Behavioural Equivalences for Web Services

Thesis submitted to the University of Dublin
towards the degree of Doctor of Philosophy

July 2013

Giovanni Bernardi

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.

_____

Giovanni Bernardi

To the memory of Anna Mondin and Anna Maria Artusato

Antonius Block: I want knowledge! Not faith, not assumptions, but knowledge.

— Ingmar Bergman, *The Seventh Seal*

To reduce a romantic ideal to a working plan is a very difficult thing.

— Erskine Childers, *The Riddle of The Sands*

Summary

This thesis is a *foundational* and *systematic* investigation of the principles which ensure that a piece of communicating software can be replaced by another piece of communicating software, without hindering the operations of the pre-existing system.

By "foundational" we mean that our approach is mathematical; we define in formal terms notions such as "system correctness" and "safe software replacement", thereby providing reasoning techniques (i.e. proof methods) for the notions themselves.

By "systematic" we mean that our results lay bare the principles which allow the replacement of software in all the roles it can take: servers, clients and peers.

The investigation presented in this thesis stems from practical questions, such as the following:

Q1) if the client r is satisfied by the server p_1, what relation between p_1 and a server p_2 guarantees that p_2 satisfies r?

Q2) if the server p satisfies the client r_1, what relation between r_1 and a client r_2 guarantees that p satisfies r_2?

Q3) if the peer p_1 satisfies and is satisfied by the peer q, what relation between p_1 and a peer p_2 guarantees that p_2 satisfies and is satisfied by q?

The questions above are motivated by the practice of software maintenance; as they stand, though, they are rather vague, and a priori it is clear neither what they really mean, nor how to answer them.

Our foundational approach allows us to formulate the precise meaning of the questions above, and to answer them.

The contributions of this thesis are the following:

- we enrich the standard testing theory of [De Nicola and Hennessy, 1984] with new pre-orders, one for clients and one for peers, and present their behavioural characterisations (Theorem 4.2.37, Theorem 4.3.17);
- we introduce a compliance relation inspired to [Castagna et al., 2009; Laneve and Padovani, 2007], the pre-orders that arise from it, respectively for servers, clients, and peers; and we present the behavioural characterisations of these pre-orders (Theorem 5.1.15, Theorem 5.2.25, Theorem 5.3.26);
- we show a fully abstract model of the sub-typing à la [Gay and Hole, 2005] on first-order session types (Theorem 6.3.4). We define the model in two alternative ways, one uses to the testing theory, and one uses to the compliance theory (Proposition 6.5.19). The model justifies and explains in behavioural terms the sub-typing relation;
- we generalise the first-order model so as to exhibit a fully abstract model of the sub-typing on higher-order (i.e. standard) session types (Theorem 8.4.9). We also explain which conditions are necessary and sufficient to extend the first-order model to the higher-order setting.

Contents

Acknowledgements

My supervisor Matthew Hennessy is the person that during my PhD has influenced me most, both from a technical standpoint and a human standpoint. Matthew has incessantly given to me all the advices and explanations that I needed, and he has displayed an everlasting patience by listening to my ideas. He has also been my most constructive critic. I thank Matthew for all these things; I am grateful to have been one of his students.

One of the most remarkable people that I have met in Trinity is Vasileios Koutavas. I thank him for the time that he has spent discussing technical matters with me, and for all the pictures of the board that he has taken. His keen observations and his cheerful demeanour kept me motivated throughout the second half of my PhD, and helped me rearrange some thoughts.

As to my fellow PhD students, I thank Andrea Cerone, Colm Bhandal, and Carlo Spaccasassi. Andrea welcomed me in the fair city, and introduced me to its jolly social life. Colm and his thoughts out of the box secured us from boredom, making the office a lively place, and letting us forget the gloomy environment. Carlo supplied the office with an everlasting stock of fine coffee, and an even better mood.

I wish to thank my examiners, Andrew Butterfield and Simon Gay. Their careful review of this thesis helped in improving it, and lead to many engaging questions during my viva.

Many people filled my PhD years with good memories.

I thank the "Sunday brunch bunch": Francesco Caiazza, Florence De Filippis, and Silvia Taddei. They made me face many rainy, gray, and windy Mondays with a smile.

I thank Brendan Dunne, Paul Hynds, Anna Madden, Andres Mori, Heather Quinn, and Carlos Rodriguez. They have been an endless source of wit, wisdom, and have provided subjects for discussion aplenty. The climbing trip to Sardinia has been a turning point in my life, and I am glad that some of these fine fellows were there, and began sharing their ideas with me; let alone mirth and mirto. Also, it is a great pleasure to be still in the position of thanking all of them for having saved my life ever so often.

The regular escapades to Fontainebleau have been a panacea for the mood. This is true also because of Françoise and Bernard at the Gîtes des Jonquilles. Their dog Leo helped as well, for he always cheered me up, even after the most unsuccessful climbing days.

As a matter of fact, I happened to stay often in France, and not just around Fontainebleau. For this I have to thank Marion Lamé, Federico Ulliana, Hélène De Pooter, and Christophe Barnier. They have been very kind, and have proven to be good companions of chats and thoughts. I can hardly count the times they hosted me, but for sure I recall the fun we had in Paris, and, with Christophe, around Hyères. I thank Federico also for his providential rescue at the end of a most peculiar soirée.

Ringrazio Alessia, che oltre ad avermi definitivamente svegliato da un certo sonno dogmatico, ha deciso di unire le forze con me. Speriamo di farcela.

Ringrazio Gina e Antonio, per quello che hanno fatto, e perchè, nonostante tutto, mandano avanti il campo base in modo egregio.

Menzione d'onore ad Antonio Bazzo; nella speranza che nella diaspora non lo abbia fatto sentire distante.

Giovanni Bernardi, Dublin, July 4, 2013

Chapter 1

Introduction

Software that relies on the Internet pervades our lives. To check e-mails, read the news on the web, interact with people on social networks, buy items on-line, look for information on wikipedia, these are almost daily routines for many of us. At present even the political life of countries can be influenced by the discussions that take place via the World Wide Web [Grillo, 2013; Hauslohner, 2011; Woodward, 2011].

Much of the computations that let us carry out the above activities amount to a series of communications between *two* software systems:

- to browse the web we use clients that interact with `http` servers
- to read e-mails we use clients that interact with `pop/imap` servers
- to call a friend over the Internet, our friend and us use programs that implements `VoIP`

Two scenarios emerge from the examples above, the communication takes place between a client and a server, or between two peers. In a client/server setting the overall aim of the communication is to satisfy the client. In a peer to peer setting the aim is to satisfy *both* communicating parties.

The programs mentioned in the example list above follow communication protocols that are completely independent from each other; consider for instance the `pop3` and the `http` protocols. A consequence is that different programs have to be used to avail of different communication protocols.

The widespread adoption of the World Wide Web has lead to a general consensus on the technologies and the languages that underpin it, such as XML, CSS, javascript, and `http`. In turn, this has lead to a shift of paradigm in the development of applications: today much of the activities listed in our initial example can be carried out within a web browser, rather than requiring stand-alone applications. Via the World Wide Web, major IT companies offer to their users services such as e-mail boxes, programs to chat, and even simple office programs. In this setting web technologies provide the backbone for the deployment of the applications, and the protocols other than the `http` are used only by the servers that provide the services.

These facts give an intuition of what web-services are. According to [`w3c`, 2004],

> A Web-service is a software system identified by a URI [RFC 2396], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web-service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

Web-services depend on standard and widespread web technologies such as XML. This feature is meant to help integrate the middlewares and information systems of companies connected via a network, thereby facilitating the business to business (`b2b`) activities. If a company runs a service p,

and another company runs a program r that needs p to work, then via a network and thanks to the standard format of the messages, r can interact with p, and the two programs can cooperate.

Also web-services for b2b fall into the client/server and the peer to peer scenarios that we described earlier on.

From now on we disregard the technological details and the particular aim of software, and focus only on the fundamental aspects that characterise clients, servers, peers and their correctness.

Let us think of "systems" as compositions of two programs r and p that interact with each other; we write $r \mid\mid p$ to denote the concurrent execution of the programs r and p. The systems that we normally use are correct, in the sense that when we let clients interact with servers, the clients are in some sense satisfied; and similarly for peers. The situation is analogous for companies and the services that they offer. The meaning of "correctness" of a software system, or of "satisfaction" is not clear a priori. If we want to test software, our tests are satisfied if they are passed. If we want to browse the web, the client that we use is satisfied if all its requests are answered by some web server. If we chat with a friend via a network, our programs are satisfied as long as they can keep on interacting.

It is not easy to understand when a piece of software can be changed without breaking the correctness of a system. For instance, suppose that a client r_1 is satisfied by a server p, that is $r_1 \mid\mid p_1$ is a correct system. If we were to replace a client r_1 with a client r_2 would the new system still be correct? In particular,

(Q1) what is the relation between r_1 and r_2 which guarantees that r_2 is satisfied by p?

We have also the dual question, if $r \mid\mid p_1$ is a correct system, and we were to replace p_1 with a server p_2 how could we tell whether r would still be satisfied? That is

(Q2) what is the relation between p_1 and p_2 which guarantees that p_2 satisfies r?

If we deal with peers, then the question becomes symmetric; if $p_1 \mid\mid q$ is a correct system of peers, and we want to replace p_1 with p_2, then

(Q3) what is the relation between p_1 and p_2 which guarantees that p_2 satisfies q and that q satisfies p_2?

The practice of software maintenance shows that the questions above are not mere theoretical speculation. For instance, the e-mail service of Google, Gmail, underwent a failure on the 31^{st} of April, 2011. [Treynor, 2011] commented on the official Gmail blog as follows,

> So what caused this problem? We released a storage software update that introduced the unexpected bug, which caused 0.02% of Gmail users to temporarily lose access to their email. When we discovered the problem, we immediately stopped the deployment of the new software and reverted to the old version.

In December 2012 the same e-mail service was hampered again, by a problem traced down to a bug in a routine update to the load balancing software.

As for web-services and the b2b scenarios, similar issues have taken place. On the 29^{th} of June 2012, Amazon web-services, AWS, underwent a service disruption. The [AWS Team, 2012] explains in great detail what happened. One problem was due to software,

> [...] a small number of Multi-AZ RDS[1] instances did not complete failover, due to a software bug. The bug was introduced in April when we made changes to the way we handle storage failure. It is only manifested when a certain sequence of communication failure is experienced, situations we saw during this event as a variety of server shutdown sequences occurred.

[1] Availability Zone, Relational Database Service.

Both explanations remark that the issues in the software systems were due to software updates. The explanation provided by Amazon also lay bare that their problem was due to the communication sequence that the software engaged in.

These facts highlight the importance of having means to guarantee that software update can take place without hampering the existing communication patterns, and without introducing unexpected communication patterns that may disrupt the operations of a software system.

This thesis is a *systematic* investigation of five theories that let us answer the questions on software replacement that we posed earlier, (Q1), (Q2), and (Q3). By "systematic" we mean that our results lay bare the principles which allow the replacement of software in all the roles it can take: servers, clients and peers.

The aim of this thesis is largely *foundational*; we want to state in a mathematical and rigorous way what it means for a software system to be correct, and understand when a program p can be replaced by a program q keeping the system correct. In our thinking we focus on the characteristic features of software interaction, inputs and output operations, and we abstract away from minor details.

Our foundational approach sheds light on the design of software. To answer rigorously the questions (Q1), (Q2), and (Q3), we will have to put forth a series of mathematical notions; these notions give us design principles to write software that does not invalidate the correctness of existing systems. The design principles are correct by virtue of the mathematical treatment that they emerge from.

Our mathematical reasoning is based on set theory; we will use relations on programs, denoted by $\mathcal{S}$ and $\sqsubseteq$, to mean that it is safe to run the client r with the server p, $r\ \mathcal{S}\ p$; and that if we replace p_1 with p_2 in a correct system $r \parallel p_1$, then the new system $r \parallel p_2$ is still correct; this is denoted $p_1 \sqsubseteq p_2$.

We have argued that there are different ways to think of the satisfaction of clients, and of the correctness of software systems. Thinking in terms of sets and relations, this means that we can define many relations $\mathcal{S}$ on programs, to express some notion of satisfaction. As different relations $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_2, \ldots$ are of interest, different theories have to be investigated.

The main objects of our study are not the relations to express the satisfaction of a program. Rather, our efforts are devoted towards the understanding of the refinement relations that arise from the satisfaction relations that we pick. Let us fix a relation $\mathcal{S}$, and write $p \sqsubseteq_{\text{SVR}} q$ if $r\ \mathcal{S}\ p$ implies that $r\ \mathcal{S}\ q$. The relation $\sqsubseteq_{\text{SVR}}$ that we obtain tells us when a server q can replace a server p without hindering the correctness of a system $r \parallel p$. The relation $\sqsubseteq_{\text{SVR}}$ embodies a principle to replace *servers*. In similar ways we can define refinements for clients and for peers, so as to obtain three relations,

$$\sqsubseteq_{\text{SVR}}, \qquad \sqsubseteq_{\text{CLT}}, \qquad \sqsubseteq_{\text{P2P}}$$

It is necessary to study all the pre-orders above. Suppose we studied only how to safely replace servers, that is $\sqsubseteq_{\text{SVR}}$, and not how to safely replace clients. Then the correctness of a system could be hampered by replacing the clients. The study of $\sqsubseteq_{\text{CLT}}$ solves the problem; and similarly does the study of $\sqsubseteq_{\text{P2P}}$ for the peer to peer scenario.

The properties of the relations above depend on the notion of satisfaction $\mathcal{S}$ that generated them.

One may wish the pre-orders to enjoy some particular properties. For instance, if two servers are related, say $p \sqsubseteq_{\text{SVR}} q$, then q may offer to the clients more choices than p. This property is known as "width extension" [Laneve and Padovani, 2007].

We give a more concrete example. Let ATM_A denote a cash machine that allows us withdraw cash, and then starts again,

$$ATM_A = ?\texttt{withdraw}.ATM_A$$

and let ATM_B be a similar cash machine that allows us also to top-up mobile phones,

$$ATM_B = ?\texttt{withdraw}.ATM_B + ?\texttt{phone}.ATM_B$$

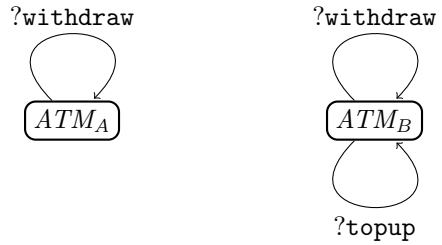


Figure 1.1: Two cash machines

Intuitively, the equalities above define two programs; the symbols ?`withdraw` and ?`phone` stand for interactions that the programs can perform with the environment. The symbol + in the definition of ATM_B represents a choice, ATM_B lets the environment decide which interaction to perform. The syntax ?`withdraw`.ATM_B means that after the interaction ?`withdraw` the program behaves accordingly to the definition of ATM_B. The behaviours of the programs ATM_A and ATM_B are sketched in Figure 1.1.

One could argue that ATM_B is in some sense better than ATM_A, in that ATM_B satisfies all the customers satisfied by ATM_A, and it also offers them more choices. So, once we have fixed a notion of satisfaction $\mathcal{S}$, we may wish that the following be true,

$$ATM_A \sqsubseteq_{\text{SVR}} ATM_B$$

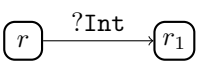
In general, this may or may not be the case; that is the refinement for servers given by a particular $\mathcal{S}$ may or may not allow the inequality above.

If a refinement $\sqsubseteq_{\text{SVR}}$ generated by a relation $\mathcal{S}$ does not enjoy a certain property, and we cannot change $\mathcal{S}$, then other ways to vary $\sqsubseteq_{\text{SVR}}$ have to be investigated. For example, a relation for satisfaction $\mathcal{S}$ can give rise to many different refinements for servers, clients and peers; the properties of these pre-orders depends on the language used to write programs. This introduces yet another reason to study different theories.

In this thesis we we investigate five theories. Rather than putting forth new formalisms, we examine existing ideas and deepen our understanding of them, shedding light on their implications and on the connections between them. Two theories that we investigate are due to two notions of satisfaction applied to the same general language, the well known CCS *without τ's* of [De Nicola and Hennessy, 1987]. The other theories are given by applying the same notions of satisfaction to the more restrictive language of *session contracts* [Bernardi and Hennessy, 2012].

Before describing the theories that we will be concerned with, we explain how we think of interactions between programs.

To formalise the sequence of operations (i.e. interactions) that a program performs, we use graphs such as the ones in Figure 1.2. In that figure, the graph at the top describes a program r that inputs two values, an integer and a boolean, and then is in a successful state, (i.e. it can perform ✓). Graphs such as the ones in Figure 1.2 describe the *operational* semantics of software, and they are referred to as *labelled transition systems*, LTS. An LTS contains the states that a program can be in, and the transitions that lead from one state to another. The transitions are usually depicted as arrows, and are decorated with a label. Labels describe the interactions that make a program change states; that is, they explain *why* a program changes state. For instance, the fact that the program r Figure 1.2 is willing to input an integer, thereby moving to state r_1 is represented by the following transition:



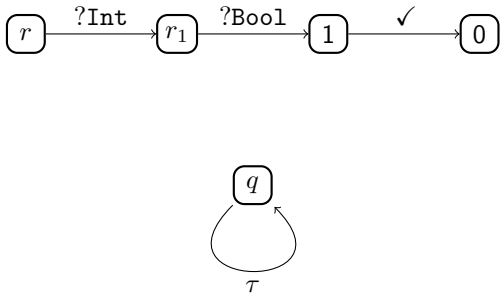


Figure 1.2: A program that succeeds after the input of an integer and of a boolean; and a program that loops forever

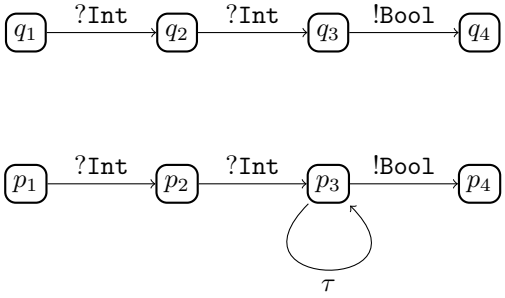
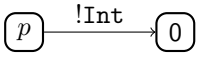


Figure 1.3: Two programs that interact according to the sequence ?Int?Int!Bool

Intuitively, interactions between r and a program q take place when we execute them concurrently, denoted $r \parallel p$, and each input/output operation of one program is matched by a *co-action* performed by the other program. For instance, let the operations of p be described by the following LTS



The action !Int represents the output of an integer, and it is a co-action of ?Int.

The program p is willing to output an integer, and then perform no other operations. According to the intuitions we have described, the parallel composition $r \parallel p$ performs one interaction, denoted $r \parallel p \xrightarrow{\tau} r_1 \parallel \mathtt{0}$, and then becomes stable; the system cannot proceed further.

The bottom graph in Figure 1.2 describes a program q that performs only one transition, labelled by τ. That transition represents a computation that takes place inside of q, and that the environment has no power over. Since the τ action represents internal computation, the program q performs an infinite sequence of internal computations,

$$q \xrightarrow{\tau} q \xrightarrow{\tau} q \xrightarrow{\tau} q \xrightarrow{\tau} \dots$$

We say that q *diverges*, whereas programs that perform only finite internal computations *converge*.

If we execute q in parallel with r, then also the resulting composition diverges,

$$q \parallel r \xrightarrow{\tau} q \parallel r \xrightarrow{\tau} q \parallel r \xrightarrow{\tau} \dots$$

The framework that we have sketched is essentially CCS; it has been put forth by Milner, and his book [Milner, 1989] is a standard reference on the topic.

1.1 Must theory

The theory of MUST testing, known also as testing theory, has been presented in [De Nicola and Hennessy, 1984], and the subsequent [Hennessy, 1985]. Testing theory is a landmark within the formalisms to assess software equivalence, and this renders it a good starting point for our research.

It is common practice to test software in order to exhibit errors. Testing can be performed also in presence of communications between programs.

By running a process p in parallel with r, denoted $r \parallel p$, we can check whether in all the possible executions r reaches its successful state. This means that the interactions offered by p satisfy the test r; in other words p must pass the test r. Intuitively, this is the meaning of the relation MUST; in particular of statements such as p MUST r.

Plainly, in this setting a process q is better than a process p if q passes more tests; and q and p are equivalent is they pass the same set of tests.

The MUST testing relation has been introduced in [De Nicola and Hennessy, 1984], to define a refinement for processes described by the intuition above, the well known MUST pre-order, $\sqsubseteq_{\text{MUST}}$.

The relation MUST expresses the satisfaction of tests, so the theory of MUST testing can be smoothly casted into a more general client/server setting. As the satisfaction is biased towards the tests, they can be seen as clients, while processes can be seen as servers.

To study peers we use a symmetric version of MUST which requires the satisfaction of *both* parties involved in a software system $r \parallel p$. To this end we have also to combine the language of processes and the language of tests; this allows us to write terms that model programs that can reach satisfaction, while testing another program.

In this thesis we will extend the standard framework by studying three pre-orders, one for servers (i.e. processes), one for clients (i.e. tests), and a pre-order for peers,

$$\sqsubseteq_{\text{SVR}}, \qquad \sqsubseteq_{\text{CLT}}, \qquad \sqsubseteq_{\text{P2P}}$$

Note that it is necessary to study anew the pre-order for servers, because we allow processes to perform $\checkmark$, thereby extending the language of [De Nicola and Hennessy, 1984; Hennessy, 1985]; so we have to check the impact of this extension, as $\sqsubseteq_{\text{SVR}}$ may differ from $\sqsubseteq_{\text{MUST}}$.

Roughly speaking, the outcome of our investigation is that in this setting,

- a server p_2 is better than a server p_1 if (a) all the interaction sequences of p_2 can be performed also by p_1; and (b) the two servers converge *along* the execution of the interaction sequences in the same manner
- a client r_2 is better than a client r_1 if (a) all the interaction sequences that r_2 performs without being satisfied are performed also by r_1 without being satisfied;(b) if an interaction leads r_1 out of a deadlock, then that interaction leads also r_2 out of a deadlock
- a peer q is better than a peer p if q is a better client than p, and if this is true, then q is also a better server than p.

As to the meaning of "converging along", observe Figure 1.3. There the program q_1 converges along the interaction sequence ?`Int`?`Int`!`Bool`, because all the states it reaches during the execution converge. On the contrary, the program p_1, does not converge along that sequence, because after having performed ?`Int`?`Int`! it reaches a divergent state.

Observe that a posteriori the principles to replace servers are the same prescribed by the standard MUST pre-order; moreover, they imply that

$$ATM_A \not\sqsubseteq_{\text{SVR}} ATM_B \tag{1.1}$$

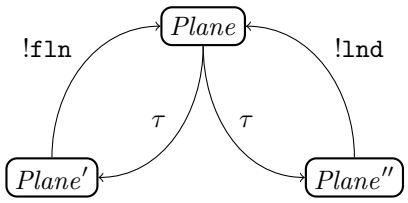
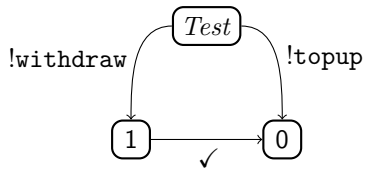


Figure 1.4: A program for a plane

To see why Eq. (1.1) is true, consider the following LTS,



The only interaction that *Test* can engage in with ATM_A is due to `withdraw`, and *Test* reports success after that interaction. With ATM_B there exists the possibility of an interaction via `topup`, which leads *Test* to a deadlock state. The fact that *Test* is always passed by ATM_A but it may not be passed by ATM_B proves Eq. (1.1).

To extend the standard theory with new pre-orders for clients and peers has also the advantage of laying bare some fundamental notions that we will need to study the refinements of a second theory.

1.2 Compliance theory

The second formalism that we study is a theory of compliance, and it is an alternative to the MUST testing theory.

We motivate the investigation of the compliance theory. Recall the ATM we described earlier on, ATM_A; and observe *Plane* in Figure 1.4. The program *Plane* keeps on performing some internal computation, represented by the τ actions, whereby it decides to communicate to the environment either that it is landed (`!lnd`), or that it is flying (`!fln`). Plainly, the program *Plane* follows a completely different communication pattern than ATM_A. Nevertheless, if we compare the two programs as clients according the the MUST setting, they are equivalent, $ATM_A \eqsim_{\text{CLT}} Plane$. This may look surprising, but indeed it is sensible. If clients are tests, then when we use $\sqsubseteq_{\text{CLT}}$ we compare software *as tests*. Now we see why $ATM_A \eqsim_{\text{CLT}} Plane$; as none of these programs ever reach a successful state (i.e. perform ✓), they are indeed equivalent as tests, for they are never passed.

This criterion to assess when a client can replace another client neither fits with our daily experience of the services we use on the web, nor fits with the usage of web-services at large.

This motivates the introduction of an alternative relation to formalise the satisfaction of a client, or of two peers. This is the compliance relation, $\dashv$.

Roughly speaking, according to the compliance relation, a client r is satisfied by a server p, $r \dashv p$, as long as the requests of the client are answered by the server, and if the interactions cannot go on, then the client has successfully completed its computation. This notion of satisfaction differs from MUST; the compliance relation essentially ensures that the interactions can go on, whereas the MUST testing checks the presence of successful states in the computations.

The compliance relation and its symmetric version also give rise to three pre-orders in a natural fashion,

$$\sqsubseteq_{\text{SVR}}, \qquad \sqsubseteq_{\text{CLT}}, \qquad \sqsubseteq_{\text{P2P}}$$

In this setting we can indeed prove that $ATM_A \neq_{\text{CLT}} Plane$. For example, the server $S = !\texttt{withdraw}.S$ satisfies ATM_A, but not $Plane$. Intuitively, this is true because ATM_A keep on interacting with forever S; whereas the composition $Plane \parallel S$ enters in a deadlock, without $Plane$ being satisfied:

$$Plane \parallel S \xrightarrow{\tau} Plane \parallel S \overset{\tau}{\not\longrightarrow}$$

Even though the MUST testing and the relation $\dashv$ are altogether different, the reasoning techniques for the pre-orders due to the compliance are quite close to the reasoning techniques for the pre-orders due to MUST.

