Tanenbaum, A. S., and van Steen, M. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[67] van Glabbeek, R. J. The linear time – branching time spectrum. Tech. Rep. CS-R9029, CWI, 1990.

[68] van Glabbeek, R. J. The linear time – branching time spectrum II (the semantics of sequential systems with silent moves). In *Proceedings of CONCUR '93* (1993), pp. 66–81.

[69] Zimmerman, H. OSI reference model - The ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications 28*, 4 (Apr. 1980), 425–432.