Our investigation of the compliance theory leads to the following principles:

- a server p_2 is better than a server p_1 if (a) all the interaction sequences of p_2 can be performed also by p_1; and (b) the two servers converge *after* the execution of the interaction sequences in the same manner
- a client r_2 is better than a client r_1 if (a) all the interaction sequences that r_2 performs reaching a deadlock are performed also by r_1 reaching a deadlock; (b) if an interaction leads r_1 out of a deadlock, then that interaction leads also r_2 out of a deadlock
- a peer q is better than a peer p if q is a better client than p, and if this is true, then q is also a better server than p.

We have already argument that the program p_1 in Figure 1.3 does not converge along the interaction sequence $?\texttt{Int}?\texttt{Int}!\texttt{Bool}$; on the other hand it converges after having performed the sequence, for the state that it reaches, p_4, does not diverge.

Also in this case the principles to replace servers let us prove that ATM_B is not a better server than ATM_A,

$$ATM_A \not\sqsubseteq_{\text{SVR}} ATM_B$$

The inequality above and Eq. (1.1) ensure that neither the compliance pre-orders nor the MUST pre-orders that allow width extension to take place. However, by restricting the power of the programming language at hand, we can use MUST and $\dashv$ to obtain refinements that allow width extension. This brings us to the study of session types.

1.3 Session types

Most of the software we use interacts via *binary* communication channels, the so-called sockets. These channels are binary because are made of two end-points. If we represent abstractly the channels as $a, b, c, \ldots$, then the end-points of a channel are a^+ and a^-. Intuitively, at each moment in a network, one program owns the end-point a^+, and one program owns the dual end-point, a^-.

Session types, proposed first by [Honda, 1993], are *syntactic* annotations usually assigned to the end-points used by programs. These types describe the sequence of data input/outputs that a program is willing to perform on a given end-point.

Let the session types S be defined as follows,

$$S = \mu X.\, \&\langle\, \texttt{add}\colon\ ?[\,\texttt{Int}\,]; ?[\,\texttt{Int}\,]; ![\,\texttt{Int}\,]; X, \texttt{stop}\colon \text{END}\,\rangle \tag{1.2}$$

Let us comment the syntax above. The type S is defined using *recursion* on the variable X ($recX\ldots$); the body of S is defined by a *branch* constructor ($\&\langle\ldots\rangle$), which maps two labels (`add` and `stop`) to other two types. After `add` there is a type defined by two input constructs ($?[-]$), and output ($![-]$) and the variable to perform recursion. After `stop` there is the *termination* type, END, that represents a terminated communication.

If a program p uses an end-point a^+ at type S, then p operates on a^+ according the this logic: p offers a "menu" with two choices, `add` and `stop`, and waits for the program with the end-point a^- to make a choice;

- if the choice is `add`, then p proceeds as follows,
  1) p reads via a^+ a datum of type `Int`,
  2) p reads via a^+ a datum of type `Int`,
  3) p write on a^+ a datum of type `Int`,
  4) p starts again to act on a^+ as we described;
- if the choice is `stop`, then p does not interact any longer via the end-point a^+.

The notion of *duality* in the theory of session types is a central one. The end-points associated to a channel are "dual", and the programs that use them are supposed to show "dual", that is complementary, behaviours on the end-points. If a program offers some choices, then the program at the other end of the channel chooses among these choices. If one program performs an input, the other program has to perform an output, and so forth.

The duality between types expresses the notion of satisfaction in the setting of session types, for if S is the dual of $\overline{S}$, than each communication performed on one end-point, will be matched by a communication on the dual end-point (if communications can take place at all).

The main refinement in the theory of session types is the sub-typing relation, $\preccurlyeq_{\mathsf{sbt}}$, defined by [Gay and Hole, 2005].[2] This sub-typing adds flexibility to the overall theory, in that it allows to replace session types without breaking the correctness of a system. For instance, if $S_1 = \&\langle\, \texttt{tea}\colon \text{END} \,\rangle$ and $S_2 = \&\langle\, \texttt{tea}\colon \text{END},\ \texttt{moka}\colon \text{END} \,\rangle$, then the sub-typing ensures that $S_1 \preccurlyeq_{\mathsf{sbt}} S_2$. Note that this means that the refinement $\preccurlyeq_{\mathsf{sbt}}$ allows width extension.

Session types lack any semantics; they are merely syntactic entities. As a consequence, the relation $\preccurlyeq_{\mathsf{sbt}}$ is inevitably defined by using the syntax of terms. This has some drawbacks:

- the definition of $\preccurlyeq_{\mathsf{sbt}}$ explains neither how the behaviours of programs are related, nor how the behaviour of types are related
- there is a degree of arbitrariness in the definition of $\preccurlyeq_{\mathsf{sbt}}$. Why should we use that definition of [Gay and Hole, 2005], rather than some other relation?
- a priori, it is not clear how to adapt the definition of $\preccurlyeq_{\mathsf{sbt}}$ to alternative type refinements; in particular the ones biased towards clients

To overcome the drawbacks above, in our study we explain the existing theory of session types. We do this by interpreting types into a language equipped with an operational semantics; in turn, the semantics lets us define pre-orders for servers and clients in a straightforward and *non arbitrary* way; and these pre-orders let us justify the standard sub-typing via a fully abstract model. Essentially, a posteriori it turns out that if we assign a semantics to session types, and $S_1 \preccurlyeq_{\mathsf{sbt}} S_2$, then the behaviours of S_1 and S_2 are indeed related.

Other than the models of the sub-typing, the outcome of our investigation is that it is possible to reason about the behavioural pre-orders for session types merely by looking at the syntax of types.

First-order session types

Our investigation on session types begin with *first-order* types. First-order session types are types that can express inputs and outputs operations only on terms that are not session types. For instance

[2] In that paper the sub-typing is denoted $\leq_c$.

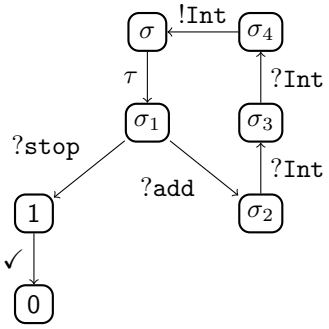


Figure 1.5: Operational semantics of the encoding of the type S (see Eq. (1.2))

the following terms are first-order session types

$$?[\,\texttt{Real}\,];\text{END}, \qquad \mu X.\ ![\,\texttt{Bool}\,];?[\,\texttt{Int}\,];X,$$

the intuition being that `Real` and `Int` are base types, so not session types. The following terms are not first-order,

$$![\,\mu X.\ ?[\,\texttt{Bool}\,];X\,];\text{END}, \qquad \&\langle\, \texttt{opt1}\colon\ ?[\,\text{END}\,];\text{END},\ \texttt{opt2}\colon\ ![\,?[\,\text{END}\,];\,];\text{END}\,\rangle \tag{1.3}$$

The input/output fields $?[-]$ and $![-]$ of the types in Eq. (1.3) contain session types, so the terms in Eq. (1.3) are higher-order.

The first task that we carry out is to assign an operational semantics to first-order session types. To this end, we adapt the semantics of the processes in $\mathsf{CCS}_{\mathrm{w}\tau}$, and define the language of session contracts. A straightforward encoding of session types into session contracts lets us associate the semantics of session contracts to session types. In Figure 1.2 we saw the operational semantics of simple processes; essentially some graphs. Intuitively, we use the same approach to describe how a type is meant to interact with the environment, thereby giving a meaning to session types.

For instance, we will associate the term S in Eq. (1.2) with the graph depicted in Figure 1.5; in that graph σ is the session contract resulting from the encoding of S.

In view of the operational semantics assigned to session types, we use the relations for satisfaction MUST and $\dashv$ to study the refinements for servers and clients that arise in the setting of session types. These refinements are respectively

$$\sqsubset^{\mathsf{fo}}_{\sim\mathrm{SVR}}, \qquad \sqsubset^{\mathsf{fo}}_{\sim\mathrm{CLT}}$$

and

$$\sqsubset^{\mathsf{fo}}_{=\mathrm{SVR}}, \qquad \sqsubset^{\mathsf{fo}}_{=\mathrm{CLT}}$$

The definitions of the pre-orders above follow the intuitions that lead to the definitions of the pre-orders for the general theories of testing and of compliance. Hence the relations above are *not* defined in a syntactic way, but in a behavioural way.

We do not define the refinements for peers, for we use the sub-typing on first-order types, $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$, as the refinement for peers (up-to the interpretation of contracts into types).

In the sequel of this discussion, let us assume that σ_1 and σ_2 are the encoding of two types S_1 and S_2. If σ_1 is related by $\sqsubset^{\mathsf{fo}}_{\sim\mathrm{SVR}}$ with σ_2, then the way in which σ_2 interacts with the environment, i.e. its observable behaviour, will satisfy all the clients passed by σ_1. A similar property is true also for $\sqsubset_{\mathrm{SVR}}$.

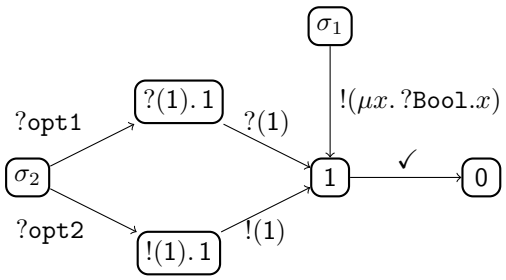


Figure 1.6: An instance of the LTS of higher-order session contracts. The higher-order session contracts appear on the transitions as well as in the states

By and large, the outcome of our study is that the behavioural pre-orders can be characterised purely in a syntactic manner, and that the intersections of these pre-orders are a fully abstract model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ (via our encoding):

- $S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2$ if and only if σ_1 $(\sqsubset^{\mathsf{fo}}_{\sim\mathrm{SVR}} \cap \sqsubset^{\mathsf{fo}}_{\sim\mathrm{CLT}})$ σ_2
- $S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2$ if and only if σ_1 $(\sqsubset^{\mathsf{fo}}_{=\mathrm{SVR}} \cap \sqsubset^{\mathsf{fo}}_{=\mathrm{CLT}})$ σ_2

First-order session types have limited applications, so we extend the language of session types by allowing them to input/output also session types; and we investigate the resulting theory.

Higher-order session types

As we saw in Eq. (1.3), higher-order session types can contain session types in their input/output fields. The language of higher-order session types amounts to the session types á la [Gay and Hole, 2005]. In this context we study only the pre-orders due to the compliance relation,

$$\sqsubset^{\mathsf{ho}}_{=\mathrm{SVR}}, \qquad \sqsubset^{\mathsf{ho}}_{=\mathrm{CLT}}$$

The outcome of our investigation is an extension of the results proven in the first-order setting:

- the behavioural pre-orders can be characterised purely in a syntactic manner
- the intersection of those pre-orders is a fully abstract model of the sub-typing $\preccurlyeq_{\mathsf{sbt}}$: $S_1 \preccurlyeq_{\mathsf{sbt}} S_2$ if and only if σ_1 $(\sqsubset^{\mathsf{ho}}_{=\mathrm{SVR}} \cap \sqsubset^{\mathsf{ho}}_{=\mathrm{CLT}})$ σ_2

We do not investigate the MUST pre-orders in the higher-order setting, and leave this as an open problem.

The chief difficulty in unravelling the results is due to a technical issue, that we briefly comment on.

To accommodate the higher-order terms in the existing model of first-order types, we extend the language of session contracts with higher-order constructs. For example, the session contracts that represent the types in Eq. (1.3) are the following terms,

$$\sigma_1 = !(\mu x.\, ?\mathtt{Bool}.x).\, 1 \qquad \sigma_2 = ?\mathtt{opt}_1.(?(1).\, 1) + ?\mathtt{opt}_2.(!(1).\, 1)$$

In turn, this forces us to extend the operational semantics of session contracts; we have to allow the transition of the semantics to be labelled with higher-order session contracts themselves. The operations of the session contracts above are depicted in Figure 1.6.

As a result, the intuitions of co-action and of when transitions should synchronise is no longer clear; in fact, this is a non-trivial matter, and to account for it we parametrise the LTS via a binary relation on session contracts, $\mathcal{B}$, and introduce a family of "dependent" LTSs.

In order to obtain an LTS that does not depend on any arbitrary relation $\mathcal{B}$, we study certain functions, which turn out to have greatest fixed points. These fixed points are the pre-orders $\sqsubseteq^{\mathsf{ho}}_{\mathrm{SVR}}$ and $\sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}$. Indeed, in devising the higher-order theory of compliance our efforts are principally oriented towards the definition of the fixed points $\sqsubseteq^{\mathsf{ho}}_{\mathrm{SVR}}$ and $\sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}$.

1.4 Contributions

From a technical standpoint, the contributions of this thesis are the following ones,

(a) we enrich the standard testing theory with new pre-orders, and present their behavioural characterisations (Theorem 4.2.37, Theorem 4.3.17);

(b) we introduce a new compliance relation, the three pre-orders that arise from it, and we show the behavioural characterisations of these pre-orders (Theorem 5.1.15, Theorem 5.2.25, Theorem 5.3.26);

(c) we show a fully abstract model of the sub-typing relation on first-order session types (Theorem 6.3.4); the model we use can be defined in two alternative ways, one due to the testing theory, and the other due to the compliance theory (Proposition 6.5.19);

(d) we generalise the model due to the compliance so as to exhibit a fully abstract model of the sub-typing on higher-order session types (Theorem 8.4.9).

The first two contributions ((a) and (b)), i.e. the systematic study of the testing theory and the compliance theory, are a necessary step to use the formalisms themselves as foundations for software maintenance.

The last two contributions ((c) and (d)), that is the models for first-order and higher-order session types, essentially show that the theory of session types à la [Gay and Hole, 2005] can be *recovered and justified* by using the MUST testing, or the compliance relation. The connection that we establish between the different formalisms shows that the testing and the compliance theories on session contracts are more primitive than the theory of session types.

Also, our model shows that session contracts in some sense *extend* the theory of session types, by virtue of a series of server pre-orders and client pre-orders.

In the first-order setting, we take the sub-typing $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ to be the peer pre-order; the MUST theory for session types amounts to the refinements

$$\sqsubseteq^{\mathsf{fo}}_{\widetilde{\mathrm{SVR}}}, \qquad \sqsubseteq^{\mathsf{fo}}_{\widetilde{\mathrm{CLT}}}, \qquad \sqsubseteq^{\mathsf{fo}}_{\widetilde{\mathrm{CLT}}} \cap \sqsubseteq^{\mathsf{fo}}_{\widetilde{\mathrm{SVR}}}$$

while the compliance theory amounts to the refinements

$$\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}, \qquad \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}}, \qquad \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \cap \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}}$$

In the higher-order setting, the pre-order for peers is the sub-typing $\preccurlyeq_{\mathsf{sbt}}$, and the compliance theory amounts to the following pre-orders

$$\sqsubseteq^{\mathsf{ho}}_{\mathrm{SVR}}, \qquad \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}, \qquad \sqsubseteq^{\mathsf{ho}}_{\mathrm{SVR}} \cap \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}$$

The material is presented in such a way as to render the transition from one theory to another theory as smooth as possible, showing the crucial differences between the subjects, and exploiting their similarities.

Equivalences and pre-orders In this thesis we study the characteristic properties of a number of refinement relations; that is pre-orders. We study these relations rather than equivalences because pre-orders are more primitive than equivalences: each pre-order we study through this thesis generates an equivalence. For instance we deem two clients r_1 and r_2 as equivalent, $r_1 =_{\text{CLT}} r_2$, if $r_1 \sqsubseteq_{\text{CLT}} r_2$ and $r_2 \sqsubseteq_{\text{CLT}} r_1$. The properties of the pre-order $\sqsubseteq_{\text{CLT}}$ then shed light immediately on the equivalence relation $=_{\text{CLT}}$.

Structure of the thesis

This thesis is divided into two parts.

The first part is devoted to the study of first-order theories; this means that the transition systems that we use as operational semantics have "basic" entities as labels, for example actions, and base types.

The second part is devoted to the extension of some results of the first part to transition systems generated by higher-order languages. Roughly speaking, in these transition systems the labels can be "programs" themselves.

We briefly describe the contents of each chapter.

In Chapter 2 we define the languages that we use throughout the first part of the thesis. While session types need little discussion and are merely syntactical entities, to define processes (i.e. $\mathsf{CCS}_{\text{w}\tau}$) and session contracts we have also to introduce the operational semantics of these languages. We conclude the chapter showing how to encode session types in session contracts, thereby assigning them an operational semantics.

In Chapter 3 we use the LTS to formalise the satisfaction of clients and peers; that is to define the MUST testing and the compliance relation. We also show that in the restricted setting of session contracts, both relations can be faithfully described using only the syntax of terms.

In Chapter 4 we extend the standard testing theory, by investigating the pre-orders for servers, clients and peers due to MUST.

Chapter 5 is organised as Chapter 4, but the theory investigated is due to the compliance relation. At the end of Chapter 5 we summarise our knowledge of the pre-orders studied on the general LTS of $\mathsf{CCS}_{\text{w}\tau}$; moreover, we prove a series of results on more restrictive LTSs.

In Chapter 6 we turn our attention to the LTS of session contracts, tailoring the definitions of the refinements to it. We study both the resulting MUST theory and the resulting compliance theory, and exhibit a fully abstract model of first-order session types. This sheds light on the known theory of session types, and also show how to extend it.

In Chapter 7 we extend the languages for sessions of Chapter 2. In particular, we parametrise on a relation $\mathcal{B}$ the LTS that describes the interactions of higher-order session contracts. This leads to the definition of dependent compliance relations, $\dashv_{\mathcal{B}}$. The chief result of the chapter is that the syntactical characterisation of $\dashv$ still holds true for the dependent compliances.

In Chapter 8 we introduce the dependent client and server pre-orders, $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ and $\sqsubseteq^{\mathcal{B}}_{\text{SVR}}$. By using these pre-orders we show that there is non-arbitrary way to remove the parameter $\mathcal{B}$ from the LTS. We do this by building the fixed points of suitable functions of the form $\lambda X.\, \sqsubseteq^{X}_{\text{CLT}}$ and $\lambda X.\, \sqsubseteq^{X}_{\text{SVR}}$. We use these results to exhibit a fully abstract model of higher-order session types.

In Chapter 9 we outline the ongoing work and the problems that we want to tackle by means of our behavioural models of session types.

In Chapter 10 we review the relevant literature, summarising the development of the formalisms we use, and showing the state of the art.

We conclude the thesis in Chapter 11, by summarising the results, and briefly discussing a series of open questions.

Prerequisites

The reader is expected to be familiar with naïve set theory, order theory, and first order logic. Standard references on these topics are respectively [Halmos, 1960], [Davey and Priestley, 2002], and [Mendelson, 1997].

The notation that we do not explain, for example how we define syntax languages, is standard and used regularly in the literature, for instance in [Hennessy, 2007; Milner, 1999; Pierce, 2002].

Throughout the thesis we use heavily induction and co-induction; an excellent and broad explanation of both techniques is [Sangiorgi, 2012]. Chapter 21 of [Pierce, 2002] is a standard reference on (co)induction, while another standard reference on induction is [Winskel, 1993, Chapters 3 and 4]. More advanced books on the matter are [Barwise and Moss, 1996] and [Sangiorgi and Rutten, 2011].

Notation We explain a few conventions on the notation that we will use.

We will use the symbols $\sqsubseteq$ and $\underset{\sim}{\sqsubseteq}$ to denote pre-orders defined by using the operational semantics of programs. We will use the symbols $\preceq$ and $\underset{\sim}{\preceq}$ to denote "alternative" relations, that shed light on the observable behaviour of programs.

We will use many relations defined by (co)induction; normally we will define these relations by using explicitly rule functionals, and taking their least or greatest fixed points. We use the symbol $\mathcal{F}$ to denote such rule functionals. When interested in the greatest fixed point of a functional, we decorate $\mathcal{F}$ with a superscript, for example $\mathcal{F}^A, \mathcal{F}^B, \ldots$; when interested in the least fixed point of a functional, we decorate $\mathcal{F}$ with a subscript $\mathcal{F}_A, \mathcal{F}_B, \ldots$.

Part I

First-order theories

Chapter 2

First-order languages

In this chapter we formally define the languages that we will use throughout the first part of this thesis: a language of communicating processes, the language of session contracts, and the language of session types. The first two languages have an operational semantics in the form of an LTS, and their terms are really meant as handy denotations for parts of the LTS. The third language, session types, has no formal semantics.

The languages we deal with in this part of the thesis are *first-order*, in the sense that they cannot perform input/output operations on the terms of the languages themselves.

Structure of the chapter. We first introduce the language of first-order session types, which is closely related to the language of [Gay and Hole, 2005], and define a sub-typing relation on the first-order session types.

In Section 2.2 we present the language of processes, and endow its terms with a structural operational semantics. Our processes are terms of infinitary CCS without τ's [De Nicola and Hennessy, 1987] enriched with the special term $\mathbf{1}$, which represents success. We will use the LTS denoted by processes in two chapters of the thesis.

It will become evident that there is a natural way to map first-order session types into a sub-LTS of the one denoted by processes. To render this mapping precisely, in Section 2.3 we introduce the language of session first-order *session contracts*. This languages is a bridge between the LTS of processes and the syntax of first-order session types. In particular, (a) first-order session contracts denote a sub-LTS of the one of processes, and (b) it is straightforward to prove that there is a bijection between the syntax of session types and session contracts.

2.1 The session type language

To define the syntax of types, we presupposes three denumerable sets; a set of labels $\mathtt{L}$, ranged over by $\mathtt{l}$, a set of ground types $\mathtt{BT}$ ranged over by $\mathtt{t}$, and a set of variables $\mathcal{V}$, ranged over by X; the last set let us express recursive types.

The syntax of terms for types is given by the grammar in Figure 2.1; let us denote with $L_{\mathsf{ST}_{\mathrm{fo}}}$ the language given by that grammar.

The use of variables leads to the usual notion of *free* and *bound* occurrences of variables in terms in the standard manner; we say that a term is *closed* if it contains no free variables. We also have the standard notion of *capture avoidance* substitution of terms for free variables. For the sake of clarity let us recall this definition: a substitution $\mathbf{s}$ is a mapping from the set $\mathcal{V}$ to the set of terms in $L_{\mathsf{ST}_{\mathrm{fo}}}$. Let

$$\mathbf{s} - X = \begin{cases} \mathbf{s} \setminus \{(X, \mathbf{s}(X))\} & \text{if } X \in dom(\mathbf{s}) \\ \mathbf{s} & \text{otherwise} \end{cases}$$

R, S, T	$::=$		**Session types**
		END	*Terminated session*
		$?[\,\mathtt{t}\,]; S$	*Input*
		$![\,\mathtt{t}\,]; S$	*Output*
		$\&\langle\, \mathtt{l}_1 \colon S_1, \ldots, \mathtt{l}_n \colon S_n \,\rangle$	*Branch*
		$\oplus\langle\, \mathtt{l}_1 \colon S_1, \ldots, \mathtt{l}_n \colon S_n \,\rangle$	*Choice*
		X	*Type variable*
		$\mu X.\, S$	*Recursive session type*

Where $n > 0$, and $i \neq j$ implies that $\mathtt{l}_i \neq \mathtt{l}_j$.

Figure 2.1: Grammar for first-order session types

$$S\mathbf{s} = \begin{cases} \text{END} & \text{if } S = \text{END} \\ \mathbf{s}(X) & \text{if } S = X, \text{ and } X \in dom(S) \\ X & \text{if } S = X, \text{ and } X \notin dom(S) \\ ![\,\mathtt{t}\,]; (S'\mathbf{s}) & \text{if } S = ![\,\mathtt{t}\,]; S' \\ ?[\,\mathtt{t}\,]; (S'\mathbf{s}) & \text{if } S = ?[\,\mathtt{t}\,]; S' \\ \&\langle\, \mathtt{l}_1 : (S_1\mathbf{s}), \ldots, \mathtt{l}_n : (S_n\mathbf{s}) \,\rangle & \text{if } S = \&\langle\, \mathtt{l}_1 : S_1, \ldots, \mathtt{l}_n : S_n \,\rangle \\ \oplus\langle\, \mathtt{l}_1 : (S_1\mathbf{s}), \ldots, \mathtt{l}_n : (S_n\mathbf{s}) \,\rangle & \text{if } S = \oplus\langle\, \mathtt{l}_1 : S_1, \ldots, \mathtt{l}_n : S_n \,\rangle \\ \mu X.\, (S'(\mathbf{s} - X)) & \text{if } S = \mu X.\, S' \\ \mathtt{t} & \text{if } S = \mathtt{t} \end{cases}$$

Figure 2.2: Application of substitution to first-order session types.

The result of applying a substitution $\mathbf{s}$ to the term S is defined in Figure 2.2, by structural induction.

In the final clause of the definition, the application of $\mathbf{s} - X$ embodies the idea that in $\mu X.\, S'$ occurrences of X in the sub-term S' are bound, and therefore substitutions have no effect on them.

It is easy to check that the effect of a substitution depends only on free variables; that is, $S\mathbf{s}_1 = S\mathbf{s}_2$ whenever $\mathbf{s}_1(X) = \mathbf{s}_2(X)$ for every free variable X occurring in S. We use $\left\{ {}^{T}/_{X} \right\}$ to denote the singleton substitution $\{(X, T)\}$.

In the language $L_{\mathsf{ST}_{\text{fo}}}$ we have recursive terms, so we introduce a way to unfold them. We formalise the notion of unfolding, which we define *inductively*.

Notation If X denotes a set, then we let $\mathcal{P}(X)$ denote the powerset of X; that is $\mathcal{P}(X)$ is the set of subsets of X.

Definition 2.1.1. [Unfolding]
Let $\mathcal{F}_{\text{UNF}} : \mathcal{P}(L^2_{\mathsf{ST}_{\text{fo}}}) \longrightarrow \mathcal{P}(L^2_{\mathsf{ST}_{\text{fo}}})$ be the rule functional given by the inference rules in Figure 2.3. We denote the least fixed point of $\mathcal{F}_{\text{UNF}}$ with the symbol UNF, and we refer to it as the *unfold* function. □

The relation UNF is a partial function, so we write $\text{UNF}(S) = T$ in place of $S\text{UNF}(T)$.

$$\frac{}{T \text{ UNF } T}\ T \neq \mu Z.\, S;\ [\text{UNF-A}] \qquad \frac{T' \left\{ {}^{\mu X.\, T}/_{X} \right\} \text{ UNF } S}{T \text{ UNF } S}\ T = \mu X.\, T';\ [\text{UNF-R}]$$

Figure 2.3: Inference rules for the rule functional $\mathcal{F}_{\text{UNF}}$

$$\frac{}{T \ depth \ 0} \ T \neq \mu Z.\, S \qquad \frac{T' \left\{ {}^{\mu Y.\, T}/_{Y} \right\} \ depth \ n}{\mu Y.\, T \ depth \ 1 + n} \ T = \mu Z.\, T'$$

Figure 2.4: Inference rules to compute the *depth* of *closed* terms

Intuitively, $\text{UNF}(T)$ unfolds top-level recursive terms until a type constructor appears, which is not μ. This will be extremely useful in manipulating session types. Not all the terms can be unfolded, for instance $\mu X.\, X$ cannot be unfolded.

The fact that we can unfold a closed term of $L_{\mathsf{ST}_{\text{fo}}}$ does not imply that also its sub-terms can be unfolded.

Example 2.1.2. [Unfolding and sub-terms]
Let $T = \&\langle \texttt{moka}\colon \mu X.\, X \rangle$. On the one hand, the term T is closed and the top-most constructor in it is not a recursion, thus we can prove that $\text{UNF}(T) = T$. The proof of this is given by the axiom of Figure 2.3:

$$\frac{}{T \ \text{UNF} \ T} \ T \neq \mu Z.\, S$$

On the other hand, a sub-term of T is $\mu X.\, X$, and it can not be unfolded. □

We will need to deal only with types whose subterms can be unfolded. To rule out terms that do not satisfy this property we introduce guarded recursion, which we now explain formally.

Definition 2.1.3. [Type term depth]
Let $\mathcal{F}_{depth} : \mathcal{P}(L^2_{\mathsf{ST}_{\text{fo}}}) \longrightarrow \mathcal{P}(L^2_{\mathsf{ST}_{\text{fo}}})$ be the rule functional given by the inference rules in Figure 2.4. We denote with *depth* the least fixed point of the rule functional $\mathcal{F}_{depth}$, and we refer to it as the *depth* function. □

The function *depth* from terms to $\mathbb{N}$ provides a measure of session types over which we can perform induction. Moreover, the depth of a closed term is defined if and only if the term can be unfolded.

Lemma 2.1.4. For every closed $T \in L_{\mathsf{ST}_{\text{fo}}}$, $depth(T) \in \mathbb{N}$ if and only if $\text{UNF}(T) = S$ for some S.

Proof. We are required to prove two implications,

a) if $depth(T) \in \mathbb{N}$ then $\text{UNF}(T) = S$ for some S

b) if $\text{UNF}(T) = S$ for some S then $depth(T) \in \mathbb{N}$

The proof of both implications are by rule induction. □

In [Gay and Hole, 2005] the function UNF is defined *co-inductively*. This seems to contrast with our inductive definition. In Example 2.1.5 we show that a co-inductive definition does not give rise to a function. In Proposition 2.1.6 we explain how the co-inductive definition can lead to the inductive notion of unfolding: when reasoning on terms with finite depth there is no difference between the inductive and the co-inductive definitions of UNF.

Example 2.1.5. Let $S = \mu X.\, X$. In this example we show that the pairs $(S,\, ?[\,\texttt{Int}\,];\text{END})$ and $(S,\, \&\langle \texttt{l}\colon \text{END} \rangle)$ are in the fixed point $\nu X.\mathcal{F}_{\text{UNF}}(X)$. Thanks to the Knaster-Tarski theorem all we have to do is to exhibit a prefixed point of $\mathcal{F}_{\text{UNF}}$ which contains the pairs at hand.

Consider the ensuing relation

$$\mathcal{R} = \{\, (S,\, ?[\,\texttt{Int}\,];\text{END}),\, (S,\, \&\langle \texttt{l}\colon \text{END} \rangle) \,\}$$

We prove that $\mathcal{R} \subseteq \mathcal{F}_{\text{UNF}}(\mathcal{R})$; to this aim, we have to to show that each pair in $\mathcal{R}$ can be derived by instantiating one of the inference rules in Figure 2.3. As $X \left\{ {}^{S}/_{X} \right\} = S$, we have the derivations

$$\frac{S \text{ UNF } ?[\,\texttt{Int}\,];\text{END}}{S \text{ UNF } ?[\,\texttt{Int}\,];\text{END}}\ [\text{UNF-R}] \qquad \frac{S \text{ UNF } \&\langle\,\texttt{l}\colon \text{END}\,\rangle}{S \text{ UNF } \&\langle\,\texttt{l}\colon \text{END}\,\rangle}\ [\text{UNF-R}]$$

Thus $\mathcal{R}$ is a prefixed point of UNF. The Knaster-Tarski theorem ensures that the pairs at hand are in the fixed point $\nu X.\mathcal{F}_{\text{UNF}}(X)$. □

The intuition behind the previous example is that if the depth of the first element of a pair $(T,\, S)$ is not defined, then the axiom [UNF-A] is not necessary to derive the pair itself.

Lemma 2.1.6. If $\mathit{depth}(T) \in \mathbb{N}$ and $(T,\, S) \in \nu X.\mathcal{F}_{\text{UNF}}(X)$, then $\text{UNF}(T) = S$.

Proof. We reason by induction on $\mathit{depth}(T) \in \mathbb{N}$.

If $\mathit{depth}(T) = 0$, then $T \neq \mu X.\, T'$, thus we can derive

$$\frac{}{T \text{ UNF } T}\ T \neq \mu X.\, T';\ [\text{UNF-A}]$$

This proves that $\text{UNF}(T) = T$.

In the inductive case $\mathit{depth}(T) = n+1$, thus $T = \mu X.\, T'$, and the hypothesis $(T,\, S) \in \nu X.\mathcal{F}_{\text{UNF}}(X)$ implies that there exists the derivation

$$\frac{\dfrac{}{T' \left\{ {}^{T}/_{X} \right\} \text{ UNF } S}}{T \text{ UNF } S}\ [\text{UNF-R}]$$

This proves that $(T' \left\{ {}^{T}/_{X} \right\},\, S) \in \nu X.\mathcal{F}_{\text{UNF}}(X)$. Since $\mathit{depth}(T' \left\{ {}^{T}/_{X} \right\}) = n$, we can apply the inductive hypothesis, and state that $\text{UNF}(T' \left\{ {}^{T}/_{X} \right\}) = S$; this means that there exists a finite derivation

$$\frac{\vdots}{T' \left\{ {}^{T}/_{X} \right\} \text{ UNF } S}$$

It follows that also the following derivation is finite

$$\frac{\dfrac{\vdots}{T' \left\{ {}^{T}/_{X} \right\} \text{ UNF } S}}{T \text{ UNF } S}\ [\text{UNF-R}]$$

so $\text{UNF}(T) = S$. □

To make sure that all the sub-terms of a terms T can be unfolded, we introduce the predicate GD.

Definition 2.1.7. [Guarded term]
A term T is *guarded*, $T\,\mathsf{gd}$ if every sub-term of the form $\mu X.\, S$ satisfies $\mathit{depth}(S) \in \mathbb{N}$. □

In view of Definition 2.1.7, Example 2.1.2 shows that to have a depth and to be guarded are different properties. In Example 2.1.15 we will see why we need to ensure that all the sub-terms of a given term can be unfolded.

Definition 2.1.8. [Language of first-order session types]
Let ST_{fo} denote the set of closed guarded terms,

$$\mathsf{ST}_{\text{fo}} = \{\, T \in L_{\mathsf{ST}_{\text{fo}}} \mid T\,\mathsf{gd} \,\}$$

We refer to the elements in ST_{fo} as *first-order session types*. □

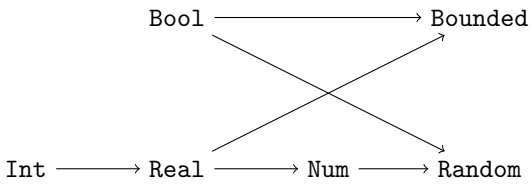


Figure 2.5: A sub-type relation on a set of basic types BT; the arrow represents the relation $\preccurlyeq_{\mathtt{b}}$ (see Example 2.1.10).

First-order session types afford the following properties.

Proposition 2.1.9. Let $\mathcal{T}$ be a session type,

a) the depth of T is finite

b) the unfolding of T is a session type

Proof. The first point follows from the definition of $\mathsf{ST}_{\mathrm{fo}}$ and Definition 2.1.7. To prove the second point, we have to show that $\textsc{unf}(T)$ is closed and guarded. The proof is by induction on $\mathit{depth}(T)$. It relies on the fact that each step of unfolding replaces one variable with a closed and guarded term; hence the overall unfolding is closed. □

2.1.1 Sub-typing

There are three sources for the sub-typing relation over types. The first is some predefined pre-order over the base types, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$, which intuitively says that all data-values of type $\mathtt{t}_1$ may be safely used where values of $\mathtt{t}_2$ are expected.

Example 2.1.10. An example of sub-typing on base types is given in Figure 2.5, for the ensuing set of types, $\mathtt{BT} = \{\, \mathtt{Bounded}, \mathtt{Bool}, \mathtt{Int}, \mathtt{Real}, \mathtt{Num}, \mathtt{Random} \,\}$. In the figure the pre-order $\preccurlyeq_{\mathtt{b}}$ is depicted by the arrows; for instance, the arrow from type `Int` to type `Real` means that $\mathtt{Int} \preccurlyeq_{\mathtt{b}} \mathtt{Real}$. □

More generally, if $[\![\mathtt{t}]\!]$ denotes the set of values of the base type $\mathtt{t}$ then we can define $\preccurlyeq_{\mathtt{b}}$ by letting $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ whenever $[\![\mathtt{t}_1]\!] \subseteq [\![\mathtt{t}_2]\!]$. The other sources for the sub-typing are two constructs of the language: the *branch* construct allows sub-typing by extending the set of labels involved, while in the *choice* construct the set of labels may be restricted. We give two examples.

Example 2.1.11. [Sub-typing on branch types]
In this example we explain how the sub-typing relates the branch types. Consider the type

$$\textsc{BarTender} = \&\langle\, \mathtt{espresso}\colon T_1 \,\rangle$$

Intuitively, the BarTender offers only the label `espresso`, thus all the customers satisfied by BarTender, are satisfied by any other type that offers *at least* the label `espresso`. Let

$$\begin{aligned}\textsc{ItalianBarTender} = \&\langle\, & \mathtt{espresso}\colon T'_1,\\ & \mathtt{deka}\colon T'_2,\\ & \mathtt{double}-\mathtt{deka}-\mathtt{restr}\colon T'_3,\\ & \mathtt{double}-\mathtt{espresso}\colon T'_4 \,\rangle\end{aligned}$$

Following the intuition, The ItalianBarTender will satisfy all the customers satisfied by the BarTender; this is formalised by the sub-typing, which relates the two types as follows

$$\textsc{BarTender} \preccurlyeq_{\mathsf{sbt}} \textsc{ItalianBarTender}$$

as long as also the continuations T_1 and T_1' are related as well (ie. $T_1 \preccurlyeq_{\mathsf{sbt}} T_1'$).

We have shown that, intuitively, it is safe to replace a branch type with a branch type that offers more labels. □

Example 2.1.12. [Sub-typing on choice types]
In this example we show how the sub-typing relate the choice types. Let ItalianCustomer describe the different coffees that a process may want to order when interacting with a bar tender.

$$\begin{aligned}\textsc{ItalianCustomer} = \oplus\langle\, &\texttt{espresso}\colon T_1',\\ &\texttt{deka}\colon T_2',\\ &\texttt{double}-\texttt{deka}-\texttt{restr}\colon T_3',\\ &\texttt{double}-\texttt{espresso}\colon T_4'\,\rangle\end{aligned}$$

All the bar tenders that are able to satisfy this range of choices, have to offer *at least* the four labels that appear in ItalianCustomer. Now consider the type

$$\textsc{Customer} = \oplus\langle\, \texttt{espresso}\colon T_1'\,\rangle$$

Since Customer chooses among fewer options than ItalianCustomer, it is safe to use a channel at type Customer in place of a channel at type ItalianCustomer. This is formalised by the sub-typing relation as follows,

$$\textsc{ItalianCustomer} \preccurlyeq_{\mathsf{sbt}} \textsc{Customer}$$

In this example we have shown that, intuitively, it is safe to replace a choice type, with a choice type that chooses among fewer labels. □

Moreover, we will have the standard co-variance/contra-variance of input/output types [Pierce and Sangiorgi, 1996], extended to both the *branch* and *choice* constructs.

Because of the recursive nature of our collection of types, the formal definition of the sub-typing relation is given co-inductively.

Definition 2.1.13. [First-order sub-typing]
Let $\mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}} : \mathcal{P}(\mathsf{ST}_{\mathrm{fo}}{}^2) \longrightarrow \mathcal{P}(\mathsf{ST}_{\mathrm{fo}}{}^2)$ be the rule functional given by the inference rules in Figure 2.6. If $X \subseteq \mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}(X)$, then we say that X is a *first-order type simulation*. Lemma C.0.17 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}(X)$; we call this solution the *first-order sub-typing relation*, and we denote it $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$. That is $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} = \nu X.\mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}(X)$. □

Figure 2.6 contains a schema of inference rules rather than a set of finite inference rules. This because the premises of the rules [r-Branch] and [r-Choice] depend on the cardinality of two sets, respectively I and J. The schema gives rise to an infinite amount of inference rules. An infinite set of inference rules gives rise to a rule functional as a finite set of rules does. The details of the construction are explained in [Sangiorgi, 2012, Chapter 2].

Example 2.1.14. [Side conditions on *depth* of terms]
In this example we explain why rule [r-Unfold] in Figure 2.6 has the side condition $depth(S_1) +$

$$\frac{}{\text{END} \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \text{END}}\ [\text{A-END}]$$

$$\frac{S_1' \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2'}{?[\,\mathtt{t}_1\,]; S_1' \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} ?[\,\mathtt{t}_2\,]; S_2'}\ \mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2;\ [\text{R-IN}]$$

$$\frac{S_1' \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2'}{![\,\mathtt{t}_1\,]; S_1' \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} ![\,\mathtt{t}_2\,]; S_2'}\ \mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1;\ [\text{R-OUT}]$$

$$\frac{S_1^1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_1^2 \quad \ldots \quad S_m^1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_m^2}{\&\langle\, \mathtt{l}_1 : S_1^1, \ldots l_m : S_m^1 \,\rangle \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \&\langle\, \mathtt{l}_1 : S_1^2, \ldots, \mathtt{l}_n : S_n^2 \,\rangle}\ m \leq n;\ [\text{R-BRANCH}]$$

$$\frac{S_1^1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_1^2 \quad \ldots \quad S_n^1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_n^2}{\oplus\langle\, \mathtt{l}_1 : S_1^1, \ldots l_m : S_m^1 \,\rangle \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \oplus\langle\, \mathtt{l}_1 : S_1^2, \ldots, \mathtt{l}_n : S_n^2 \,\rangle}\ n \leq m;\ [\text{R-CHOICE}]$$

$$\frac{\text{UNF}(S_1) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \text{UNF}(S_2)}{S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2}\ \mathit{depth}(S_1) + \mathit{depth}(S_2) > 0;\ [\text{R-UNFOLD}]$$

Figure 2.6: Inference rules for the rule functional $\mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}$

$\mathit{depth}(S_2) > 0$. Intuitively, the side condition ensures that the rules cannot be applied vacuously, that is to terms that equal their unfoldings.

Let $S_1 = ![\,\mathtt{Int}\,]; \text{END}$ and $S_2 = ?[\,\mathtt{Bool}\,]; \text{END}$, and let $\preccurlyeq_{\mathsf{bad}}$ be the relation defined as $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$, but without the side condition in rule [R-UNFOLD]. We prove that the bad sub-typing $\preccurlyeq_{\mathsf{bad}}$ relates S_1 to S_2. We have to show a prefixed point of $\mathcal{F}^{\preccurlyeq_{\mathsf{bad}}}$ that contains the pair (S_1, S_2); let $\mathcal{R} = \{\,(S_1, S_2)\,\}$. To prove that $\mathcal{R} \subseteq \mathcal{F}^{\preccurlyeq_{\mathsf{bad}}}(\mathcal{R})$ we need to show just that $(S_1, S_2) \in \mathcal{F}^{\preccurlyeq_{\mathsf{bad}}}(\mathcal{R})$. This follows form the equalities $\text{UNF}(S_1) = S_1$, $\text{UNF}(S_2) = S_2$, and an application of rule [R-UNFOLD]:

$$\frac{\text{UNF}(S_1) \preccurlyeq_{\mathsf{bad}} \text{UNF}(S_2)}{S_1 \preccurlyeq_{\mathsf{bad}} S_2}$$

Note that thanks to the side condition $\mathit{depth}(S_1) + \mathit{depth}(S_2) > 0$ we cannot prove $S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2$. □

The requirement that session types be guarded is crucial for the first-order sub-typing relation to be well-defined. We explain this fact in the next example.

Example 2.1.15. [Sub-typing and guardedness]
In this example we show why the guardedness of terms is required in the definition of session types. Consider again the term $T = \&\langle\, \mathtt{l} : \mu X.\, X \,\rangle$ of Example 2.1.2. Suppose we wanted to check whether $T \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \&\langle\, \mathtt{l} : S \,\rangle$ for some term S. The definition of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ requires us to check whether

$$\text{UNF}(\mu X.\, X) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \text{UNF}(S)$$

This check, though, can not be done because $\text{UNF}(\mu X.\, X)$ is *not* defined at all (and $\text{UNF}(S)$ may not be defined either). □

The co-inductive type simulations are closed with respect to unfolding.

Lemma 2.1.16. [$\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ and unfolding]
For every co-inductive type simulation $\mathcal{R}$, and every $S, T \in \mathsf{ST}_{\mathrm{fo}}$, if $S\ \mathcal{R}\ T$ then $\text{UNF}(T)\ \mathcal{R}\ \text{UNF}(T)$.

Proof. Let $S\ \mathcal{R}\ T$. The argument depends on the depths of S and T. If $\mathit{depth}(S) + \mathit{depth}(T) = 0$, then $\text{UNF}(S) = S$ and $\text{UNF}(T) = T$, so the assumption $S\ \mathcal{R}\ T$ implies $\text{UNF}(S)\ \mathcal{R}\ \text{UNF}(T)$. If $\mathit{depth}(S) + \mathit{depth}(T) > 0$, then recall that by hypothesis $\mathcal{R} \subset \mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}(\mathcal{R})$. It follows that $(S, T) \in \mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}(\mathcal{R})$. The

assumption on the depths of the types at hand ensures that the only way to prove $(S, T) \in \mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}(\mathcal{R})$ is an application of rule [R-UNFOLD],

$$\frac{\textsc{unf}(S) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \textsc{unf}(T)}{S \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} T} \; \mathit{depth}(S) + \mathit{depth}(T) > 0; \; [\text{R-UNFOLD}]$$

It follows that $\textsc{unf}(S) \; \mathcal{R} \; \textsc{unf}(T)$. □

Proposition 2.1.17. The relation $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ is a pre-order on $\mathsf{ST}_{\mathrm{fo}}$.

Proof. See [Gay and Hole, 2005]. □

In [Gay and Hole, 2005] the set of types $\mathsf{ST}_{\mathrm{fo}}$ are used to give a typing system for the pi calculus, and appropriate Type Safety and Type Preservation theorems are proved.

In Chapter 6 our aim will be to find a pre-order isomorphic to $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$, which is based on the semantics of terms rather than their syntax. In particular, as session types are meant to be protocol descriptions, a natural decision is to try to map session types into CCS without τ's, and use the operational semantics of this language to define a fully abstract model for $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$. We will achieve this result in Theorem 6.3.4.

In the next two sections we introduce respectively, an infinitary version of CCS without τ's; and then a language that lets us focus on a sub-LTS of the general one given by $\mathsf{CCS}_{\mathrm{w}\tau}$, and interpret session types via a bijection.

2.2 Processes

In this section we define an LTS of processes by means of a language and its operational semantics. The language is an infinitary version of $\mathsf{CCS}_{\mathrm{w}\tau}$, and the operational semantics of the terms is specified by standard inference rules.

Let Act be a set of actions, ranged over by $\alpha, \beta, \gamma, \ldots$ and let τ, $\checkmark$ be two distinct actions *not* in Act; the first will denote internal unobservable activity while the second will be used to report the success of an experiment. To emphasise their distinctness we use $\mathsf{Act}_{\tau\checkmark}$ to denote the set $Act \cup \{\tau, \checkmark\}$, Act_{τ} to denote the set $Act \cup \{\tau\}$, and $\mathsf{Act}_{\checkmark}$ to denote the set $Act \cup \{\checkmark\}$. We assume Act has an idempotent complementation function, with $\overline{\alpha}$ being the complement to α.

We let $\alpha_{\checkmark}$ to range over $\mathsf{Act}_{\checkmark}$, α_{τ} range over Act_{τ}, and μ range over $\mathsf{Act}_{\tau\checkmark}$.

Labelled Transition System A labelled transition system, LTS, consists of a triple

$$\langle P, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$$

where P is a set of processes and $\longrightarrow \subseteq P \times \mathsf{Act}_{\tau\checkmark} \times P$ is a transition relation between processes decorated with labels drawn from the set $\mathsf{Act}_{\tau\checkmark}$. We use the infix notation $p \xrightarrow{\mu} q$ in place of $(p, \mu, q) \in \longrightarrow$. An LTS is finite-branching if for all $p \in P$ and for all $\mu \in \mathsf{Act}_{\tau\checkmark}$, the set $\{ q \mid p \xrightarrow{\mu} q \}$ is finite. In this thesis we will not assume that all LTSs are finite-branching.

We use standard notation for operations in LTSs. For example $\mathsf{Act}^{\star}_{\tau\checkmark}$, ranged over by t, denotes the set of *finite* sequences of actions from the set $\mathsf{Act}_{\tau\checkmark}$, and for any $t \in \mathsf{Act}^{\star}_{\tau\checkmark}$ we let $p \xrightarrow{t} q$ be the obvious generalisation of the single transition relations to sequences. For an infinite sequence $u \in \mathsf{Act}^{\infty}_{\tau\checkmark}$ of the form $\mu_0\mu_1\ldots$ we write $p \xrightarrow{u}$ to mean that there is an infinite sequence of actions $p \xrightarrow{\mu_0} p_o \xrightarrow{\mu_1} p_1 \ldots$. The relation $\xrightarrow{t}$ is lifted to the weak case in the standard manner, using a function which projects a sequence $t \in \mathsf{Act}^{\star}_{\tau\checkmark}$ into a sequence $\langle t \rangle_{\setminus\tau} \in \mathsf{Act}^{\star}_{\checkmark}$, by ignoring all occurrences of τ. Then for $s \in \mathsf{Act}^{\star}_{\checkmark}$ we write $p \stackrel{s}{\Longrightarrow} q$ if $p \xrightarrow{t} q$ for some $t \in \mathsf{Act}^{\star}_{\tau\checkmark}$ such that $s = \langle t \rangle_{\setminus\tau}$. Similarly for $u \in \mathsf{Act}^{\infty}_{\checkmark}$ we write $p \stackrel{u}{\Longrightarrow}$ to mean $p \xrightarrow{t}$ for some $t \in \mathsf{Act}^{\infty}_{\tau\checkmark}$ such that for every finite prefix u_n of u there exists some prefix t_k of t such that $\langle t_k \rangle_{\setminus\tau} = u_n$.

p, q, r	$::=$	**Processes**
	$\mathsf{1}$	*Succesful process*
	A	*Definitional constant*
	$\alpha.p$	*Prefix*
	$p \mid\mid q$	*Parallel composition*
	$\sum_{i\in I} p_i$	*External choice*
	$\bigoplus_{j\in J} p_i$	*Internal choice*

where $\alpha \in Act$, I, J are countable index sets, with $J \neq \emptyset$. $A, B, C, \ldots$ range over a set of definitional constants each of which has an associated definition $A \stackrel{\mathsf{def}}{=} p_A$.

Figure 2.7: Syntax of infinitary $\mathsf{CCS}_{\mathrm{w}\tau}$.

Language and semantics To describe LTSs we use an infinitary version of CCS *without* τ*s*, [De Nicola and Hennessy, 1987], augmented with a *success* operator, $\mathsf{1}$.

Definition 2.2.1. Let $\mathsf{CCS}_{\mathrm{w}\tau}$ be the set of terms defined by the grammar in Figure 2.7. □

We use $\mathsf{0}$ to denote the empty external sum $\sum_{i\in\emptyset} p_i$, and $p_1 + p_2$ for the binary sum $\sum_{i\in\{1,2\}} p_i$. Similarly the binary internal sum $\bigoplus_{i\in\{1,2\}} p_i$ will be rendered as $p_1 \oplus p_2$.

Each term in the language $\mathsf{CCS}_{\mathrm{w}\tau}$ denotes an LTS; the easiest way to describe this interpretation is to define one overarching LTS, whose states are the terms in $\mathsf{CCS}_{\mathrm{w}\tau}$ and where the relations $p \xrightarrow{\mu} q$ are the least ones determined by the (standard) rules in Figure 2.8. Note the side condition in rule [P-SYNCH]; when reasoning on processes, we assume the relation $\bowtie$ to be given by the co-action function: $\alpha \bowtie \overline{\alpha}$.

In the next chapters we will need the following concepts.

Definition 2.2.2. [Computation]
For every process r and p a sequence of reductions

$$r \mid\mid p \xrightarrow{\tau} r_1 \mid\mid p_1 \xrightarrow{\tau} r_2 \mid\mid p_2 \longrightarrow \ldots$$

is called a *computation* of $r \mid\mid p$ and each derivative $r_i \mid\mid p_i$ is a *state* of the computation. □

Definition 2.2.3. [Convergent]
We write $p \Downarrow$ whenever there exists a $k \in \mathbb{N}$ such that if $p \xrightarrow{\tau}{}^{n} p'$, then $n \leq k$. If $p \Downarrow$ then we say that p *converges*. □

2.3 Session Contracts

In Section 2.2 we have define a general LTS given by the terms of $\mathsf{CCS}_{\mathrm{w}\tau}$ and their operational semantics.

Here we turn our attention to a sub-LTS of $\langle \mathsf{CCS}_{\mathrm{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$, that is denoted by the terms of yet another language, the language of *session contracts*. In this section we assume Act to be the set

$$\{\, ?\mathtt{b}, !\mathtt{b} \mid \mathtt{b} \in \mathtt{BT} \,\} \cup \{\, \mathtt{l}?, \mathtt{l}! \mid \mathtt{l} \in \mathtt{L} \,\}$$

The syntax for the language $L_{\mathsf{SC}_{\mathrm{fo}}}$ is given in Figure 2.9. The language $L_{\mathsf{SC}_{\mathrm{fo}}}$ is not a sublanguage of $\mathsf{CCS}_{\mathrm{w}\tau}$, because of recursion; the term $\mu x.\, \alpha.x$ is not in $\mathsf{CCS}_{\mathrm{w}\tau}$. Despite this apparent difference, it will become evident that the LTSs denoted by the two languages are related. For instance we will

$$\frac{}{1 \xrightarrow{\checkmark} 0}\ [\text{A-Ok}] \qquad \frac{}{\alpha.p \xrightarrow{\alpha} p}\ [\text{A-Pre}]$$

$$\frac{}{p \oplus q \xrightarrow{\tau} p}\ [\text{A-In-L}] \qquad \frac{}{p \oplus q \xrightarrow{\tau} q}\ [\text{A-In-R}]$$

$$\frac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'}\ [\text{R-Ext-L}] \qquad \frac{q \xrightarrow{\alpha} q'}{p + q \xrightarrow{\alpha} q'}\ [\text{R-Ext-R}]$$

$$\frac{p \xrightarrow{\tau} p'}{p + q \xrightarrow{\tau} p' + q}\ [\text{R-In-L}] \qquad \frac{q \xrightarrow{\tau} q'}{p + q \xrightarrow{\tau} p + q'}\ [\text{R-In-R}]$$

$$\frac{p_A \xrightarrow{\mu} p'}{A \xrightarrow{\mu} p'}\ A \stackrel{\texttt{def}}{=} p_A;\ [\text{R-Const}]$$

$$\frac{q \xrightarrow{\mu} q'}{q \mid\mid p \xrightarrow{\mu} q' \mid\mid p}\ [\text{P-Left}] \qquad \frac{p \xrightarrow{\mu} p'}{q \mid\mid p \xrightarrow{\mu} q \mid\mid p'}\ [\text{P-Right}]$$

$$\frac{q \xrightarrow{\alpha} q' \quad p \xrightarrow{\beta} p'}{q \mid\mid p \xrightarrow{\tau} q' \mid\mid p'}\ \alpha \bowtie \beta;\ [\text{P-Synch}]$$

Figure 2.8: The operational semantics of $\mathsf{CCS}_{\mathrm{w}\tau}$

see that the process $A \stackrel{\texttt{def}}{=} \alpha.A \oplus \alpha.A$ has the same operational semantics of $\mu x.\, \alpha.x$. To our aims the construct $\mu x.$ is not necerray; nevertheless, we introduce it to render as straightforward as possible the encoding of session types into session contracts.

As of now, we still have to formally define the language of session contracts. To do so, we make sure that session contracts can be unfolded.

Recursive definitions are handled in much the same way as we did for session types, and so we do not spell out all the details; we assume a definition of capture-avoiding substitution **s**. We define the functions *depth* and UNF as in Section 2.1, but using the inference rules on closed terms of $L_{\mathsf{SC}_{\mathrm{fo}}}$ instead of $L_{\mathsf{ST}_{\mathrm{fo}}}$.

Once again, one can prove the following lemma.

Lemma 2.3.1. For every closed $\sigma \in L_{\mathsf{SC}_{\mathrm{fo}}}$, $\mathit{depth}(\sigma) \in \mathbb{N}$ if and only $\text{UNF}(\sigma) = \sigma'$ for some σ'.

ρ, σ	::=		**First-order session contracts**
		1	*Satisfied contract*
		$?\mathtt{t}.\sigma$	*Input*
		$!\mathtt{t}.\sigma$	*Output*
		$\sum_{i \in I} ?\mathtt{l_i}.\sigma_i$	*External sum*
		$\bigoplus_{i \in I} !\mathtt{l_i}.\sigma_i$	*Internal sum*
		x	*Session contract variable*
		$\mu x.\, \sigma$	*Recursive session contract*

Where I is not empty, and $i \neq j$ implies that $\mathtt{l}_i \neq \mathtt{l}_j$.

Figure 2.9: Grammar for *first-order* session contracts

$$\frac{}{\mu x.\,\sigma \xrightarrow{\tau} \sigma\,\{{}^{\mu x.\,\sigma}/_{x}\}}\ [\text{A-UNF}]$$

Figure 2.10: Operational semantics of recursive session contracts

Proof. Analogous to the proof of Lemma 2.1.4. □

We extend to the closed terms of $L_{\mathsf{SC}_{\text{fo}}}$ also the definition of the predicate $\mathtt{gd}$; for instance the term $\mu x.\,\alpha.\mu x.\,x$ is not guarded.

Definition 2.3.2. [Language of session contracts]
Let $\mathsf{SC}_{\text{fo}} = \{\,\sigma \in L_{\mathsf{SC}_{\text{fo}}} \mid \sigma\,\mathtt{gd}\,\}$. We refer to the terms in the set SC_{fo} as *session contracts*. □

Thanks to the requirement that the σ's be guarded, one can prove that for every $\sigma \in \mathsf{SC}_{\text{fo}}$, $\sigma \Downarrow$.

Note that in session contracts

- external choices are restricted to *inputs* on labels
- internal choices are restricted to *outputs* on labels

Note also that $\mathtt{0}$ is not a session contract. Instead we have chosen $\mathtt{1}$ to be the base contract, for reasons which will become apparent.

The operational semantics of session contracts is given by the rules in Figure 2.8 with the additional rule for recursive terms given in Figure 2.10.

Also, we instantiate the relation $\bowtie$ of rule [P-SYNCH] to $\bowtie_c$, with $\bowtie_c$ determined by

$$\alpha \bowtie_c \beta \qquad \text{whenever} \begin{cases} \alpha = ?\mathtt{b},\ \beta = !\mathtt{b}' & \mathtt{b}' \preccurlyeq_{\mathtt{b}} \mathtt{b} \\ \alpha = !\mathtt{b},\ \beta = ?\mathtt{b}' & \mathtt{b} \preccurlyeq_{\mathtt{b}} \mathtt{b}' \\ \alpha = ?\mathtt{l},\ \beta = !\mathtt{l} & \\ \alpha = !\mathtt{l},\ \beta = ?\mathtt{l} & \end{cases}$$

Using the basic sub-typing relation depicted in Figure 2.5 the following examples should be clear:

1. $?\mathtt{Num} \bowtie_c !\mathtt{Int}$: a contract that can read a datum of type `Num` can read a datum of type `Int` because $\mathtt{Int} \preccurlyeq_{\mathtt{b}} \mathtt{Num}$.
2. $?\mathtt{Int} \not\bowtie_c !\mathtt{Num}$: conversely a contract ready to read a datum of type `Int` cannot read a datum of type `Num` because $\mathtt{Num} \not\preccurlyeq_{\mathtt{b}} \mathtt{Int}$.
3. $?\mathtt{Random} \bowtie_c !\mathtt{Bool}$: as in point (i), $\mathtt{Bool} \preccurlyeq_{\mathtt{b}} \mathtt{Random}$ hence an interaction between the actions $?\mathtt{Random}$ and$!\mathtt{Bool}$ can take place.

In view of the definition of *Act* and of the previous example we also introduce a way to compare sets of action, that mirrors the impact of $\preccurlyeq_{\mathtt{b}}$ on $\bowtie_c$.

Definition 2.3.3. Let A and B be subset of *Act*; we write $A \sqsubseteq_{\text{RS}} B$ whenever for every $\alpha_A \in A$ and every β such that $\beta \bowtie_c \alpha_A$ there exists some action $\alpha_B \in B$ such that $\beta \bowtie_c \alpha_B$ also. □

Example 2.3.4. In this example we show how the $\sqsubseteq_{\text{RS}}$ relates sets of actions. It turns out that $A \sqsubseteq_{\text{RS}} B$ whenever the following conditions are true,

- $!\mathtt{l} \in A$ implies $!\mathtt{l} \in B$ for every $\mathtt{l} \in \mathtt{L}$
- $?\mathtt{l} \in A$ implies $?\mathtt{l} \in B$ for every $\mathtt{l} \in \mathtt{L}$

- $?\mathtt{b}_A \in A$ implies $?\mathtt{b}_B \in B$ for some type $\mathtt{b}_B$ such that $\mathtt{b}_B \preccurlyeq_{\mathtt{b}} \mathtt{b}_A$
- $!\mathtt{b}_A \in A$ implies $!\mathtt{b}_B \in B$ for some type $\mathtt{b}_B$ such that $\mathtt{b}_A \preccurlyeq_{\mathtt{b}} \mathtt{b}_B$ □

We recall some examples of session contracts from the literature.

Example 2.3.5. [e-vote, [Barbanera and de'Liguoro, 2010; Laneve and Padovani, 2008]]

$$\text{Ballot} \quad = \quad \mu x.\, ?\mathtt{Login}.(\, !\mathtt{Wrong}.x \oplus !\mathtt{Ok}.(\, ?\mathtt{VoteA}.x + ?\mathtt{VoteB}.x\,)\,)$$

$$\text{Voter} \quad = \quad \mu x.\, !\mathtt{Login}.(\, ?\mathtt{Wrong}.x + ?\mathtt{Ok}.(\, !\mathtt{VoteA}.\,\mathtt{1} \oplus !\mathtt{VoteB}.\,\mathtt{1}\,)\,)$$

The session contract Ballot describes a service for e-voting. Such a service lets a client log in. If the log in fails the services starts anew, while if the log in succeeds the two actions are offered to the environment, namely `VoteA` and `VoteB`.

The contract Voter is a recursive client for the protocol described by the contract Ballot. □

Example 2.3.6. [e-commerce, [Bernardi et al., 2008]]

$$\begin{aligned}
\text{Customer} \quad &= \quad !\mathtt{Request}.(\, !\mathtt{PayDebit}.\rho' \oplus \\
&\qquad\qquad\quad !\mathtt{PayCredit}.\rho' \oplus \\
&\qquad\qquad\quad !\mathtt{PayCash}.\,\mathtt{1}\,) \\
\rho' \quad &= \quad !\mathtt{Long}.?\mathtt{Bool}.\,\mathtt{1} \\
\text{Bank} \quad &= \quad \mu x.\, ?\mathtt{Request}.(\, ?\mathtt{PayCredit}.?\mathtt{Long}.!\mathtt{Bool}.x + \\
&\qquad\qquad\quad ?\mathtt{PayDebit}.?\mathtt{Long}.!\mathtt{Bool}.x + \\
&\qquad\qquad\quad ?\mathtt{PayCash}.x\,)
\end{aligned}$$

The session contracts above describe the conversation that should take place between a client (described by the session contract Customer) and a bank (described by the session contract Bank) involved in an on-line payment. The conversation unfolds as follows: the Customer sends a request to the bank and afterwards it chooses the payment method; the choice is taken by an internal sum and this means that the decision of the Customer is independent from the environment (i.e., the Bank contract). If the Customer decides to pay by cash then no other action has to be taken; while if the payment is done by debit or credit card the Customer has to send the card number, this is represented by the output `!Long`. After the card number has been received the Bank answers with a boolean. Intuitively, this represents the fact that the bank can approve or reject the payment. The Customer protocol finishes after such boolean has been received, while the Bank starts anew. □

Session contracts, due to the their restrictive syntax, enjoy some properties that are fundamental to prove the results of Section 3.3 and Chapter 6. We prove these properties.

Lemma 2.3.7. Let σ be a contract.

(i) If $\sigma \overset{\tau}{\not\longrightarrow}$ then $\text{UNF}(\sigma) = \sigma$

(ii) $\sigma \overset{\tau}{\Longrightarrow} \text{UNF}(\sigma)$

Proof. Property (i) is proved by structural induction on σ;

We prove point (ii); the argument is by induction on $\mathit{depth}(\sigma)$. If $\mathit{depth}(\sigma) = 0$ then from the definition of *depth* it follows that $\sigma \neq \mu x.\, \sigma'$; by definition of UNF then $\text{UNF}(\sigma) = \sigma$. The reflexivity of $\overset{\tau}{\Longrightarrow}$ implies $\sigma \overset{\tau}{\Longrightarrow} \text{UNF}(\sigma)$.

If $\mathit{depth}(\sigma) > 1$ then, due to the definition of *depth*, $\sigma = \mu x.\, \sigma'$. The definition of UNF implies that $\text{UNF}(\sigma) = \text{UNF}(\sigma' \{ {}^{\sigma}/_{x} \})$, while the definition of *depth* implies $\mathit{depth}(\sigma) = 1 + \mathit{depth}(\sigma' \{ {}^{\sigma}/_{x} \})$,

and therefore $depth(\sigma'\{^{\sigma}/_{x}\})$ is smaller than $depth(\sigma)$. We are now allowed to use the inductive hypothesis on $\sigma'\{^{\sigma}/_{x}\}$:

$$\sigma'\{^{\sigma}/_{x}\} \stackrel{\tau}{\Longrightarrow} \textsc{unf}(\sigma'\{^{\sigma}/_{x}\})$$

We use rule [A-UNF] (see Figure 2.10) to infer $\sigma \stackrel{\tau}{\longrightarrow} \sigma'\{^{\sigma}/_{x}\}$, and then the transitivity of $\stackrel{\tau}{\Longrightarrow}$ to obtain

$$\sigma \stackrel{\tau}{\Longrightarrow} \textsc{unf}(\sigma'\{^{\sigma}/_{x}\})$$

We already know that $\textsc{unf}(\sigma) = \textsc{unf}(\sigma'\{^{\sigma}/_{x}\})$, and, by applying this equality to the reduction sequence above, we get

$$\sigma \stackrel{\tau}{\Longrightarrow} \textsc{unf}(\sigma)$$

This concludes the proof. □

A statement weaker than the converse of point (ii) of Lemma 2.3.7 is true.

Lemma 2.3.8. For every $\sigma \in \mathsf{SC}_{\mathrm{fo}}$, if $\sigma \stackrel{\tau}{\longrightarrow}^{n} \hat{\sigma}$. Let $k = depth(\sigma)$. If $n \geq k$ then $\sigma_k = \textsc{unf}(\sigma)$, where the reduction sequence above is $\sigma \stackrel{\tau}{\longrightarrow}^{k} \sigma_k \stackrel{\tau}{\longrightarrow}^{n-k} \hat{\sigma}$.

Proof. We reason by induction on k.

Base case ($k = 0$) If $k = 0$ then σ has not top-most μ, so $\textsc{unf}(\sigma) = \sigma$, and indeed $\sigma_0 = \sigma$.

Inductive case ($k = k' + 1$) In this case $\sigma = \mu x.\,\sigma'$ for some variable x and σ', and $\sigma \stackrel{\tau}{\longrightarrow} \sigma_1 \stackrel{\tau}{\longrightarrow}^{n_1} \hat{\sigma}$. The only way to derive $\sigma \stackrel{\tau}{\longrightarrow} \sigma_1$ is by applying [A-UNF], so $\sigma_1 = \sigma'\{^{\sigma}/_{x}\}$. The definition of $depth()$ ensures that $depth(\sigma_1) = k'$. Since $\sigma_1 \stackrel{\tau}{\longrightarrow}^{n-1} \hat{\sigma}$, and $n \geq k$ ensures $n - 1 \geq k - 1$, the inductive hypothesis implies that $\sigma_1 \stackrel{\tau}{\longrightarrow}^{k-1} \textsc{unf}(\sigma_1) \stackrel{\tau}{\longrightarrow}^{n-k} \hat{\sigma}$. Since the definition of UNF implies that $\textsc{unf}(\sigma) = \textsc{unf}(\sigma_1)$, it follows that It follows that the original reduction sequence is $\sigma \stackrel{\tau}{\longrightarrow}^{k} \textsc{unf}(\sigma) \stackrel{\tau}{\longrightarrow}^{n-k} \hat{\sigma}$. □

Lemma 2.3.9. Let σ be a session contract. Then

(i) $\sigma \stackrel{\checkmark}{\longrightarrow}$ if and only if $\sigma = \mathbf{1}$

(ii) $\sigma \Longrightarrow\!\!\stackrel{\checkmark}{\longrightarrow}$ if and only if $\textsc{unf}(\sigma) = \mathbf{1}$

Proof. Part (i) follows from the restrictive syntax of session contract. The proof of part (ii) requires two arguments. The *if* side, $\textsc{unf}(\sigma) = \mathbf{1}$ implies $\sigma \Longrightarrow\!\!\stackrel{\checkmark}{\longrightarrow}$, is justified by part (ii) of Lemma 2.3.7. The *only if* side, $\sigma \Longrightarrow\!\!\stackrel{\checkmark}{\longrightarrow}$ implies $\textsc{unf}(\sigma) = \mathbf{1}$, can be proven by induction on the length of the sequence $\Longrightarrow$; the base case being part (i) of this lemma. □

Lemma 2.3.10. If $\sigma \stackrel{\alpha}{\longrightarrow}$ then $\sigma \stackrel{\checkmark}{\not\longrightarrow}$.

Interpreting session types into session contracts

Session contracts play a key role in giving a semantics to session types; this is true because session contracts denote the LTS $\langle \mathsf{SC}_{\mathrm{fo}}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$, and there is a straightforward way to map session types into session contracts, and vice-versa. We define this interpretation, and prove its properties,

The next results will be crucial in Section 6.3. The interpretation of session types as contracts is expressed as a function from the language $L_{\mathsf{ST}_{\mathrm{fo}}}$ in Section 2.1 to the language $L_{\mathsf{SC}_{\mathrm{fo}}}$ in of Definition 2.3.2. The function is just a syntactic transformation.

Let $\mathcal{M} : L_{\mathsf{ST}_{\mathrm{fo}}} \longrightarrow L_{\mathsf{SC}_{\mathrm{fo}}}$ be defined by:

$$\mathcal{M}(S) = \begin{cases} \mathtt{1} & \text{if } S = \text{END} \\ !\mathtt{t}.\mathcal{M}(S) & \text{if } S = ![\,\mathtt{t}\,]; S \\ ?\mathtt{t}.\mathcal{M}(S) & \text{if } S = ?[\,\mathtt{t}\,]; S \\ \sum_{i\in[1;n]} ?\mathtt{l}_i.\mathcal{M}(S_i) & \text{if } S = \&\langle\, \mathtt{l}_1 \colon S_1, \ldots, \mathtt{l}_n \colon S_n \,\rangle \\ \bigoplus_{i\in[1;n]} !\mathtt{l}_i.\mathcal{M}(S_i) & \text{if } S = \oplus\langle\, \mathtt{l}_1 \colon S_1, \ldots, \mathtt{l}_n \colon S_n \,\rangle \\ \mu x.\, \mathcal{M}(S') & \text{if } S = \mu X.\, S' \\ x & \text{if } S = X \end{cases}$$

It is easy to see that $\mathcal{M}$ maps session types, $\mathsf{ST}_{\mathrm{fo}}$, to session contracts, $\mathsf{SC}_{\mathrm{fo}}$; indeed it defines a bijection between these sets:

- for every $\sigma \in \mathsf{SC}_{\mathrm{fo}}$ there exists some session type T such that $\mathcal{M}(T) = \sigma$
- if $\mathcal{M}(T_1) = \mathcal{M}(T_2)$ then $T_1 = T_2$

where $T_1 = T_2$ denotes syntactic identity. Further, substitution is preserved by $\mathcal{M}$.

The next two lemmas will be crucial in Section 6.3.

Lemma 2.3.11. Let $S, T \in \mathsf{ST}_{\mathrm{fo}}$. Then $\mathcal{M}(S\left\{{}^{T}/_{X}\right\}) = (\mathcal{M}(S))\left\{{}^{\mathcal{M}(T)}/_{\mathcal{M}(X)}\right\}$.

Proof. The proof is by structural induction on S. □

The interpretation also commutes with the two functions $\mathit{depth}(-)$ and $\text{UNF}(-)$:

Lemma 2.3.12. For every $T \in \mathsf{ST}_{\mathrm{fo}}$ and $\sigma \in \mathsf{SC}_{\mathrm{fo}}$ the ensuing properties are true,

(i) $\mathit{depth}(T) = \mathit{depth}(\mathcal{M}(T))$

(ii) $\text{UNF}(\mathcal{M}(T)) = \mathcal{M}(\text{UNF}(T))$

(iii) $\text{UNF}(\mathcal{M}^{-1}(\sigma)) = T$ if and only if $\text{UNF}(\sigma) = \mathcal{M}(T)$

Proof. The proofs of the first two points are by induction on $\mathit{depth}(T)$; we prove point (ii) using point (i) and Lemma 2.3.11. The third point follows immediately from point (ii). □

The interpretation $\mathcal{M}$ allows us to assign indirectly an LTS to session types; this deserves further explanation. One may wonder why we have not assigned directly an LTS to session types, defining it from scratches. The reason is that we do not wish to reason on a completely arbitrary LTS; we want to carry out our reasoning on (minor variations of) the standard LTS of CCS without τs, so as to embed session types into its well known theory, rather than developing a new behavioural theory from scratches.

2.4 Related Work

All the material we presented in this chapter is fairly standard and an account of it is given in Chapter 10.

Chapter 3

Client and peer satisfaction

One major concern is that the software systems we use be, in some sense, correct. In particular, we would like that clients we use be satisfied by servers; and that the peers be all satisfied.

A priori, there are many ways to define satisfaction, and it is not clear which one to pick. However, since the feature of software we are concerned with is communication, the notions of satisfaction that we will employ involve the interactions that take place in software systems.

The action $\checkmark$ in the LTS of processes turns out to be a key ingredient of our formalism. Intuitively, we think that if some process p can perform $\checkmark$, $p \xrightarrow{\checkmark}$, then p is satisfied.

In this chapter we introduce two relations for satisfaction, which now we discuss informally.

To test software is common practice. In our framework, we can test a process p by running it in parallel with a test r, and letting them interact as they wish. The test is passed if *eventually* it reaches a state that reports success, that is some $r' \xrightarrow{\checkmark}$, *regardless* of the particular communication pattern that take place between p and r. These intuitions are behind the well-known *testing theory* [De Nicola and Hennessy, 1984].

The testing theory, falls smoothly in a more general client/server setting. The tests are clients, and a client r is satisfied by a server p if p must pass r. This mirrors the intuition behind client/server systems, for the satisfaction is biased towards the client.

Software testing is a routine, but it is not the reason why we develop and use software. We neither use a web browser to test web servers, nor share files with our colleagues to ensure that they can access them. In daily practice, if we let a client r interact with a server p, our main concern is not the ability of r to always report success; rather that the requests of r be answered by p. The idea of satisfaction used in testing theory is not sensible in the framework we just described, and an an alternative definition of satisfaction is in order.

In the new client/server setting, it is sensible to deem r satisfied by p if

a) whenever r may require an interaction to go on computing, the server p will pay attention to r;

b) if r requires an interaction to go on, and p cannot answer to any request of r, then r has to be satisfied already

This notion of satisfaction is much more involved than the idea of passing a test, but, as we have discussed, it mirrors better the daily practice of using software. For instance, if our web browser sends to a web-server the request of a page, and the server does not reply, then neither the web-client nor we are satisfied. If we have a copy of that page, though, we already satisfied, and it does not matter that the server did not answer to the request of the client.

The two criteria for satisfaction that we described are formally described by the following relations,

$$\text{MUST}, \qquad \dashv \tag{3.1}$$

One is the well-known MUST testing relation [De Nicola and Hennessy, 1984], and the other is a compliance relation [Castagna et al., 2009; Laneve and Padovani, 2007; Padovani, 2010]. They are relations between processes, but their definitions apply equally well to session contracts, so we will abuse the notation and write statements such as ρ MUST σ, or $\rho \dashv \sigma$.

Structure of the chapter. In Section 3.1 we formally define the MUST testing, and its symmetric version for peers. We also prove the properties of these relations that we will need in the oncoming chapters. Similarly, in Section 3.2 we introduce formally the compliance relation, its symmetric version, and we prove their properties. We conclude Section 3.2 by exposing the differences between the relations MUST and $\dashv$. In the last section of the chapter, Section 3.3, we prove that in the sub-LTS of session contracts, the must testing and the compliance relation can be characterised in a syntax-oriented way.

3.1 Must testing

To formally define the MUST testing, we need some ancillary terminology. Recall the notion of computation of a composition $r \parallel p$. We say that a state $r \parallel p$ is *client-successful* if $r \xrightarrow{\checkmark}$, and if a computation contains a client-successful state, then we say that the computation is *client-successful*. If a computation is client-successful, and it contains also a state $r \parallel p$ in which $p \xrightarrow{\checkmark}$, then we say that computation is *successful*.

A computation of $r \parallel p$, say

$$r \parallel p = r_0 \parallel p_0 \xrightarrow{\tau} r_1 \parallel p_1 \xrightarrow{\tau} r_2 \parallel p_2 \xrightarrow{\tau} r_3 \parallel p_3 \xrightarrow{\tau}\xrightarrow{\tau} \dots$$

is *maximal* if one of the following conditions is true,

- the computation is infinite
- the computation is finite, and it cannot be extended

The last conditions above means that there exists some $n \in \mathbb{N}$ such that $r_n \parallel p_n \overset{\tau}{\not\longrightarrow}$.

We are ready to define the MUST testing.

Definition 3.1.1. [MUST testing]
For all processes r, p we write p MUST r if and only if all the maximal computations of $r \parallel p$ are client-successful. We refer to the relation denoted by MUST as the MUST *testing*. □

Note that in the above definition r can be thought of as a client, while p can be thought of as a server.

Example 3.1.2. For every server p, all the maximal computations of

$$1 + \alpha.\,0 \parallel p$$

are client-successful because in the first state reports success $1 + \alpha.\,0 \xrightarrow{\checkmark}$, so p MUST $1 + \alpha.\,0$.

Consider the processes $A \overset{\text{def}}{=} \alpha.A$ and $B \overset{\text{def}}{=} \overline{\alpha}.B$. No computation of $A \parallel B$ is client-successful, because A cannot perform $\checkmark$; it follows that $B \not\text{MUST}\ A$.

The same argument lets us prove that for every p, $p \not\text{MUST}\ 0$ and $p \not\text{MUST}\ \tau^\infty$. □

The definition of MUST requires maximal computations to be client-successful; that is only clients are required to reach successful states; this asymmetry is what lets us think of tests as client.

To reason on the satisfaction of peers, we focus on the symmetric version of the MUST testing.

Definition 3.1.3. [Peer MUST testing]
We write p MUST$^{\text{P2P}}$ r if and only if all the maximal computations of $p \parallel r$ are *successful*. □

If $\bowtie$ is not symmetric, then MUST^{P2P} is not a symmetric relation.

Example 3.1.4. If $\bowtie$ is not symmetric, then the relation $\dashv_{\text{P2P}}$ is not symmetric. Suppose that $\alpha \bowtie \beta$, and that $\beta \not\bowtie \alpha$. Plainly, $\beta.1 \ \text{MUST}^{\text{P2P}} \ \alpha.1$, because all the maximal computation of $\alpha.1 \mid\mid \beta.1$ are successful. However, $\alpha.1 \not\text{MUST}^{\text{P2P}} \ \beta.1$, because $\beta.1 \mid\mid \alpha.1 \stackrel{\tau}{\not\longrightarrow}$, and none of the processes is successful. □

The following statement is true.

Lemma 3.1.5. If $\bowtie$ is symmetric then the relation MUST^{P2P} is symmetric.

In the remaining part of the section, we discuss only the MUST testing and prove some of its properties; analogous properties of MUST^{P2P} follow in a straightforward way.

We will need three properties of the MUST testing relation.

Lemma 3.1.6. For every $r, p \in \mathsf{CCS}_{\text{w}\tau}$, if $p \Downarrow$ and $p \ \text{MUST} \ r$, then $r \stackrel{\checkmark}{\longrightarrow}$.

Proof. Let us pick a p such that $p \Downarrow$; for instance τ^∞. Since $p \Downarrow$, the process p engages in an infinite sequence of τ's, and there exists the maximal computation

$$r \mid\mid p_0 \stackrel{\tau}{\longrightarrow} r \mid\mid p_1 \stackrel{\tau}{\longrightarrow} r \mid\mid p_2 \stackrel{\tau}{\longrightarrow} r \mid\mid p_3 \stackrel{\tau}{\longrightarrow} \ldots$$

Since $p \ \text{MUST} \ r$, the computation above is client-successful; as the only state in the client side of the computation is r, it follows that $r \stackrel{\checkmark}{\longrightarrow}$. □

Lemma 3.1.7. Let $p, r \in \mathsf{CCS}_{\text{w}\tau}$. If every maximal computation of $r \mid\mid p$ contains a state $r_i \mid\mid p_i$ such that $p_i \ \text{MUST} \ r_i$, then $p \ \text{MUST} \ r$.

Proof. Fix a maximal computation C of $r \mid\mid p$; by hypothesis C contains a state $r_i \mid\mid p_i$ such that $p_i \ \text{MUST} \ r_i$. The suffix of C after $r_i \mid\mid p_i$ is a maximal computation of $r_i \mid\mid p_i$, thus Definition 3.1.1 ensures that in the suffix there exists a client-successful state $r' \mid\mid p'$. This implies that C is client-successful. As the only assumption on C is its being maximal, we apply the argument to every maximal computation of $r \mid\mid p$, thereby proving that every maximal computation of $r \mid\mid p$ is client-successful. This means that $p \ \text{MUST} \ r$. □

Corollary 3.1.8. *For every $r, p \in \mathsf{CCS}_{w\tau}$, $p \ \text{MUST} \ r$ if and only if every maximal computation of $r \mid\mid p$ contains a state $r' \mid\mid p'$ such that $p' \ \text{MUST} \ r'$.*

Proof. If $p \ \text{MUST} \ r$ then every computation contains a state $r' \mid\mid p'$ such that $p' \ \text{MUST} \ r'$, namely $r \mid\mid p$ itself. If every maximal computation of $r \mid\mid p$ contains a state $r' \mid\mid p'$ such that $p' \ \text{MUST} \ r'$, then Lemma 3.1.7 ensures that $p \ \text{MUST} \ r$. □

Every maximal computation of two session contracts $\rho \mid\mid \sigma$ contains the unfolding of these terms. This lets us prove that the relation MUST is closed with respect to unfolding.

Lemma 3.1.9. Let σ and ρ be session contracts. $\sigma \ \text{MUST} \ \rho$ if and only if $\text{UNF}(\sigma) \ \text{MUST} \ \text{UNF}(\rho)$.

Proof. We have to prove two implications, namely

a) if $\sigma \ \text{MUST} \ \rho$ then $\text{UNF}(\sigma) \ \text{MUST} \ \text{UNF}(\rho)$

b) if $\text{UNF}(\sigma) \ \text{MUST} \ \text{UNF}(\rho)$ then $\sigma \ \text{MUST} \ \rho$

We prove the first implication. Suppose that $\sigma \ \text{MUST} \ \rho$ and fix a maximal computation of $\text{UNF}(\rho) \mid\mid \text{UNF}(\sigma)$; we have to prove that the computing at hand contains a state in which the derivative of $\text{UNF}(\rho)$ is successful.

Append the maximal computation of $\text{UNF}(\rho) \mid\mid \text{UNF}(\sigma)$ to the computation

$$\rho \mid\mid \sigma \Longrightarrow \text{UNF}(\rho) \mid\mid \text{UNF}(\sigma)$$

We know that this computation exists in view of the rules [A-UNFOLD], [P-LEFT], and [P-RIGHT]. We can prove that in the computation above, which is due to the unfoldings of the contracts, no state can be successful. Since by assumption σ MUST ρ a state in the computation of $\text{UNF}(\sigma) \mid\mid \text{UNF}(\rho)$ must be successful.

Now we prove the second implication. Suppose that $\text{UNF}(\sigma)$ MUST $\text{UNF}(\rho)$. Fix a maximal computation of $\rho \mid\mid \sigma$; since the computation is maximal it has length at least $depth(\sigma) + depth(\rho)$, for otherwise it can be extended by unfolding one of the contracts. Lemma 2.3.8 applied twice implies that implies the maximal computation contains the state $\text{UNF}(\rho) \mid\mid \text{UNF}(\sigma)$.

It follows that every maximal computation contains the state $\text{UNF}(\rho) \mid\mid \text{UNF}(\sigma)$; the assumption that $\text{UNF}(\sigma)$ MUST $\text{UNF}(\rho)$, and Lemma 3.1.7 ensure that σ MUST ρ. □

3.2 Compliance relation

In this section we introduce the compliance relation. Our definition is a variation on that proposed in [Laneve and Padovani, 2007].

Definition 3.2.1. [Compliance relation]
Let $\mathcal{F}^{\dashv} : \mathcal{P}(\mathsf{CCS}^2_{w\tau}) \longrightarrow \mathcal{P}(\mathsf{CCS}^2_{w\tau})$ be the rule functional defined so that $(r, p) \in \mathcal{F}^{\dashv}(\mathcal{R})$ whenever the following conditions hold:

(a) if $r \Downarrow$ then $p \Downarrow$

(b) if $r \mid\mid p \stackrel{\tau}{\not\longrightarrow}$ then $r \xrightarrow{\checkmark}$

(c) if $r \mid\mid p \xrightarrow{\tau} r' \mid\mid p'$ then $r' \mathcal{R} p'$

If $X \subseteq \mathcal{F}^{\dashv}(X)$, then we say that X is a *co-inductive* compliance *relation*. Lemma C.0.18 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\dashv}(X)$; we call this solution the *compliance relation*, and we denote it $\dashv$. That is $\dashv = \nu X.\mathcal{F}^{\dashv}(X)$. If $r \dashv p$ we say that the process r *complies with* the process p. □

There is an asymmetry in the relation $r \dashv p$. The intention is that any client process r when interacting with a server process p will be satisfied, if

- if the client may need to interact with the server to go on computing , $r \Downarrow$, then the server will try to interact with r, $p \Downarrow$
- if the interaction gets stuck, the client is in a state in which it is satisfied, $r \xrightarrow{\checkmark}$
- the interaction between client and server will go on indefinitely

The first property above, which is due to condition (a), is conservative; in the following sense. Consider the client $1 \oplus 1$ and the server τ^∞. More in general, it is not clear whether $1 \oplus 1$ is satisfied or not by the server τ^∞. Consider the infinite computation due to the divergence of τ^∞,

$$1 \oplus 1 \mid\mid \tau^\infty \xrightarrow{\tau} 1 \oplus 1 \mid\mid \tau^\infty \xrightarrow{\tau} 1 \oplus 1 \mid\mid \tau^\infty \xrightarrow{\tau} \dots$$

The client $1 \oplus 1$ can reach a successful state on its own, but in the computation above this never happens; nor the computation can be extended to make it happen. Condition (a) establishes if a client r by interacting with a server p ends up in an unclear situation as the one sketched, then r is

not satisfied. For instance, $1 \oplus 1 \not\dashv \tau^\infty$, because $1 \oplus 1 \Downarrow$, whereas $\tau^\infty \not\Downarrow$. Since $1 \Downarrow$ and $\tau^\infty \not\Downarrow$, $1 \not\dashv \tau^\infty$. This shows how conservative is condition (a) of Definition 3.2.1. According to $\dashv$, a stable client r is not satisfied by a server p unless p will surely let the client perform some action.

Example 3.2.2. [Compliance and divergent terms]
In this example we show some instances of how divergent terms are related by the compliance relation. For every process p, $\tau^\infty \dashv p$; to see why this is true, note that the following relation is a co-inductive compliance,

$$\mathcal{R} = \{ (\tau^\infty, p') \mid p \Longrightarrow p' \}$$

Now we prove that $\tau^\infty + \alpha.0 \not\dashv \tau^\infty + \overline{\alpha}.0$; the reason for this is the following computation

$$\tau^\infty + \alpha.0 \,||\, \tau^\infty + \overline{\alpha}.0 \xrightarrow{\tau} 0 \,||\, 0 \overset{\tau}{\not\longrightarrow}$$

The state $0 \,||\, 0$ is stable and $0 \overset{\checkmark}{\not\longrightarrow}$, so $0 \not\dashv 0$. Point (c) of Definition 3.2.1 implies that

$$\tau^\infty + \alpha.0 \not\dashv \tau^\infty + \overline{\alpha}.0$$

□

Example 3.2.3. [Clients and ✓ action]
According to our definition of compliance, the client need not ever perform ✓. We have shown an instance of this in Example 3.2.2: $\tau^\infty \dashv \tau^\infty$. This phenomenon, though, does not depend on the divergence of the client. For example, Let $A \stackrel{\text{def}}{=} \alpha.A$ and $B \stackrel{\text{def}}{=} \overline{\alpha}.B$, and consider the set $\mathcal{R} = \{ (A, B) \}$. The relation $\mathcal{R}$ is a co-inductive compliance relation, and the client process, A, does not perform ✓ at all. □

Intuitively, 0 represents a non satisfied client; formally this is the case because $0 \not\dashv p$ for every process p. The process 1 is satisfied by all the servers that converges, and the process τ is always satisfied, for it does not need any server to carry on its computation.

Lemma 3.2.4. For every $p \in \mathsf{CCS}_{\mathrm{w}\tau}$, the following statements are true,

i) $0 \not\dashv p$

ii) if $p \Downarrow$ then $1 \dashv p$

iii) $\tau^\infty \dashv p$

Proof. The first fact is true if $p \not\Downarrow$. If $p \Downarrow$ the p reduces to some stable p', so $0 \,||\, p \Longrightarrow 0 \,||\, p' \overset{\tau}{\not\longrightarrow}$, and $0 \overset{\checkmark}{\not\longrightarrow}$. The second fact is true because 1 cannot interact with p, and all the states reached by $1 \,||\, p$ are client-successful, so the following relation is a co-inductive compliance, $\mathcal{R} = \{ (1, p') \mid p \Longrightarrow p' \}$. The third fact is proven in Example 3.2.2. □

Example 3.2.5. Referring to Example 2.3.5, it is routine work to check that the relation in Figure 3.1 is a co-inductive compliance. □

The following properties of the compliance relation will be useful later in the thesis.

Lemma 3.2.6. Let $\mathcal{R}$ be a co-inductive compliance such that $r \,\mathcal{R}\, p$. If $r \,||\, p \xrightarrow{\tau}{}^{n} r' \,||\, p'$ for some $n \in \mathbb{N}$, then $r' \,\mathcal{R}\, p'$

Proof. The proof is by induction on n.

Base case. In this case $n = 0$, so $r' \,||\, p' = r \,||\, p$. The hypothesis $r \,\mathcal{R}\, p$ ensures that $r' \,\mathcal{R}\, p'$.

$$
\begin{aligned}
\mathcal{R} = \{&(\text{Voter}, \text{Ballot}),\\
&(?\texttt{Wrong}.\text{Voter} + ?\texttt{Ok}.(\,!\texttt{VoteA}.\,1 \oplus !\texttt{VoteB}.\,1\,)),\\
&!\texttt{Wrong}.\text{Ballot} \oplus !\texttt{Ok}.(?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot})),\\
&(?\texttt{Ok}.(\,!\texttt{VoteA}.\,1 \oplus !\texttt{VoteB}.\,1\,), !\texttt{Ok}.(?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(!\texttt{VoteA}.\,1 \oplus !\texttt{VoteB}.\,1, ?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(!\texttt{VoteA}.\,1, ?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(!\texttt{VoteB}.\,1, ?\texttt{VoteA}.\text{Ballot} + ?\texttt{VoteB}.\text{Ballot}),\\
&(1, \text{Ballot})\}
\end{aligned}
$$

Figure 3.1: A co-inductive compliance $\mathcal{R}$; see Example 3.2.5

Inductive case. In this case $n = m+1$ for some $m \in \mathbb{N}$, and the reduction sequence $r \parallel p \xrightarrow{\tau}{}^{n} r' \parallel p'$ can be split as follows,

$$r \parallel p \xrightarrow{\tau} r'' \parallel p'' \xrightarrow{\tau}{}^{m} r' \parallel p'$$

The hypothesis that $\mathcal{R}$ is co-inductive compliance, and point (c) of Definition 3.2.1 imply that $r'' \;\mathcal{R}\; p''$. Since $r'' \parallel p'' \xrightarrow{\tau}{}^{m} r' \parallel p'$ and m is smaller than n, we are allowed to use the inductive hypothesis which implies that $r' \;\mathcal{R}\; p'$. □

Corollary 3.2.7. *If $r \dashv p$ and $r \parallel p \Longrightarrow r' \parallel p'$, then $r' \dashv p'$.*

Lemma 3.2.8. Let r_1, r_2, p_1, and p_2 be processes. The following statements hold,

(i) if $r \dashv p_1$, $r \dashv p_2$ then $r \dashv p_1 \oplus p_2$

(ii) if $r_1 \dashv p$, $r_2 \dashv p$ then $r_1 \oplus r_2 \dashv p$

Proof. As an example we outline the proof of (i). Let $\mathcal{R}$ be the relation defined by

$$\mathcal{R} = \{\, (r, p) \mid r \dashv p \text{ or } p = p_1 \oplus p_2 \text{ where } r \dashv p_1 \text{ and } r \dashv p_2 \,\}$$

It is straightforward to show that $\mathcal{R}$ is a compliance relation, from which the result follows. □

The next two propositions show that we can reason up-to unfolding on the compliance relation.

Corollary 3.2.9. *For every co-inductive compliance $\mathcal{R}$, and $\rho, \sigma \in \mathsf{SC}_{fo}$, if $\rho \;\mathcal{R}\; \sigma$ then $\rho \;\mathcal{R}\; \text{UNF}(\sigma)$ and $\text{UNF}(\rho) \;\mathcal{R}\; \sigma$.*

Proof. Both facts follow in a straightforward manner from point (ii) of Lemma 2.3.7 and point (c) of Definition 3.2.1. □

The converse is also true:

Proposition 3.2.10. For all session contracts ρ, σ, we have the following

(a) if $\rho \dashv \text{UNF}(\sigma)$ then $\rho \dashv \sigma$

(b) if $\text{UNF}(\rho) \dashv \sigma$ then $\rho \dashv \sigma$

Proof. Let us look at the proof of (a). Let $\mathcal{R}= \{\, (\rho,\, \sigma) \mid \rho \dashv \sigma \text{ or } \rho \dashv \text{UNF}(\sigma) \,\}$. The result will follow if we can prove that $\mathcal{R}$ is a co-inductive compliance relation, as given in Definition 3.2.1.

(i) Suppose $\rho \parallel \sigma \overset{\tau}{\not\longrightarrow}$. If $\rho \dashv \sigma$ then by definition $r \xrightarrow{\checkmark}$. Otherwise

$$\rho \dashv \text{UNF}(\sigma)$$

Note that $\sigma \stackrel{\tau}{\not\longrightarrow}$ and therefore by Lemma 2.3.7 it follows that $\text{UNF}(\sigma) = \sigma$, which means, since now $\rho \dashv \sigma$, $r \stackrel{\checkmark}{\longrightarrow}$.

(ii) Suppose $\rho \,||\, \sigma \stackrel{\tau}{\longrightarrow} \rho' \,||\, \sigma'$. We have to show $\rho' \,\mathcal{R}\, \sigma'$, which is obvious if $\rho \dashv \sigma$. On the other hand if $\rho \dashv \text{UNF}(\sigma)$ there are three cases, depending on the inference of the action $\rho \,||\, \sigma \stackrel{\tau}{\longrightarrow} \rho' \,||\, \sigma'$. If the action is due to a silent move of ρ, the result follows from point (c) of Definition 3.2.1. In the other cases the result will follow by an application of Lemma 2.3.7 of point (c) of Definition 3.2.1.

□

Definition 3.2.11. [Peer compliance relation]
Let $\mathcal{F}_{\dashv_{\text{P2P}}} : \mathcal{P}(\mathsf{CCS}^2_{\text{w}\tau}) \longrightarrow \mathcal{P}(\mathsf{CCS}^2_{\text{w}\tau})$ be the rule functional defined so that $(r, p) \in \mathcal{F}_{\dashv_{\text{P2P}}}(\mathcal{R})$ whenever both the following hold:

(i) $r \Downarrow$ if and only $p \Downarrow$

(ii) if $r \,||\, p \stackrel{\tau}{\not\longrightarrow}$ then $r \stackrel{\checkmark}{\longrightarrow}$, $p \stackrel{\checkmark}{\longrightarrow}$

(iii) if $r \,||\, p \stackrel{\tau}{\longrightarrow} r' \,||\, p'$ then $r' \,\mathcal{R}\, p'$

If $X \subseteq \mathcal{F}^{\dashv_{\text{P2P}}}(X)$, then we say that X is a *co-inductive* peer compliance *relation*. Lemma C.0.19 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\dashv_{\text{P2P}}}(X)$; we call this solution the peer compliance *relation*, and we denote it $\dashv_{\text{P2P}}$. That is $\dashv_{\text{P2P}} = \nu X.\mathcal{F}^{\dashv_{\text{P2P}}}(X)$.

If $r \dashv_{\text{P2P}} p$ we say that the contract r *mutually complies with* the contract p. □

Example 3.1.4 can be used to prove that if $\bowtie$ is not symmetric then $\dashv_{\text{P2P}}$ is not symmetric either. The following result is true.

Lemma 3.2.12. If $\bowtie$ is symmetric, then the relation $\dashv_{\text{P2P}}$ is the greatest symmetric co-inductive compliance relation.

3.2.1 Comparing satisfactions

A comparison between the MUST testing and the compliance relation is in order, at least to make sure that we have indeed formalised different criteria for satisfactions.

First we remark a nuisance. When dealing with MUST, the servers are on the right-hand and the clients appear on the left-hand side; so p MUST r means that p is a server and r a client. When dealing with the compliance, the situation is the converse, if $r \dashv p$, then p is the server, while r is the client. So to compare the compliance and the testing one has to actually use the inverse of one of the relations.

There are two differences between the criteria formalised respectively by MUST and by $\dashv$. The first difference is how ever lasting computations are treated.

Example 3.2.13. [Ever lasting computations]
In this example we prove that $\dashv \not\subseteq \text{MUST}^{-1}$. Recall the process A and B from Example 3.1.2 and Example 3.2.3. In the first example we have proven that $B \not\text{MUST}\, A$, whereas in the second example we have shown that $A \dashv B$. □

The example above shows that the compliance relation does not require the testing process to ever report success, provided that the communication between the processes can continue *indefinitely*. On the contrary, for p MUST r requires that r reports success.

The second difference between MUST and $\dashv$ is what happens after a client has reached a successful state. The subsequent computation is disregarded by MUST, whereas the compliance relation has to hold for *all* the states in *all* computations.

Example 3.2.14. [Continuations after ✓]
In this example, following the intuition given above, we prove that $\text{MUST}^{-1} \not\subseteq \dashv$. The two following facts are true.

1) $\alpha.\,0 \text{ MUST } 1 + \overline{\alpha}.\,0$

2) $1 + \overline{\alpha}.\,0 \not\dashv \alpha.\,0$

Where 1) is true because $1 + \overline{\alpha}.\,0 \xrightarrow{\checkmark}$, and 2) is true because $1 + \overline{\alpha}.\,0 \mid\mid \alpha.\,0 \xrightarrow{\tau} 0 \mid\mid 0 \not\xrightarrow{\tau}$ and $0 \not\xrightarrow{\checkmark}$.

□

We have proven that the relations MUST^{-1} and $\dashv$, in general are not comparable,

$$\text{MUST}^{-1} \not\subseteq \dashv, \qquad \dashv \not\subseteq \text{MUST}^{-1}$$

If we restrict the LTS that we use, though, a relation between $\dashv$ and MUST^{-1} may arise. For instance, let $\mathsf{CCS}^{\mathsf{fin}}_{\mathrm{w}\tau}$ be the language of processes that perform only finite action sequences.

Proposition 3.2.15. Let $\mathcal{R}$ be a co-inductive compliance relation on terms of $\mathsf{CCS}^{\mathsf{fin}}_{\mathrm{w}\tau}$. If $r\ \mathcal{R}\ p$ then $p \text{ MUST } r$.

Proof. We have to prove that, under the hypothesis that $r\ \mathcal{R}\ p$, all the maximal computations of $r \mid\mid p$ are client-successful. Fix a maximal computation C of $r \mid\mid p$,

$$r \mid\mid p = r_0 \mid\mid p_0 \xrightarrow{\tau} r_1 \mid\mid p_1 \xrightarrow{\tau} r_2 \mid\mid p_2 \xrightarrow{\tau} \dots$$

Since $r, p \in \mathsf{CCS}^{\mathsf{fin}}_{\mathrm{w}\tau}$ the computation must be finite, that is there exists a $r_k \mid\mid p_k$ for some $k \in \mathbb{N}$ such that $r_k \mid\mid p_k \not\xrightarrow{\tau}$. Since $r_k \dashv p_k$, condition (b) of Definition 3.2.1 implies that $r_k \xrightarrow{\checkmark}$. Since there is no particular assumption on C, the argument above can be applied to all the maximal computations of $r \mid\mid p$, thus $p \text{ MUST } r$. □

3.3 Syntactic characterisations

Thus far we have presented the must testing and the compliance relation in the general within setting of processes. We have mentioned session contracts only to show that we can reason on the relations MUST and $\dashv$ up-to unfolding. In this section we focus completely on session contracts, and we prove that in this setting the compliance relation and the must testing can be characterised looking only at the syntax of the terms. Essentially, this is possible because the syntax of session contracts is restrictive enough to capture the behavioural properties of session contracts.

3.3.1 Syntactic compliance

The restrictive syntax of session contracts let us give a syntactic characterisation of the compliance relation and of the must testing relation. In this section we focus on the former characterisation, which we prove in Lemma 3.3.10. This characterisation relies on a co-inductive relation (Definition 3.3.1), that we define now.

Definition 3.3.1. [Syntactic compliance relation]
Let $\mathcal{F}^{\dashv^s}_{\text{MUST}^{\text{s}}} : \mathcal{P}(\mathsf{SC}^2_{\text{fo}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{fo}})$ be the rule functional given by the inference rules in Figure 3.2. If $X \subseteq \mathcal{F}^{\dashv^s}_{\text{MUST}^{\text{s}}}(X)$, then we say that X is a *co-inductive syntactic compliance*. Lemma C.0.20 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\dashv^s}_{\text{MUST}^{\text{s}}}(X)$; we call this solution the *syntactic compliance*, and we denote it $\dashv^s$. That is $\dashv^s = \nu X.\mathcal{F}^{\dashv^s}_{\text{MUST}^{\text{s}}}(X)$. □

$$\frac{}{1 \dashv^s \sigma} \text{ [A-Unit]}$$

$$\frac{\rho' \dashv^s \sigma'}{\alpha.\rho' \dashv^s \overline{\alpha}.\sigma'} \ \alpha \text{ contains no labels [R-Alpha]}$$

$$\frac{\rho_1 \dashv^s \sigma_1 \quad \dots \quad \rho_{\hat{j}} \dashv^s \sigma_{\hat{j}}}{\sum_{i\in I} ?1_i.\rho_i \dashv^s \bigoplus_{j\in J} !1_j.\sigma_j} \ J \subseteq I, \hat{j} = |J|; \text{[R-Exch]}$$

$$\frac{\rho_1 \dashv^s \sigma_1 \quad \dots \quad \rho_{\hat{i}} \dashv^s \sigma_{\hat{i}}}{\bigoplus_{i\in I} !1_i.\rho_i \dashv^s \sum_{j\in J} ?1_j.\sigma_j} \ I \subseteq J, \hat{i} = |I|; \text{[R-Inch]}$$

$$\frac{\text{UNF}(\rho) \dashv^s \text{UNF}(\sigma)}{\rho \dashv^s \sigma} \ \mathit{depth}(\rho) + \mathit{depth}(\sigma) > 0; \text{ [R-Unfold]}$$

Figure 3.2: Inference rules for the rule functional $\mathcal{F}^{\dashv^s}_{\text{MUST}^s}$

We briefly comment the definition above. The rule functional $\mathcal{F}^{\dashv^s}_{\text{MUST}^s}$ gives rise not only to a compliance relation (by taking its greatest fixed point), but also to a MUST testing relation (by taking its least fixed point). This justifies the use of both a subscript and a superscript.

A side condition is in [R-Alpha], that requires α not to contain a label. This condition guarantees that the rule functional $\mathcal{F}^{\dashv^s}_{\text{MUST}^s}$ is invertible (see chapter 21 of Pierce [2002]). Intuitively, if there was no such requirement, the rules defining the rule functional $\mathcal{F}^{\dashv^s}_{\text{MUST}^s}$ would be ambiguous.

Example 3.3.2. In this example we show how the side condition

$$\alpha \text{ contains no label [R-Alpha]}$$

resolves a possible ambiguity in the application of the rules defining $\mathcal{F}^{\dashv^s}_{\text{MUST}^s}$.

Let $\mathcal{R}$= $\{(?1.\,1,\, !1.\,1), (1,\, 1)\}$. If rule [R-Alpha] did not have the side conditions above, then we would have two ways to prove that $\mathcal{R}$ is a co-inductive syntactic compliance:

$$\frac{\dfrac{}{(1,\, 1)}\text{ [A-Unit]}}{(?1.\,1,\, !1.\,1)}\text{ [R-Alpha]} \qquad \frac{\dfrac{}{(1,\, 1)}\text{ [A-Unit]}}{(?1.\,1,\, !1.\,1)}\text{ [R-Exch]}$$

(a) (b)

The requirement on α in rule [R-Alpha] ensures that the *only* way to prove that $\mathcal{R}$ is a co-inductive syntactic compliance is by using the inference tree (b). □

The co-inductive syntactic compliances are closed with respect to unfolding.

Lemma 3.3.3. For every co-inductive syntactic compliance $\mathcal{R}$, and $\rho, \sigma \in \mathsf{SC}_{\text{fo}}$, if $\rho\ \mathcal{R}\ \sigma$ then $\text{UNF}(\rho)\ \mathcal{R}\ \text{UNF}(\sigma)$.

Proof. The argument is similar to the proof of Lemma 2.1.16. □

Lemma 3.3.4. Let $\mathcal{R}$ be a co-inductive compliance on session contracts. The relation $\mathcal{R}$ is a co-inductive syntactic compliance.

Proof. We have to prove that if $\mathcal{R} \subseteq \mathcal{F}_{\dashv}(\mathcal{R})$, then $\mathcal{R} \subseteq \mathcal{F}^{\dashv^s}_{\text{MUST}^s}(\mathcal{R})$.

Fix a pair $\rho\ \mathcal{R}\ \sigma$; we are required to show that $(\rho, \sigma) \in \mathcal{F}^{\dashv^s}_{\text{MUST}^s}(\mathcal{R})$; this amounts in proving that we can derive the pair (ρ, σ) by instantiating one of the rules in Figure 3.2, and using the pairs in $\mathcal{R}$ as premises.

We first look at the depth of ρ and σ. Suppose that $depth(\rho) + depth(\sigma) > 0$. Corollary 3.2.9 implies that $\text{UNF}(\rho)\ \mathcal{R}\ \text{UNF}(\sigma)$. The following derivation proves that $(\rho, \sigma) \in \mathcal{F}^{\dashv^s}_{\text{MUST}^s}(\mathcal{R})$,

$$\frac{\text{UNF}(\rho) \dashv^s \text{UNF}(\sigma)}{\rho \dashv^s \sigma}\ [\text{R-UNFOLD}]$$

Suppose now that $depth(\rho) + depth(\sigma) = 0$. The argument proceeds by case analysis on ρ. Since $depth(\rho) = 0$ the term ρ has no top-most μ.

i) If $\rho = \mathsf{1}$, then we have the derivation

$$\frac{}{\rho \dashv^s \sigma}\ [\text{A-UNIT}]$$

ii) Suppose that $\rho = \alpha.\rho'$ with α containing no labels. We show that $\rho \dashv^s \sigma$ by using rule [R-ALPHA]; we explain how.

Plainly $\rho \overset{\checkmark}{\not\longrightarrow}$, thus Definition 3.2.1 implies that $\rho \parallel \sigma \overset{\tau}{\longrightarrow}$. Since $\rho \overset{\tau}{\not\longrightarrow}$, either ρ can interact with σ, or $\sigma \overset{\tau}{\longrightarrow}$.

As $\sigma \Longrightarrow \hat{\sigma} \overset{\tau}{\not\longrightarrow}$ implies that ρ must interact with $\hat{\sigma}$, the restrictive syntax of session types implies that $\hat{\sigma} = \overline{\alpha}.\sigma'$. Thanks to the syntax of session contracts and to the assumption $depth(\sigma) = 0$, we can prove that $\sigma = \beta.\sigma'$.

There exists the computation $\rho \parallel \sigma \Longrightarrow \rho' \parallel \sigma'$, thus Corollary 3.2.7 ensures that $\rho'\ \mathcal{R}\ \sigma'$.

We can now instantiate [E-ALPHA] as follows,

$$\frac{\rho' \dashv^s \sigma'}{\rho \dashv^s \sigma}\ [\text{R-ALPHA}]$$

iii) If $\rho = \sum_{i\in I} ?\mathtt{l}_i.\rho_i$, then $\rho \overset{\checkmark}{\not\longrightarrow}$. We derive (ρ, σ) by using rule [R-EXCH]. Consider the set $\mathcal{S} = \{\, \sigma' \mid \sigma \Longrightarrow \sigma' \overset{\tau}{\not\longrightarrow} \}$. The syntax of session contracts ensures that the set $\mathcal{S}$ is non-empty, and the syntax ensures that it is finite; so let $\mathcal{S} = \{\sigma_1, \ldots, \sigma_n\}$. Corollary 3.2.7 ensures that that $\rho\ \mathcal{R}\ \sigma_j$, for every $j \in [1; n]$.

Fix a $j \in [1; n]$; as $\rho \overset{\checkmark}{\not\longrightarrow}$, Definition 3.2.1 implies that $\rho \parallel \sigma_j \overset{\tau}{\longrightarrow}$. Since both ρ and σ_j are stable, they must interact. Given the syntax of ρ, we know that $\rho \overset{\mathtt{l}_i}{\longrightarrow} \rho'$ for some $i \in I$, that $\sigma_j \overset{\overline{\mathtt{l}}_i}{\longrightarrow} \sigma'_j$ for some $j \in [1; n]$, and $\rho'\ \mathcal{R}\ \sigma'_j$. In view of the syntax of session contracts, it must be the case that

$$\sigma = !\mathtt{l}_1.\sigma'_{j_1} \oplus \ldots \oplus !\mathtt{l}_n.\sigma'_{j_n}$$

and $n \leq |I|$. Moreover for every $k \in [1; n]$ we have $\rho_k\ \mathcal{R}\ \sigma'_{j_k}$.

Now we apply rule [R-EXCH]

$$\frac{\rho_1 \dashv^s \sigma_1 \quad \ldots \quad \rho_{\hat{k}} \dashv^s \sigma_{\hat{k}}}{\rho \dashv^s \sigma}\ [\text{R-EXCH}]$$

iv) If $\rho = \bigoplus_{i\in I} !\mathtt{l}_i.\rho_i$, then the argument is similar to the one used in the previous case of this proof.

□

The converse of Lemma 3.3.4 is not true.

Example 3.3.5. We prove that a co-inductive syntactic compliance needs not be a co-inductive compliance. First let

$$\begin{aligned}\rho &= !\mathtt{l}_1.1 \oplus (!\mathtt{l}_2.1 \oplus !\mathtt{l}_3.1)\\ \sigma &= ?\mathtt{l}_1.1 + ?\mathtt{l}_2.1 + ?\mathtt{l}_3.1\end{aligned}$$

and then let

$$\mathcal{R} = \{(\rho,\,\sigma), (1,\,1), (!\mathtt{l}_1.1,\,\sigma), (!\mathtt{l}_2.1,\,\sigma), (!\mathtt{l}_3.1,\,\sigma)\}$$

We prove that $\mathcal{R}$ is a co-inductive syntactic compliance. Consider the following inference tree

$$\dfrac{\dfrac{\dfrac{}{(1,\,1)}\ [\text{A-Unit}]}{(!\mathtt{l}_1.1,\,\sigma)}\ [\text{R-Inch}] \quad \dfrac{\dfrac{}{(1,\,1)}\ [\text{A-Unit}]}{(!\mathtt{l}_2.1,\,\sigma)}\ [\text{R-Inch}] \quad \dfrac{\dfrac{}{(1,\,1)}\ [\text{A-Unit}]}{(!\mathtt{l}_3.1,\,\sigma)}\ [\text{R-Inch}]}{(\rho,\,\sigma)}\ [\text{R-Inch}]$$

The tree above proves that $\mathcal{R} \subseteq \mu X.\mathcal{F}^{\dashv^s}_{\text{MUST}^\text{s}}(X)$, thus $\mathcal{R} \subseteq \nu X.\mathcal{F}^{\dashv^s}_{\text{MUST}^\text{s}}(X) = \dashv^s$.

The relation $\mathcal{R}$ is not a co-inductive compliance. To prove this, we show a pair $\rho\ \mathcal{R}\ \sigma$ that does not satisfy point (c) of Definition 3.2.1, that is $\rho \parallel \sigma \xrightarrow{\tau} \rho' \parallel \sigma'$ and $\rho'\ \not\mathcal{R}\ \sigma'$.

We can infer the reduction $\rho \xrightarrow{\tau} !\mathtt{l}_2.1 \oplus !\mathtt{l}_3.1$, thus $\rho \parallel \sigma \xrightarrow{\tau} !\mathtt{l}_2.1 \oplus !\mathtt{l}_3.1 \parallel \sigma$. Checking the pairs in $\mathcal{R}$, we see that $!\mathtt{l}_2.1 \oplus !\mathtt{l}_3.1\ \not\mathcal{R}\ \sigma$. □

Notwithstanding Example 3.3.5, we would like to prove that the syntactic compliance coincides with the compliance relation (on session contracts). To prove this, we need two ancillary results; they show that the problem exhibited in Example 3.3.5 can be easily solved, and does not affect the relations $\dashv^s$, ie. the *greatest* co-inductive syntactic compliance.

Lemma 3.3.6. For every co-inductive syntactic compliance $\mathcal{R}$, and $\rho, \sigma \in \mathsf{SC}_{\text{fo}}$, if $\rho\ \mathcal{R}\ \sigma$, then the following implications are true

a) if $\rho \xrightarrow{\tau} \rho'$ then the relation $\{(\rho',\,\sigma)\} \cup \mathcal{R}$ is a co-inductive syntactic compliance

b) if $\sigma \xrightarrow{\tau} \sigma'$ then the relation $\{(\rho,\,\sigma')\} \cup \mathcal{R}$ is a co-inductive syntactic compliance

Proof. The proofs of a) and b) are similar, so we discuss only a).

Let $\mathcal{R}' = \{(\rho',\,\sigma)\} \cup \mathcal{R}$. We are required to prove the set inclusion $\mathcal{R}' \subseteq \mathcal{F}^{\dashv^s}_{\text{MUST}^\text{s}}(\mathcal{R}')$. To this end, we have to show that if $\hat\rho\ \mathcal{R}'\ \hat\sigma$ then we can use one of the rules in Figure 3.2 to infer

$$\dfrac{\cdots}{\hat\rho \dashv^s \hat\sigma}$$

using in the premises in elements of $\mathcal{R}'$ itself.

Fix a pair $\hat\rho\ \mathcal{R}'\ \hat\sigma$. Either $\hat\rho\ \mathcal{R}\ \hat\sigma$, or $\hat\rho = \rho'$ and $\hat\sigma = \sigma$. In the first case we know that the derivation we have to show exists; this is true because by hypothesis $\mathcal{R} \subseteq \mathcal{F}^{\dashv^s}_{\text{MUST}^\text{s}}(\mathcal{R})$.

Suppose that $\hat\rho = \rho'$ and $\hat\sigma = \sigma$. The argument is by case analysis and depends on why $\rho \xrightarrow{\tau} \rho'$; the restrictive syntax of session contracts guarantees that the silent move can have been inferred by applying one of the rules [A-Unf], [A-In-L], [A-In-R] (see Figure 2.10 and Figure 2.8).

If the silent move is due to the axiom [A-Unf], then $\mathit{depth}(\rho) > 0$ and $\text{UNF}(\rho) = \text{UNF}(\rho')$. The hypothesis $\rho\ \mathcal{R}\ \sigma$ and Lemma 3.3.3 imply that $\text{UNF}(\rho)\ \mathcal{R}\ \text{UNF}(\sigma)$, thus $\text{UNF}(\rho')\ \mathcal{R}\ \text{UNF}(\sigma)$. If $\mathit{depth}(\rho') + \mathit{depth}(\sigma) > 0$ then we can instantiate [R-Unfold],

$$\dfrac{\text{UNF}(\rho') \dashv^s \text{UNF}(\sigma)}{\rho' \dashv^s \sigma}\ \mathit{depth}(\rho') + \mathit{depth}(\sigma) > 0;\ [\text{R-Unfold}]$$

If $\mathit{depth}(\rho') + \mathit{depth}(\sigma) = 0$, then $\text{UNF}(\rho') = \rho'$ and $\text{UNF}(\sigma) = \sigma$. The fact that $\text{UNF}(\rho')\ \mathcal{R}\ \text{UNF}(\sigma)$ implies that $\rho'\ \mathcal{R}\ \sigma$, so the hypothesis $\mathcal{R} \subseteq \mathcal{F}^{\dashv^s}_{\text{MUST}^\text{s}}(\mathcal{R})$ ensures that $(\rho',\,\sigma) \in \mathcal{F}^{\dashv^s}_{\text{MUST}^\text{s}}(\mathcal{R})$.

If the silent move is due to the axiom [A-In-L], then the session contract ρ must be a choice of labels, $\rho = \rho' \oplus \rho''$; in view of the syntax of session contracts we have the following equality,

$$\rho = !\mathtt{l}_1.\rho_1 \oplus (\bigoplus_{i\in I} !\mathtt{l}_i.\rho_i)$$

The hypothesis that $\mathcal{R} \subseteq \mathcal{F}^{\dashv s}_{\text{MUST}^s}(\mathcal{R})$ implies that we have the derivation

$$\frac{\rho_1 \dashv^s \sigma_1 \ \ldots \ \rho_{|I|} \dashv^s \sigma_{|I|}}{\rho \dashv^s \sigma} \ [\text{R-Inch}]$$

It follows that $\sigma = \sum_{j\in J} ?\mathtt{l}_j.\sigma_j$, and $\{1\} \cup I \subseteq J$. We know that ρ' is either $!\mathtt{l}_1.\rho_1$ or $\bigoplus_{i\in I} !\mathtt{l}_i.\rho_i$; to see why $(\rho',\, \sigma) \in \mathcal{F}^{\dashv s}_{\text{MUST}^s}(\mathcal{R}')$ notice that can instantiate [R-Inch] as follows,

$$\frac{\rho_1 \dashv^s \sigma_1}{!\mathtt{l}_1.\rho_1 \dashv^s \sum_{j\in J} ?\mathtt{l}_j.\sigma_j} \ \{1\} \subseteq J; \ [\text{R-Inch}]$$

and

$$\frac{\rho_{i_1} \dashv^s \sigma_{i_1} \ \ldots \ \rho_{i_1} \dashv^s \sigma_{i_1}}{\bigoplus_{i\in I} !\mathtt{l}_i.\rho_i \dashv^s \sum_{j\in J} ?\mathtt{l}_j.\sigma_j} \ I \subseteq J; \ [\text{R-Inch}]$$

If the internal move is due to the axiom [A-In-R] the argument is analogous. □

Corollary 3.3.7. *Let $\rho \dashv^s \sigma$. The following statements are true,*

(a) if $\rho \xrightarrow{\tau} \rho'$ then $\rho' \dashv^s \sigma$

(b) if $\sigma \xrightarrow{\tau} \sigma'$ then $\rho \dashv^s \sigma'$

Proof. We prove (a). We apply Lemma 3.3.6 to $\dashv^s$,

$$\begin{array}{lcll} \{\,(\rho',\, \sigma)\,\} \cup \dashv^s & \subseteq & \mathcal{F}^{\dashv s}_{\text{MUST}^s}(\{\,(\rho',\, \sigma)\,\} \cup \dashv^s) & \text{By Lemma 3.3.6} \\ \{\,(\rho',\, \sigma)\,\} \cup \dashv^s & \subseteq & \nu X.\mathcal{F}^{\dashv s}_{\text{MUST}^s}(X) & \text{By the Knaster-Tarski theorem} \\ & = & \dashv^s & \text{By Definition 3.3.1} \end{array}$$

From the argument above, it follows that $\rho' \dashv^s \sigma$. The proof of (b) is similar. □

Lemma 3.3.8. Let $\mathcal{R} \subseteq \mathcal{F}^{\dashv s}_{\text{MUST}^s}(\mathcal{R})$. If $\rho \,\mathcal{R}\, \sigma$ and $\rho \,||\, \sigma \overset{\tau}{\not\longrightarrow}$ then $\rho \xrightarrow{\checkmark}$.

Proof. From the hypothesis $\rho \,||\, \sigma \overset{\tau}{\not\longrightarrow}$, it follows ρ, and σ are stable. Lemma 2.3.7 states the ensuing equalities,

$$\text{UNF}(\rho) = \rho, \qquad \text{UNF}(\sigma) = \sigma$$

These equalities implies that $\mathit{depth}(\rho) + \mathit{depth}(\sigma) = 0$.

In view of the restrictive syntax of session contracts, the fact that ρ and σ are stable, ensures that these terms are not defined by an internal choice.

Observe now that the rule instantiation that proves $(\rho,\, \sigma) \in \mathcal{F}^{\dashv s}_{\text{MUST}^s}(\mathcal{R})$ does not involve the rules [R-Alpha], [R-Exch], and [R-Inch], because in all these cases ρ and σ would engage in an interaction, and this cannot happen by hypothesis ($\rho \,||\, \sigma \overset{\tau}{\not\longrightarrow}$). Further, the rules [R-Unfold] and [R-Fold!!!!!!!!] can not have been used either, because $\mathit{depth}(\rho) + \mathit{depth}(\sigma) = 0$.

It follows that the proof of $(\rho,\, \sigma) \in \mathcal{F}^{\dashv s}_{\text{MUST}^s}(\mathcal{R})$ must be due to the axiom [A-Unit], and so $\rho = 1$. It follows that $\rho \xrightarrow{\checkmark}$. □

Lemma 3.3.9. The relation $\dashv^s$ is a co-inductive compliance. Formally, $\dashv^s \subseteq \mathcal{F}^{\dashv}_{(}\dashv^s)$.

Proof. We have to prove that if $\rho \dashv^s \sigma$, then $(\rho,\, \sigma) \in \mathcal{F}_{\dashv}(\dashv^s)$. To this aim, we have to prove that the pair $(\rho,\, \sigma)$ enjoys three properties,

a) if $\rho \Downarrow$ then $\sigma \Downarrow$

b) if $\rho \mid\mid \sigma \overset{\tau}{\not\longrightarrow}$, then $\rho \xrightarrow{\checkmark}$

c) if $\rho \mid\mid \sigma \xrightarrow{\tau} \rho' \mid\mid \sigma'$, then $\rho' \mathcal{R} \sigma'$

Fix a pair $\rho \dashv^s \sigma$. Condition a) is true because every session contract converges. If $\rho \mid\mid \sigma \overset{\tau}{\not\longrightarrow}$, then Lemma 3.3.8 ensures that $\rho \xrightarrow{\checkmark}$. We have proven b). If $\rho \mid\mid \sigma \xrightarrow{\tau} \rho' \mid\mid \sigma'$, then our work amounts in checking that $\rho' \dashv^s \sigma'$; the argument depends on the rule used to infer the internal move.

If [P-Left] or [P-Right] were applied, then $\rho' \dashv^s \sigma'$ follow from Corollary 3.3.7.

If the internal move is due to rule [P-Synch], then we know that its premises are true,

$$\frac{\rho \xrightarrow{\alpha} \rho' \quad \sigma \xrightarrow{\overline{\alpha}} \sigma'}{\rho \mid\mid \sigma \xrightarrow{\tau} \rho' \mid\mid \sigma'} \text{ [P-Synch]}$$

The argument is by case analysis on the form of ρ. Since ρ and σ performs visible actions, we have $\text{UNF}(\rho) = \rho$, and $\text{UNF}(\sigma) = \sigma$, neither ρ nor σ have a top-most recursive constructor.

i) If $\rho = \alpha.\rho'$ and α contains no labels, then $(\rho, \sigma) \in \mathcal{F}^{\dashv^s}_{\text{MUST}^s}(\dashv^s)$ must be proven by the derivation

$$\frac{\rho' \dashv^s \sigma'}{\alpha.\rho' \dashv^s \beta.\sigma'} \; \alpha \bowtie_c \beta; \text{ [R-Alpha]}$$

The premises of the rule ensure that $\rho' \dashv^s \sigma'$.

ii) If $\rho = !\mathtt{l}.\rho'$, then then $(\rho, \sigma) \in \mathcal{F}^{\dashv^s}_{\text{MUST}^s}(\dashv^s)$ must be proven by the derivation

$$\frac{\rho' \dashv^s \sigma'}{!\mathtt{l}_1.\rho' \dashv^s ?\mathtt{l}_1.\sigma'} \text{ [R-Inch]}$$

Again, the premises of the rule ensure that $\rho' \dashv^s \sigma'$.

iii) If $\rho = ?\mathtt{l}.\rho'$ the argument is similar to the one used in the previous case; the only difference being that now we use rule [R-Exch] in place of [R-Inch].

□

Lemma 3.3.10. [Syntactic characterisation compliance]
For every $\rho, \sigma \in \mathsf{SC}_{\text{fo}}$, $\rho \dashv \sigma$ if and only if $\rho \dashv^s \sigma$.

Proof. Suppose that $\rho \dashv \sigma$; then Lemma 3.3.9 and the Knaster-Tarski theorem ensure that $\rho \dashv^s \sigma$. Suppose that $\rho \dashv^s \sigma$, then Lemma 3.3.4 and the Knaster-Tarski theorem ensure that $\rho \dashv \sigma$. □

3.3.2 Syntactic must testing

Definition 3.3.11. [Syntactic must testing]
In Figure 3.2 replace the symbol $\dashv^s$ with the symbol MUST^s. Lemma C.0.20 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}^{\dashv^s}_{\text{MUST}^s}(X)$; we call this solution the *syntactic* must, and we denote it MUST^s: That is $\text{MUST}^s = \mu X.\mathcal{F}^{\dashv^s}_{\text{MUST}^s}(X)$. □

The syntactic must is the inverse of the must relation on session contracts. We explain first why the syntactic must is contained in the inverse of must (Lemma 3.3.13), and then we go on showing the converse.

The maximal computations performed by session contracts are finite.

Lemma 3.3.12. For every $\rho, \sigma \in \mathsf{SC}_{\text{fo}}$, let C be a computation of $\rho \mid\mid \sigma$;

a) if a derivative of ρ is succesful, then the computation C is finite

b) if a derivative of σ is successful, then the computation is finite

Proof. Consider a successful computation of $\rho \parallel \sigma$ that contains a state $\rho' \parallel \sigma'$ that is client-successful. Since $\rho \xrightarrow{\checkmark}$, point (i) of Lemma 2.3.9 implies that $\rho' = \mathbf{1}$. Since σ' is a session contract, it must converge, $\sigma' \Downarrow$. As ρ' offers no visible action and is stable, the computation has to be finite.

A symmetric argument let us prove point (b). □

Lemma 3.3.13. For every $\rho, \sigma \in \mathsf{SC}_{\mathrm{fo}}$, if ρ MUST$^{\mathrm{s}}$ σ, then σ MUST ρ.

Proof. Let ρ MUST$^{\mathrm{s}}$ σ; as MUST$^{\mathrm{s}}$ is defined inductively, this means that there exists a finite derivation tree

$$\frac{\vdots}{\rho \text{ MUST}^{\mathrm{s}} \sigma} \tag{3.2}$$

done by using the rules in Figure 3.2. The proof of this lemma is by induction on the derivation above, with a case analysis on the last rule applied.

In the base case the last rule used is [AX-SMST], thus $\rho = \mathbf{1}$; it follows that σ MUST ρ.

Now we discuss the inductive cases.

- If the last rule applied was [R-ALPHA] then the derivation in (3.2) is

$$\frac{\dfrac{\vdots}{\rho' \text{ MUST}^{\mathrm{s}} \sigma'}}{?\alpha.\rho' \text{ MUST}^{\mathrm{s}} !\overline{\alpha}.\sigma'} \text{ [R-ALPHA]}$$

  $\rho = ?\alpha.\rho'$, $\sigma = !\overline{\alpha}.\sigma'$, and ρ' MUST$^{\mathrm{s}}$ σ'. Since the derivation of ρ' MUST$^{\mathrm{s}}$ σ' is shorter than the derivation of ρ MUST$^{\mathrm{s}}$ σ, we can apply the inductive hypothesis and state that σ' MUST ρ'. All the maximal computations of $\sigma \parallel \rho$ contain the state $\sigma' \parallel \rho'$, thus Lemma 3.1.7 let us conclude that σ MUST ρ.

- If the last rule applied was [R-EXCH] then the derivation in (3.2) ends as follows,

$$\frac{\dfrac{\vdots}{\rho_1 \text{ MUST}^{\mathrm{s}} \sigma_1} \quad \ldots \quad \dfrac{\vdots}{\rho_n \text{ MUST}^{\mathrm{s}} \sigma_n}}{\sum_{i\in[1,k]} ?\mathtt{l_i}.\rho_i \text{ MUST}^{\mathrm{s}} \bigoplus_{j\in[1,n]} !\mathtt{l}_j.\sigma_j} \; n \leq k; \text{ [R-EXCH]}$$

  and so $\sigma = \bigoplus_{j\in[1,n]} !\mathtt{l}_j.\sigma_j$, $\rho = \sum_{i\in[1,k]} ?\mathtt{l_i}.\rho_i$, $n \leq k$. For every $i \in [1,n]$ the derivation of ρ_i MUST$^{\mathrm{s}}$ σ_i is shorter than the derivation of ρ MUST$^{\mathrm{s}}$ σ, thus we can apply the inductive hypothesis to state that if $i \in [1,n]$ then σ_i MUST ρ_i. Let $\mathcal{S} = \{ (\sigma_i, \rho_i) \mid i \in [1,n] \}$; we have proven that $\mathcal{S} \subseteq$ MUST. Since all the computations of $\sigma \parallel \rho$ contain a state whose components are in $\mathcal{S}$, Lemma 3.1.7 ensures that σ MUST ρ.

- If the last rule used was [R-INCH] the argument is similar.

- If the last rule applied was [R-UNFOLD], then (3.2) is

$$\frac{\dfrac{\vdots}{\text{UNF}(\rho) \text{ MUST}^{\mathrm{s}} \text{ UNF}(\sigma)}}{\rho \text{ MUST}^{\mathrm{s}} \sigma} \; \mathit{depth}(\rho) + \mathit{depth}(\sigma) > 0, \text{ [R-UNFOLD]}$$

  The inductive hypothesis implies that UNF(σ) MUST UNF(ρ). An application of Lemma 3.1.9 shows that σ MUST ρ.

□

Lemma 3.3.14. [Inductive characterisation MUST]
Let ρ and σ be session contracts; σ MUST ρ if and only if ρ MUST^{s} σ.

Proof. We have to prove two set inclusions, namely

(a) $\text{MUST}^{\text{s}} \subseteq \text{MUST}^{-1}$, and

(b) $\text{MUST}^{-1} \subseteq \text{MUST}^{\text{s}}$

We have proven the set inclusion (a) in Lemma 3.3.13, so we prove just the second set inclusion (b). To this end, for every pair σ MUST ρ, we have to exhibit a finite derivation as the following one,

$$\frac{\vdots}{\rho \ \text{MUST}^{\text{s}} \ \sigma}$$

by using the rules in Figure 3.2.

Let be σ and ρ be two session contracts such that σ MUST ρ. Lemma 3.3.12 ensures that all the maximal computations of $\sigma \mid\mid \rho$ are finite. Since the LTS of session contracts is finite state we know that there exists a longest maximal computation of $\sigma \mid\mid \rho$, whose length we denote n. We proceed by induction on n. The intuition being that we have to mimic each synchronisation that takes place between σ and ρ with one of the inference rules in Figure 3.2, the axiom being used if no synchronisation happens.

In the base case $n = 0$, so the longest maximal computation is $\rho \mid\mid \sigma \stackrel{\tau}{\not\longrightarrow}$. Since by assumption σ MUST ρ, Definition 3.1.1 implies that $\rho \stackrel{\checkmark}{\longrightarrow}$; point (i) of Lemma 2.3.9 implies that $\rho = \mathbf{1}$, and so we can derive

$$\frac{}{\rho \ \text{MUST}^{\text{s}} \ \sigma} \ [\text{AX-SMST}]$$

In the inductive case $n = m + 1$ for some $m \in \mathbb{N}$. It follows that there exists a reduction $\rho \mid\mid \sigma \stackrel{\tau}{\longrightarrow} \rho' \mid\mid \sigma'$. The rest of the argument is a case analysis on the rule used to derive this internal move. There are three cases, due respectively to the rules [P-SYNCH], [P-LEFT], and [P-RIGHT]; we discuss only two of them.

If rule [P-SYNCH] was applied then this derivation exists

$$\frac{\rho \stackrel{\alpha}{\longrightarrow} \rho' \quad \sigma \stackrel{\overline{\alpha}}{\longrightarrow} \sigma'}{\rho \mid\mid \sigma \stackrel{\tau}{\longrightarrow} \rho' \mid\mid \sigma'} \ [\text{P-SYNCH}]$$

Our reasoning in this case amounts in two parts; first we justify the use of the inductive hypothesis on the pair (ρ', σ'), and then we show a derivation of ρ MUST^{s} σ.

Since ρ engages in a visible action, Lemma 2.3.10 imply that $\rho \stackrel{\checkmark}{\not\longrightarrow}$, and so the state $\rho \mid\mid \sigma$ is not successful. It follows that σ' MUST ρ'. The longest maximal computation of $\rho' \mid\mid \sigma'$ must have length m, so we can use the inductive hypothesis to state that there exists a derivation

$$\frac{\vdots}{\rho' \ \text{MUST}^{\text{s}} \ \sigma'}$$

To extend the derivation of one further step we have to know which one of the inference rules to apply; the choice depends on the form of α. We discuss one case; if $\alpha = !\mathtt{t}$ then $\sigma = ?\mathtt{t}.\sigma'$ and $\rho = !\mathtt{t}.\rho'$.

Since α contains no labels we can apply rule [R-ALPHA],

$$\dfrac{\dfrac{\vdots}{\rho' \text{ MUST}^{\text{S}} \sigma'}}{!\mathtt{t}.\rho' \text{ MUST}^{\text{S}} ?\mathtt{t}.\sigma'} \text{ [R-ALPHA]}$$

If $\alpha = ?\mathtt{t}$ then the rule to use is [R-ALPHA], if $\alpha = ?\mathtt{l}$ the rule to use if [R-EXCH], and if $\alpha = !\mathtt{l}$ then the rule to be applied is [R-INCH].

We have discussed the case in which the internal move of $\rho \mid\mid \sigma \xrightarrow{\tau} \rho' \mid\mid \sigma'$ is due to [P-SYNCH].

If rule [P-RIGHT] was applied then $\sigma \xrightarrow{\tau} \sigma'$ and $\rho = \rho'$; the last equality implying that σ MUST ρ'. Since $\sigma \xrightarrow{\tau} \sigma'$ the session contract σ must be defined by an internal choice or a recursion.

- If σ is defined by an internal choice, since σ is a session contract it must be

$$\sigma = \bigoplus_{i\in[1,m]} !\mathtt{l_i}.\sigma_m = (\bigoplus_{i\in[1,m-1]} !\mathtt{l_i}.\sigma_i) \oplus !\mathtt{l}_m.\sigma_m$$

  for some $m \in \mathbb{N}$. It must be $\sigma' = \bigoplus_{i\in[1,m-1]} !\mathtt{l_i}.\sigma_i$ or $\sigma' = !\mathtt{l}_m.\sigma_m$; in both cases the hypothesis σ MUST ρ implies that σ' MUST ρ', moreover the longest maximal computation of $\sigma' \mid\mid \rho'$ has length at most m. It follows that we can apply he inductive hypothesis to exhibit two derivations, namely

$$\dfrac{\vdots}{\rho \text{ MUST}^{\text{S}} \bigoplus_{i\in[1,k-1]} !\mathtt{l_i}.\sigma_i} \qquad \dfrac{\vdots}{\rho \text{ MUST}^{\text{S}} !\mathtt{l}_k.\sigma_k}$$

$$(\alpha) \qquad\qquad (\beta)$$

  Note that the last rule used in the derivation (α) must be [R-EXCH], we have therefore, that $\rho = \sum_{j\in[1,p]} ?\mathtt{l}_j.\rho_j$, with $k \leq p$. The inference (β) must be due to rule [R-EXCH] as well, so it ensures that for some $\hat{p} \in [1,p]$,

$$\dfrac{\vdots}{\rho_{\hat{p}} \text{ MUST}^{\text{S}} \sigma_k}$$

  A similar argument holds for the derivation (α), thus for every $i \in [1, k-1]$, a derivation

$$\dfrac{\vdots}{\rho_i \text{ MUST}^{\text{S}} \sigma_i}$$

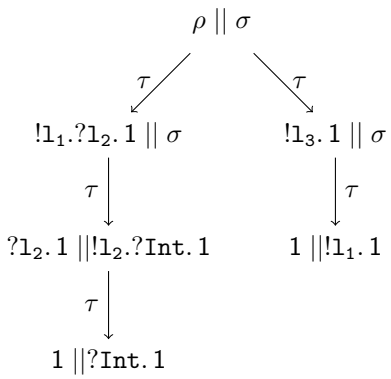
  must exist. Now we know enough to show the following derivation

$$\dfrac{\dfrac{\vdots}{\rho_1 \text{ MUST}^{\text{S}} \sigma_1} \quad \ldots \quad \dfrac{\vdots}{\rho_{\hat{p}} \text{ MUST}^{\text{S}} \sigma_m}}{\sum_{j\in[1,p]} ?\mathtt{l}_j.\rho_j \text{ MUST}^{\text{S}} \bigoplus_{i\in[1,m]} !\mathtt{l_i}.\sigma_m} \; m \leq p; \text{ [R-EXCH]}$$

- Suppose that σ is defined by recursion and $\sigma \xrightarrow{\tau} \sigma'$ is due to [A-UNF]. The assumption implies that $\mathit{depth}(\sigma) > 0$. Lemma 3.1.9 implies that UNF(σ) MUST UNF(ρ). As $\rho \mid\mid \sigma \Longrightarrow$ UNF$(\rho) \mid\mid$ UNF(σ), the successful computation of UNF$(\rho) \mid\mid$ UNF(σ) has length at most m, so we can use the inductive hypothesis; there exists a derivation of

$$\dfrac{\vdots}{\text{UNF}(\rho) \text{ MUST}^{\text{S}} \text{ UNF}(\sigma)}$$

  Since UNF(σ) = UNF(σ') and $\mathit{depth}(\sigma) > 0$ we can extend the derivation above by applying rule



$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{1} \text{ MUST}^{\mathrm{s}} ?\mathtt{Int}.\mathtt{1}}\ [\text{AX-SMST}]}{?\mathtt{l}_2.\mathtt{1} \text{ MUST}^{\mathrm{s}} !\mathtt{l}_2.?\mathtt{Int}.\mathtt{1},}\ [\text{R-BRANCH}]}{!\mathtt{l}_1.?\mathtt{l}_2.\mathtt{1} \text{ MUST}^{\mathrm{s}} ?\mathtt{l}_1.!\mathtt{l}_2.?\mathtt{Int}.\mathtt{1},}\ [\text{R-BRANCH}] \qquad \dfrac{\dfrac{}{\mathtt{1} \text{ MUST}^{\mathrm{s}} !\mathtt{l}_1.\mathtt{1}}\ [\text{AX-SMST}]}{!\mathtt{l}_3.\mathtt{1} \text{ MUST}^{\mathrm{s}} ?\mathtt{l}_3.!\mathtt{l}_1.\mathtt{1}}\ [\text{R-BRANCH}]}{\rho \text{ MUST}^{\mathrm{s}} \sigma}\ [\text{R-BRANCH}]$$

Figure 3.3: Successful computations of $\rho \,||\, \sigma$ and derivation of $\rho \text{ MUST}^{s} \sigma$ (see Example 3.3.15)

[R-UNFOLD],

$$\dfrac{\dfrac{\vdots}{\text{UNF}(\rho) \text{ MUST}^{\mathrm{s}} \text{ UNF}(\sigma)}}{\rho \text{ MUST}^{\mathrm{s}} \sigma'}\ \mathit{depth}(\rho) + \mathit{depth}(\sigma) > 0,\ [\text{R-UNFOLD}]$$

If rule [P-LEFT] was applied then the argument is symmetric to the one used for rule [P-RIGHT]. □

We comment on the previous result. Its proof exhibits how the structure of the derivation of $\rho \text{ MUST}^{\mathrm{s}} \sigma$ mirrors the internal moves that appear in the successful computations of $\rho \,||\, \sigma$; that is , each application of one the rules [P-LEFT], [P-RIGHT], and [P-SYNCH] is mirrored by an application of a (series of) rules of Figure 3.2. We give an example of this fact.

Example 3.3.15. [Succesful computations and derivations of MUST^{s}]
In this example we show one instance of the mentioned similarity between the successful computations of a composition $\rho \,||\, \sigma$ such that $\sigma \text{ MUST } \rho$, and the derivation of $\rho \text{ MUST}^{\mathrm{s}} \sigma$.

Consider the session contracts

$$\begin{aligned} \sigma &= ?\mathtt{l}_1.!\mathtt{l}_2.?\mathtt{Int}.\mathtt{1} + ?\mathtt{l}_3.!\mathtt{l}_1.\mathtt{1} \\ \rho &= !\mathtt{l}_1.?\mathtt{l}_2.\mathtt{1} \oplus !\mathtt{l}_3.\mathtt{1} \end{aligned}$$

In Figure 3.3 we have depicted the successful computations of the composition $\rho \,||\, \sigma$, and the derivation which proves that $\rho \text{ MUST}^{\mathrm{s}} \sigma$.

The similarity between the pairs of session contracts that appear in the computations and the ones that appear in the derivation is striking. To begin with, the successful states of the computations are the ones derived by applying the axiom [AX-SMST]. Secondly, each time a composition $\rho' \,||\, \sigma'$ performs an interaction, the terms in the reduct appear in the premises of the derivation of $\rho' \text{ MUST}^{\mathrm{s}} \sigma'$. □

According to what we have stated so far, the meaning of Lemma 3.3.14 is that, given a pair σ, ρ, there is a correspondence between the computational tree made by the successful computations of $\rho \,||\, \sigma$, and the proof that $\rho \text{ MUST}^{\mathrm{s}} \sigma$.

3.4 Related Work

The MUST testing of Definition 3.1.1 is taken directly from [De Nicola and Hennessy, 1984; Hennessy, 1985]; the only novelty is that in our framework processes and tests are not distinguished; for instance $\mathsf{1}$ is a process. This is not the case in the original presentations of [De Nicola and Hennessy, 1984], in the sense that there processes cannot perform the action ω. To mix tests and processes is necessary in order to express the notion of peer, and render Definition 3.1.3 sensible.

The first compliance defined in terms of the interactions of clients, servers and the action $\checkmark$ is the *behavioural compliance* [Laneve and Padovani, 2007]; let us denote it $\dashv^{\mathsf{bhv}}$.

The comparison of our results with the work of Laneve and Padovani is complicated by the fact that in their theory compliance judgements take the form $\mathrm{I}_1[\rho] \dashv \mathrm{I}_2[\sigma]$ where I_1, I_2 are finite sets of actions representing in some sense the interfaces of the processes guaranteeing the contracts; moreover, for a contract $\mathrm{I}[\sigma]$ to be valid its interface I has to contain all the action names (including $\checkmark$) that appear in the behaviour σ. Let us refer to the pairs $\mathrm{I}[\sigma]$ as *constrained contracts*. It is relatively easy to prove the following facts,

$$\mathsf{1} \not\dashv \tau^\infty \qquad\qquad \emptyset[\mathsf{1}] \dashv^{\mathsf{bhv}} \emptyset[\tau^\infty]$$

$$\tau^\infty \dashv \tau^\infty \qquad\qquad \emptyset[\tau^\infty] \not\dashv^{\mathsf{bhv}} \emptyset[\tau^\infty]$$

Regardless of the difference between the relations $\dashv$ and $\dashv^{\mathsf{bhv}}$, Example 3.2.13 and Example 3.2.14 are similar to the examples used by Laneve and Padovani. Moreover, the results we will show in Chapter 5 can be adapted to the framework that arise from $\dashv^{\mathsf{bhv}}$, as long as we disregard the interfaces and work only with the LTS.

The most recent compliance theories are [Padovani, 2010] and [Castagna et al., 2009].

Let us denote $\dashv^{\mathsf{str}}$ the *strong compliance* of [Padovani, 2010, Definition 2.1]. In that paper the LTS is

$$\langle\, \mathsf{CCS}_{\mathrm{w}\tau}^{\mathsf{rec},\mathsf{fb},\Downarrow},\ \mathsf{Act}_{\tau\checkmark},\ \longrightarrow \rangle \tag{3.3}$$

where the states are terms of a recursive version of $\mathsf{CCS}_{\mathrm{w}\tau}$, and the LTS is convergent and finite state. Up-to the encoding of recursive terms into our syntax, the LTS in Eq. (3.3) is contained in our LTS, and our relation $\dashv$ and the relation $\dashv^{\mathsf{str}}$ coincide.

Now we discuss the compliance of [Castagna et al., 2009]. A striking difference between our setting and the theory of that paper is that their LTS is deterministic, while ours is not.

Let $\mathsf{CCS}_{\mathrm{w}\tau}^{\mathsf{coind},\Downarrow}$ denote the language (a) generated *co-inductively* by the grammar for finite terms of $\mathsf{CCS}_{\mathrm{w}\tau}$, and (b) whose terms are regular (in the sense of [Courcelle, 1983, Section 4.1]) and convergent.

Castagna et al. define the transitions of the following LTS by using inference rules that are not the standard ones for CCS;

$$\langle\, \mathsf{CCS}_{\mathrm{w}\tau}^{\mathsf{coind},\Downarrow},\ \mathsf{Act}_{\checkmark},\ \longmapsto \rangle \tag{3.4}$$

The rules are designed so that if $\sigma \overset{\alpha}{\longmapsto} \sigma'$ and $\sigma \overset{\alpha}{\longmapsto} \sigma''$ then $\sigma' = \sigma''$. This implies that the acceptance sets of contracts à la Castagna et al. contain at most one ready set. Note also that the set of labels $\mathsf{Act}_{\checkmark}$ contains no τ, there are no τ transitions in the LTS of Eq. (3.4).

The LTS that we used, $\langle\, \mathsf{CCS}_{\mathrm{w}\tau},\ \mathsf{Act}_{\tau\checkmark},\ \longrightarrow \rangle$, is more general than the LTS in Eq. (3.4), in the following sense. The transitions are **not** deterministic: $p \overset{\alpha}{\longrightarrow} p'$ and $p \overset{\alpha}{\longrightarrow} p''$ do **not** imply that $p' = p''$; and so the acceptance set of a process p after a trace s may contain an infinite amount of ready sets. Moreover processes may diverge. The LTS used by Castagna et al., though, is not contained in our LTS, because of the difference in the transitions. Consider the term $p = \alpha.\,\mathsf{1} + \alpha.\,\mathsf{0}$; by using the rules of that paper one can infer

$$p \overset{\alpha}{\longmapsto} \mathsf{1} \oplus \mathsf{0}, \qquad p \not\overset{\alpha}{\longmapsto} \mathsf{0}, \qquad p \not\overset{\alpha}{\longmapsto} \mathsf{1}$$

whereas in our setting

$$p \overset{\alpha}{\not\longrightarrow} 1 \oplus 0, \qquad p \xrightarrow{\alpha} 0, \qquad p \xrightarrow{\alpha} 1$$

In principle it is not clear whether our compliance relation coincides with their strong compliance (Definition 2.6 in their paper), $\dashv^{09}$.

A relation similar to $\dashv^{09}$ but presented in a different setting, is the *Type Compliance*, denoted $\propto$, of [Acciai and Boreale, 2008, Definition 5]. There the states of the LTS are basic parallel processes [Christensen et al., 1993] and the transitions are given by standard inference rules of operational semantics. We leave as an open problem the comparison of $\propto$ with $\dashv$ (see (Q1) in Section 11.2).

The syntactic characterisation of compliance is similar to [Barbanera and de'Liguoro, 2010, Proposition 2.9]. To the best of our knowledge, the syntactic characterisations of the MUST testing on session contracts is original.

Fair theories Other compliances have been proposed, which are inspired to the fair testing of [Rensink and Vogler, 2007] rather than the MUST testing. One such theory has been presented in [Bravetti and Zavattaro, 2009]. We will compare our results with the theory of Bravetti and Zavattaro in the oncoming chapters.

Chapter 4

Must pre-orders

This chapter is devoted to the study of the pre-orders that arise from the MUST relation (see Definition 3.1.1). Processes can be seen as clients, servers, or peers, so there are three pre-orders: a server pre-order, a client pre-order, and a peer pre-order. They are denoted respectively

$$\sqsubseteq_{\text{SVR}}, \qquad \sqsubseteq_{\text{CLT}}, \qquad \sqsubseteq_{\text{P2P}} \tag{4.1}$$

The relations in (4.1) let us state when a process is "better" than another one, given the role that we want the processes to play. For instance $\mathsf{0} \sqsubseteq_{\text{CLT}} \mathsf{1}$ means that the process $\mathsf{1}$ is a better *client* than the process $\mathsf{0}$, in the sense that all the servers that satisfy $\mathsf{0}$ satisfy also $\mathsf{1}$. Similarly, the inequality $\alpha.\mathsf{0} \oplus \beta.\mathsf{0} \sqsubseteq_{\text{SVR}} \alpha.\mathsf{0} + \beta.\mathsf{0}$ means that the *server* process $\alpha.\mathsf{0} + \beta.\mathsf{0}$ satisfies all the clients that the process $\alpha.\mathsf{0} \oplus \beta.\mathsf{0}$ satisfies.

The formal definitions of these pre-orders have two disadvantages. First, they use a universal quantification on *all* processes. If we were to prove $r_1 \sqsubseteq_{\text{CLT}} r_2$ by using the definition of $\sqsubseteq_{\text{CLT}}$ (Definition 4.2.1), we would have to prove that all the processes p that satisfy r_1 satisfy also r_2. The quantification on all processes is a burden, as they are infinite, so it is not clear how to account for all of them. Second, it is not clear when two processes are related by the pre-orders in (4.1); that is, as long as we use the definitions of the pre-orders above, we cannot answer questions such as

$$\text{under which conditions are } p_1 \text{ and } p_2 \text{ related by } \sqsubseteq_{\text{CLT}} ? \tag{4.2}$$

To solve these issues we introduce reasoning techniques for the pre-orders in (4.1). Throughout this chapter we will introduce such notions as to let us define three relations, namely

$$\precsim_{\text{SVR}}, \qquad \precsim_{\text{CLT}}, \qquad \precsim_{\text{P2P}} \tag{4.3}$$

We will prove that each one of the "alternative" relations above equal one of the pre-orders in (4.1). We refer to the relations in Eq. (4.3) as the *behavioural characterisations* of the MUST pre-orders. The equalities between the relations in (4.1) and the relations in (4.3) have two advantages.

- The alternative relations are defined by using properties that describe the observable behaviour of processes; so the alternative pre-orders lay bare how the relations in (4.1) relate the operational semantics of processes. This sheds light on what it means for two processes to be related by one of the pre-orders in (4.1), and lets us answer questions such as the one in (4.2)

- The definitions of the behavioural characterisations do not present a universal quantification on processes; rather, they explain why two processes, say p and q, are related, just by examining p and q themselves.

The behavioural characterisation of the server pre-order, gives us a straightforward way to prove when two processes are related by $\sqsubseteq_{\mathrm{SVR}}$ (see Example 4.1.22).

Structure of the chapter. We begin our study by discussing the server pre-order, as it is the simplest of the three relations. The server pre-order is a mild generalisation of the testing pre-order [De Nicola and Hennessy, 1984; Hennessy, 1985], in that it relates terms that (a) can report success and (b) can be infinite branching, whereas the standard presentation of the testing pre-order does not. Nevertheless, the behavioural characterisation (Theorem 4.1.21) of $\sqsubseteq_{\mathrm{SVR}}$ coincides with the well-known characterisation of the testing pre-order [Hennessy, 1985]. In Section 4.2 we use the characterisation of the server pre-order as starting point to characterise the client pre-order $\sqsubseteq_{\mathrm{CLT}}$. By means of examples we expose the subtleties of the pre-order for clients, we explain why the characterisation of the server pre-order fails to capture the client pre-order, and then we show how to amend that characterisation so as to describe $\sqsubseteq_{\mathrm{CLT}}$ (Theorem 4.2.37). In Section 4.3 we discuss the peer pre-order. In particular, we use the machinery devised in the earlier sections to characterise the peer pre-order; the discussion is facilitated by the fact that Theorem 4.1.21 and Theorem 4.2.37 have the same form.

4.1 Server pre-order

We begin our investigation of the MUST pre-orders by unravelling the characteristic properties of the server pre-order. We consider a server p_2 better than a server p_1 if p_2 satisfies all the clients that are satisfied by p_1. This intuition is formalised by the oncoming definition.

Definition 4.1.1. [MUST server-pre-order]
We write $p_1 \sqsubseteq_{\mathrm{SVR}} p_2$ whenever p_1 MUST r implies p_2 MUST r. We refer to the relation denoted by the symbol $\sqsubseteq_{\mathrm{SVR}}$ as the MUST *server pre-order* . □

Notation Throughout this thesis we will at times use the symbols $\sum$ and $\bigoplus$ to write processes, for instance

$$\begin{array}{ll} \sum_{i\in[1;m]} p_i & \text{in place of} \quad p_1 + p_2 + \ldots + p_m \\ \bigoplus_{i\in[1;n]} p_i & \text{in place of} \quad p_1 \oplus p_2 \oplus \ldots \oplus p_n \end{array}$$

This is by justified to the fact that

$$\begin{array}{ll} p \oplus r \eqsim_{\mathrm{SVR}} r \oplus p & p \oplus (p' \oplus p'') \eqsim_{\mathrm{SVR}} (p \oplus p') \oplus p'' \\ p + r \eqsim_{\mathrm{SVR}} r + p & p + (p' + p'') \eqsim_{\mathrm{SVR}} (p + p') + p'' \end{array}$$

where $\eqsim_{\mathrm{SVR}}$ is the equivalence relation given in the obvious way by $\sqsubseteq_{\mathrm{SVR}}$.

In order to acquaint ourselves with the relation $\sqsubseteq_{\mathrm{SVR}}$, we prove a lemma and discuss two examples.

The pre-order $\sqsubseteq_{\mathrm{SVR}}$ has bottom elements, namely all the processes that diverge.

Lemma 4.1.2. [Bottom elements]
If $p_1 \Downarrow$ then $p_1 \sqsubseteq_{\mathrm{SVR}} p_2$ for every process p_2.

Proof. We have to show that if p_1 MUST r then p_2 MUST r, for every process r. Suppose that p_1 MUST r; the hypothesis that $p_1 \Downarrow$ and Lemma 3.1.6 imply that $r \xrightarrow{\checkmark}$. It follows that p_2 MUST r for every p_2. □

We discuss why some processes are (not) related by the server pre-order.

Example 4.1.3. The relation $p \sqsubseteq_{\mathrm{SVR}} q$ essentially compares the interactions *offered* by the servers p and q. For instance,

$$\begin{array}{lll} \alpha.\mathsf{1} & \sqsubseteq_{\mathrm{SVR}} & \alpha.\mathsf{0} \\ \mathsf{1} & \sqsubseteq_{\mathrm{SVR}} & \mathsf{1} \oplus \mathsf{1} \\ \alpha.\mathsf{1} & \not\sqsubseteq_{\mathrm{SVR}} & \mathsf{1} \oplus \mathsf{1} \\ p & \not\sqsubseteq_{\mathrm{SVR}} & \mathsf{0} \end{array}$$

where $p = \alpha.(\beta.\mathsf{0} + \delta.\mathsf{1}) + \alpha.(\beta.\mathsf{1} + \delta.\mathsf{0})$.

The first inequality, $\alpha.\mathsf{1} \sqsubseteq_{\mathrm{SVR}} \alpha.\mathsf{0}$, is true because the server $\alpha.\mathsf{1}$ offers to the clients as much interaction as $\alpha.\mathsf{0}$; that is the action α.

The same argument lets us prove also the second inequality, which is an instance of the more general fact $p \sqsubseteq p \oplus p$ for every process p. To see why the general inequality is true, note that all the maximal computations of $r \,||\, p \oplus p$ are also maximal computations of $r \,||\, p$; so p MUST r implies that $p \oplus p$ MUST r.

To prove the third inequality we have to exhibit a client that is satisfied by $\alpha.\mathsf{1}$ and not by $\mathsf{1} \oplus \mathsf{1}$. Consider $r = \overline{\alpha}.\mathsf{1}$. Plainly, $\alpha.\mathsf{1}$ MUST r, whereas

$$r \,||\, \mathsf{1} \oplus \mathsf{1} \xrightarrow{\tau} r \,||\, \mathsf{1} \stackrel{\tau}{\not\longrightarrow}$$

and $r \stackrel{\checkmark}{\not\longrightarrow}$, so $\mathsf{1} \oplus \mathsf{1}$ $\not$MUST r. The crucial difference between $\alpha.\mathsf{1}$ and $\mathsf{1} \oplus \mathsf{1}$ is that the latter server offers fewer interactions than $\alpha.\mathsf{1}$.

A similar argument lets us prove the fourth inequality, for the server $\mathsf{0}$ offers fewer interactions than p; the client $\overline{\alpha}.\overline{\beta}.\overline{\delta}.\mathsf{1}$ is satisfied by p and is not satisfied by $\mathsf{0}$. □

In Example 4.1.3 we have compared only processes that converges. In the next example we discuss divergent processes.

Example 4.1.4.

$$\begin{array}{lll} \mathsf{1} + \tau^\infty & \sqsubseteq_{\mathrm{SVR}} & \beta.\tau^\infty \\ \beta.\mathsf{1} & \not\sqsubseteq_{\mathrm{SVR}} & \mathsf{1} \oplus \tau^\infty \end{array}$$

The first inequality is true because any term that diverges is a bottom element of $\sqsubseteq_{\mathrm{SVR}}$; we have seen this in Lemma 4.1.2.

To prove the second inequality we use the client $\overline{\beta}.\mathsf{1}$, which is satisfied by $\beta.\mathsf{1}$ and not by $\mathsf{1} \oplus \tau^\infty$. Intuitively, $\beta.\mathsf{1}$ cannot be smaller than $\mathsf{1} \oplus \tau^\infty$ because $\beta.\mathsf{1}$ is not a bottom element of $\sqsubseteq_{\mathrm{SVR}}$, whereas $\mathsf{1} \oplus \tau^\infty$ is a bottom element. □

The characterisation of $\sqsubseteq_{\mathrm{SVR}}$ that we will provide is based on a comparison of the observable behaviours of processes. In Lemma 4.1.2 and Example 4.1.4 we have seen that if a process p_1 diverges, then the reasons why $p_1 \sqsubseteq_{\mathrm{SVR}} p_2$ has nothing to do with the relation between the behaviour of p_2 and the behaviour of p_1. It follows that to provide a complete characterisation of $\sqsubseteq_{\mathrm{SVR}}$ we need some notation to express the convergence of processes after a trace.

Definition 4.1.5. [Residuals after trace]
For any process $p \in \mathsf{CCS}_{\mathrm{w}\tau}$ and $s \in Act^\star$ let $(p \text{ AFTER } s) = \{\, q \mid p \stackrel{s}{\Longrightarrow} q \,\}$. □

Definition 4.1.6. [Convergence along trace]
Let $\mathcal{F}_\Downarrow : \mathcal{P}(\mathsf{CCS}_{\mathrm{w}\tau} \times Act^\star) \longrightarrow \mathcal{P}(\mathsf{CCS}_{\mathrm{w}\tau} \times Act^\star)$ be the rule functional given by the inference rules in Figure 4.1. Lemma C.0.21 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_\Downarrow(X)$; we call this solution the *convergence predicate*, and we denote it $\Downarrow$: That is $\Downarrow \;=\; \mu X.\mathcal{F}_\Downarrow(X)$. We extend the relation $\Downarrow$ to infinite strings by letting, for every $u \in Act^\infty$, $p \Downarrow u$, if and only if for every $n \in \mathbb{N}$, $p \Downarrow u_n$. □

$$\frac{}{p \Downarrow \varepsilon}\ p \Downarrow\ ;\ [\text{CONV-AX}]$$

$$\frac{}{p \Downarrow \alpha s}\ p \Downarrow, p \overset{\alpha}{\not\Longrightarrow};\ [\text{CONV-AX-NOT}]$$

$$\frac{\bigoplus(p \text{ AFTER } \alpha) \Downarrow s'}{p \Downarrow \alpha s'}\ p \Downarrow, p \overset{\alpha}{\Longrightarrow};\ [\text{CONV-ALPHA}]$$

Figure 4.1: Inference rules for the functional $\mathcal{F}_{\Downarrow}$

Lemma 4.1.7. Let p be a process; $p \Downarrow s$ if and only if for every s' prefix of s, if $p \overset{s'}{\Longrightarrow} p'$ then $p' \Downarrow$

Proof. We have to show two implications, namely

i) if $p \Downarrow s$ then for every s' prefix of s, if $p \overset{s'}{\Longrightarrow} p'$ then $p' \Downarrow$

ii) if for every s' prefix of s, $p \overset{s'}{\Longrightarrow} p'$ implies $p' \Downarrow$, then $p \Downarrow s$.

We prove them separately. We begin by showing (i). The assumption that $p \Downarrow s$ ensures that there exists a finite derivation

$$\frac{\vdots}{p \Downarrow s} \tag{4.4}$$

We have to prove that

$$\text{for every } s' \text{ prefix of } s, p \overset{s'}{\Longrightarrow} p' \text{ implies } p' \Downarrow . \tag{4.5}$$

The argument is by induction on the derivation in (4.4).

Base case. In this case the whole derivation in (4.4) amounts an application of the axiom [CONV-AX], or of the axiom [CONV-AX-NOT].

If [CONV-AX] was applied then (4.4) is

$$\frac{}{p \Downarrow \varepsilon}\ p \Downarrow\ ;\ [\text{CONV-AX}]$$

It follows that s is the empty string, and so (4.5) requires us to prove that if $p \overset{\varepsilon}{\Longrightarrow} p'$ then $p' \Downarrow$. Fix such a p'; the definition of $\Longrightarrow$ ensures that $p \xrightarrow{\tau}{}^{n} p'$ for some $n \in \mathbb{N}$. Reasoning by induction on n, we can show that the side condition $p \Downarrow$ implies $p' \Downarrow$.

If [CONV-AX-NOT] was applied, then (4.4) is

$$\frac{}{p \Downarrow \alpha s'}\ p \Downarrow, p \overset{\alpha}{\not\Longrightarrow};\ [\text{CONV-AX-NOT}]$$

It follows that $s = \alpha s'$. The side condition $p \overset{\alpha}{\not\Longrightarrow}$ implies that the only prefix of s performed by p is the empty string, so (4.5) requires us to prove that if $p \overset{\varepsilon}{\Longrightarrow} p'$ then $p' \Downarrow$. To prove this we reason as we did discussing the application the axiom [CONV-AX].

Inductive case. In this case the last step in the derivation in (4.4) is rule [CONV-ALPHA], so the derivation in (4.4) is

$$\frac{\dfrac{\vdots}{\bigoplus(p \text{ AFTER } \alpha) \Downarrow s'}}{p \Downarrow \alpha s'}\ p \Downarrow, p \overset{\alpha}{\Longrightarrow};\ [\text{CONV-AX-NOT}]$$

The derivation of $\bigoplus(p \text{ AFTER } \alpha) \Downarrow s'$ is shorter than the derivation above, so we are allowed to use the inductive hypothesis, for every s'' prefix of s', if $\bigoplus(p \text{ AFTER } \alpha) \overset{s''}{\Longrightarrow} p'$ then $p' \Downarrow$.

We prove (4.5). Fix a $\hat{s}$ prefix of s such that $p \stackrel{s'}{\Longrightarrow} p'$. If $\hat{s} = \varepsilon$, then we reason as we did in the base case to prove that $p' \Downarrow$. If $\hat{s} \neq \emptyset$ then $\hat{s} = \alpha s''$ for some s'' prefix of s'. The definition of AFTER ensures that $\bigoplus(r_1 \text{ AFTER } \alpha) \stackrel{s''}{\Longrightarrow} p'$, and so the inductive hypothesis implies that $p' \Downarrow$.

We have proven the first implication of the lemma, namely i. Now we prove the second implication, ii. Let us assume that for every s' prefix of s, $p \stackrel{s'}{\Longrightarrow} p'$ implies $p' \Downarrow$. Our aim is to exhibit a finite derivation of $p \Downarrow s$. First we prove $p \Downarrow$. The string ε is a prefix of every string s, so the assumption and $p \stackrel{\varepsilon}{\Longrightarrow} p$ imply that $p \Downarrow$.

The main part of the argument is by induction on s.

Base case ($s = \varepsilon$) In this case use the axiom [CONV-AX],

$$\frac{}{p \Downarrow \varepsilon}\ p \Downarrow ;\ [\text{CONV-AX}]$$

Inductive case ($s = \alpha s'$) In this case the argument depends on $p \stackrel{\alpha}{\Longrightarrow}$.

If $p \stackrel{\alpha}{\not\Longrightarrow}$, then we use [CONV-AX-NOT]:

$$\frac{}{p \Downarrow \alpha s'}\ p \Downarrow ;\ [\text{CONV-AX-NOT}]$$

If $p \stackrel{\alpha}{\Longrightarrow}$, then the process $\bigoplus(p \text{ AFTER } \alpha)$ is well-defined, so let $\hat{p} = \bigoplus(p \text{ AFTER } \alpha)$. The string s' is shorter than s, so the inductive hypothesis states that

$$\text{if for every } s'' \text{ prefix of } s',\ p' \stackrel{s''}{\Longrightarrow} p'' \text{ implies } p \Downarrow s, \text{ then } p'' \Downarrow s'$$

Let s'' be a prefix of s' such that $\hat{p} \stackrel{s''}{\Longrightarrow} p''$. The construction of $\hat{p}$ implies that $p \stackrel{\alpha s''}{\Longrightarrow} p''$, and $\alpha s''$ is a prefix of s. Our assumption on p implies that $p'' \Downarrow$. This prove that for every prefix s'' of s', $\hat{p} \stackrel{s''}{\Longrightarrow} p''$ implies that $p'' \Downarrow$. The inductive hypothesis now ensures that there exists a finite derivation of $\hat{p} \Downarrow s'$. Now we use [CONV-ALPHA]:

$$\frac{\begin{array}{c}\vdots\\ \bigoplus(p \text{ AFTER } \alpha) \Downarrow s'\end{array}}{p \text{ AFTER } \alpha s'}\ p \Downarrow , p \stackrel{\alpha}{\Longrightarrow};\ [\text{CONV-ALPHA}]$$

□

We are ready to prove the first relation that $\sqsubseteq_{\text{SVR}}$ enforces on the behaviour of a server p_1 that converges (after a trace), and the behaviour of a better server p_2.

Lemma 4.1.8. Let $p_1 \sqsubseteq_{\text{SVR}} p_2$. For every $s \in Act^\star$, if $p_1 \Downarrow s$ then $p_2 \Downarrow s$.

Proof. In view of Lemma 4.1.7, we have to show that

$$\text{for every } s' \text{ prefix of } s,\ \text{if } p_2 \stackrel{s'}{\Longrightarrow} p_2' \text{ then } p_2' \Downarrow \tag{4.6}$$

Let s' be the longest prefix of s performed by p_2, and let $s' = \alpha_1 \alpha_2 \ldots \alpha_n$. To prove the lemma, we define a client C such that

1. p_1 MUST C
2. p_2 MUST C implies (4.6)

For every $0 \leq k \leq n$, let

$$C_k \stackrel{\text{def}}{=} \begin{cases} (1 \oplus 1) + \overline{\alpha}_{k+1}.C_{k+1} & \text{if } 0 \leq k < n \\ 1 \oplus 1 & \text{if } k = n \end{cases}$$

Note that if s' is a prefix of s, and $C_0 \stackrel{\overline{s'}}{\Longrightarrow} r \stackrel{\tau}{\not\longrightarrow}$, then $r \stackrel{\checkmark}{\longrightarrow}$.

We prove that p_1 MUST C_0. We have to show that all the maximal computations of $C_0 \parallel p_1$ are client-successful. By construction $C_0 \Downarrow \overline{s}$ and $\overline{s}$ is the longest trace that C_0 can perform; moreover, by hypothesis $p_1 \Downarrow s$, and s is finite; it follows that all the maximal computations of $C_0 \parallel p_1$ are finite. Fix such a computation,

$$C_0 \parallel p_1 = r_0 \parallel p_1^0 \stackrel{\tau}{\longrightarrow} r_2 \parallel p_1^1 \stackrel{\tau}{\longrightarrow} \ldots \stackrel{\tau}{\longrightarrow} r_k \parallel p_m^1 \stackrel{\tau}{\not\longrightarrow}$$

Since $C_0 \stackrel{\overline{s'}}{\Longrightarrow} r_k \stackrel{\tau}{\not\longrightarrow}$ for some s' prefix of s, it follows that $r_k \stackrel{\checkmark}{\longrightarrow}$; the maximal computation is client-successful. This argument is true for every maximal computation of $C_0 \parallel p_1$, so we have proven that p_1 MUST C_0.

Since p_1 MUST C_0, the hypothesis ensure that p_2 MUST C_0. To prove (4.6) we reason by contradiction. Suppose that (4.6) is false: there exists a k such that $p_2 \stackrel{s_k}{\Longrightarrow} p_2^0$ and p_2^0 diverges; then the following maximal computation is not client-successful,

$$C_0 \parallel p_2 \Longrightarrow C_{k+1} \parallel p_2^0 \stackrel{\tau}{\longrightarrow} C_{k+1} \parallel p_2^1 \stackrel{\tau}{\longrightarrow} C_{k+1} \parallel p_2^2 \stackrel{\tau}{\longrightarrow} \ldots$$

It follows that $p_2 \not$MUST C_0, which contradicts p_2 MUST C_0. In view of this contradiction, (4.6) is true. □

Lemma 4.1.9. Let $p_1 \sqsubseteq_{\text{SVR}} p_2$. For every $s \in Act^\star$, if $p_1 \Downarrow s$ and $p_2 \stackrel{s}{\Longrightarrow}$ then $p_1 \stackrel{s}{\Longrightarrow}$.

Proof. We have to prove that under the hypothesis there exists p_1' such that $p_1 \stackrel{s}{\Longrightarrow} p_1'$. To prove this we define a test A such that

1. $p_2 \not$MUST A
2. $p_1 \not$MUST A implies that there exists a p_1' such that $p_1 \stackrel{s}{\Longrightarrow} p_1'$

We define a suitable client. Let $s = \alpha_1\alpha_2 \ldots \alpha_n$, and let

$$A_i \stackrel{\texttt{def}}{=} \begin{cases} (1 \oplus 1) + \overline{\alpha}_{i+1}.A_{i+1} & \text{if } 0 \leq i < n \\ \mathsf{0} & \text{if } i = n \end{cases}$$

We prove that $p_2 \not$MUST A_0. The hypothesis imply that there exists a p_2' such that $p_2 \stackrel{s}{\Longrightarrow} p_2'$. If p_2' diverges we can infer the maximal computation

$$A_0 \parallel p_2 \Longrightarrow \mathsf{0} \parallel p_2' \stackrel{\tau}{\longrightarrow} \mathsf{0} \parallel p_2^1 \stackrel{\tau}{\longrightarrow} \mathsf{0} \parallel p_2^2 \stackrel{\tau}{\longrightarrow} \ldots$$

If p_2' does not diverge, then there exists a p_2'' such that $p_2' \stackrel{\varepsilon}{\Longrightarrow} p_2'' \stackrel{\tau}{\not\longrightarrow}$. We can infer the maximal computation

$$A_0 \parallel p_2 \Longrightarrow \mathsf{0} \parallel p_2'' \stackrel{\tau}{\not\longrightarrow}$$

The computation is not client-successful. In both cases we have shown non client-successful computation of $A_0 \parallel p_2$, so we have proven that $p_2 \not$MUST A_0. It follows that $p_1 \not$MUST A_0.

The hypothesis $p_1 \Downarrow s$, and $p_1 \not$MUST A_0 imply that $p_1 \stackrel{s}{\Longrightarrow} p_1'$ for some p_1', for otherwise all the maximal computations of $A_0 \parallel p_1$ would be client-successful. We explain this fact. Let $k < n$ be such that s_k is the longest prefix of s performed by p_1. Every maximal computation of $A_0 \parallel p_1$ must be finite, because $p \Downarrow s$ and the longest trace that A_0 performs is $\overline{s}$. Every maximal computation of $A_0 \parallel p_1$ must contain the contributions

$$A_0 \stackrel{\overline{s}_k}{\Longrightarrow} A_k, \qquad p_1 \stackrel{\overline{s}_k}{\Longrightarrow} p_1'$$

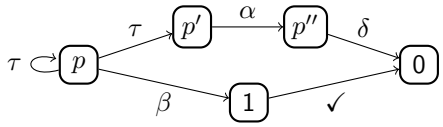


Figure 4.2: Process p of Example 4.1.12.

for otherwise the computations can be extended with one further interaction due an α. Since $k \leq i < n$, by construction we know that $A_k = (1 \oplus 1) + \overline{\alpha}_{k+1}.A_{k+1}$. As the computation at hand is maximal and $A_i \xrightarrow{\tau}$, the computation can be extended. Since $p'_1 \overset{\alpha_{k+1}}{\not\Longrightarrow}$, the only way to extend the computation is an internal move of A_k. The move reduces the client to a successful state, namely $1 + \overline{\alpha}_{k+1}.A_{k+1}$, thus the computation is client-successful.

We have shown that if $p_1 \overset{s}{\not\Longrightarrow}$ all the maximal computations of $A_0 \parallel p_1$ are client-successful. This contradicts $p \not\asymp_{\text{MUST}} A_0$, so $p_1 \overset{s}{\Longrightarrow} p'_1$ for some p'_1. □

The property of $\sqsubseteq_{\text{SVR}}$ that we have exhibited in Lemma 4.1.8 is not enough to characterise the pre-order.

Example 4.1.10. It is not true that for every p and q, if $p \Downarrow s$ then $q \Downarrow s$ imply that $p \sqsubseteq_{\text{CLT}} q$. For instance, for every $s \in Act^\star$, $\alpha.0 \Downarrow s$ and $\beta.0 \Downarrow s$. Nevertheless, one sees easily that $\alpha.0 \not\sqsubseteq_{\text{SVR}} \beta.0$; to prove this, let us use the client $\overline{\alpha}.1$. The server $\alpha.0$ satisfy $\overline{\alpha}.1$, because all the maximal computations of $\overline{\alpha}.1 \parallel \alpha.0$ are client-successful. On the contrary $\beta.0 \not\asymp_{\text{MUST}} \overline{\alpha}.1$, because $\overline{\alpha}.1 \parallel \beta.0 \overset{\tau}{\not\longrightarrow}$ and $\overline{\alpha}.1 \overset{\checkmark}{\not\longrightarrow}$. □

To characterise $\sqsubseteq_{\text{SVR}}$ we need some notation to compare the interactions offered by processes. In particular, we wish to compare the actions that a process can perform after a trace s.

Definition 4.1.11. [Acceptance set]
For every $s \in Act^\star$, and process p, we let

$$\text{ACC}(p, s) = \{\, S(p') \mid p \overset{s}{\Longrightarrow} p' \overset{\tau}{\not\longrightarrow} \,\}$$

where $S(p) = \{\, \alpha \in Act \mid p \xrightarrow{\alpha} \,\}$. □

Example 4.1.12. Consider the process p of Figure 4.2. We show the non-empty acceptance sets of p.

$$\begin{array}{lcllcl} \text{ACC}(p,\varepsilon) & = & \{\{\alpha\}\} & \text{ACC}(p,\alpha\delta) & = & \{\emptyset\} \\ \text{ACC}(p,\alpha) & = & \{\{\delta\}\} & \text{ACC}(p,\beta) & = & \{\emptyset\} \end{array}$$

Plainly, for all the string $s \in Act^\star$ not used above it holds $\text{ACC}(p, s) = \emptyset$; the reason being that there exists no p' such that $p \overset{s}{\Longrightarrow} p'$.

Note though that the ready set in $\text{ACC}(p, \beta)$ does not contain the action ✓. This is crucial for the characterisation to capture the equality $1 \eqsim_{\text{SVR}} 0$. □

Example 4.1.13. It is easy to show that whenever $\text{UNF}(\sigma) = 1$ then $S(\sigma) \subseteq A$ for every A. Consider the action ✓; it does not belong to Act, therefore by definition $\checkmark \notin S(\sigma)$. But ✓ is the *only* visible action performed by σ, hence $S(\sigma) = \emptyset$. □

Corollary 4.1.14. *For every $p_1, p_2 \in \mathsf{CCS}_{w\tau}$, and every $s \in Act^\star$, if $p_1 \sqsubseteq_{\text{SVR}} p_2$, $p_1 \Downarrow s$ and $\text{ACC}(p_2, s) \neq \emptyset$ then $\text{ACC}(p_1, s) \neq \emptyset$.*

Proof. The hypothesis $\text{ACC}(p_2, s) \neq \emptyset$ implies that $p_2 \stackrel{s}{\Longrightarrow}$. Since $p_1 \Downarrow s$ Lemma 4.1.9 implies that $p_1 \stackrel{s}{\Longrightarrow} p_1'$ for some p_1'. The hypothesis $p_1 \Downarrow s$ implies that $p_1' \Downarrow$, and so there exists a p_1'' such that $p_1 \stackrel{s}{\Longrightarrow} p_1'' \stackrel{\tau}{\not\longrightarrow}$. This implies that $S(p_1'') \in \text{ACC}(p_1, s)$. □

Lemma 4.1.15. Let $p_1 \sqsubseteq_{\text{SVR}} p_2$. For every $s \in Act^\star$, if $p_1 \Downarrow s$, then for every $B \in \text{ACC}(p_2, s)$ there exists a set $A \in \text{ACC}(p_1, s)$ such that $A \subseteq B$.

Proof. Fix a $s \in Act^\star$ such that $B \in \text{ACC}(p_2, s)$. It follows that $\text{ACC}(p_2, s) \neq \emptyset$, thus Corollary 4.1.14 implies that $\text{ACC}(p_1, s) \neq \emptyset$, so $\text{ACC}(p_1, s) = \{ A_i \mid i \in I \}$ for some non-empty set I.

The proof proceeds by contradiction; we suppose that

$$\text{for every } i \in I, \text{ the set } A_i \text{ contains an action } \alpha_i \notin B \tag{4.7}$$

We use this assumption to build a client C that distinguishes p_1 and p_2; that is

1. p_1 MUST C
2. $p_2 \not\text{MUST}\ C$

The string s is finite, so let $s = \beta_0\beta_1 \ldots \beta_n$ for some $n \in \mathbb{N}$.

Let

$$C_k \stackrel{\text{def}}{=} \begin{cases} (1 \oplus 1) + \overline{\beta_k}.C_{k+1} & \text{if } 0 \leq k < n \\ \sum_{i \in I} \overline{\alpha_i}.\,1 & \text{if } k = n \end{cases}$$

We depict the LTS of the processes C_k in Figure 4.3.

The client we are after is C_0. We prove that $p_2 \not\text{MUST}\ C_0$; that is, we exhibit a maximal computation of $p_2 \,||\, C_0$ that is not client-successful.

By construction, we can infer $C_0 \stackrel{\overline{s}}{\Longrightarrow} C_n$, and none of the states in the sequence is successful. Definition 4.1.11 and the hypothesis $B \in \text{ACC}(p_2, s)$ imply that there exists a p_2' such that $S(p_2') = B$ and $p_2 \stackrel{s}{\Longrightarrow} p_2'$. Either p_2' diverges or it converges. In the first case, observe the ensuing computation

$$C_0 \,||\, p_2 \Longrightarrow C_n \,||\, p_2' \stackrel{\tau}{\longrightarrow} C_n \,||\, p_2^1 \stackrel{\tau}{\longrightarrow} C_n \,||\, p_2^2 \stackrel{\tau}{\longrightarrow} \ldots$$

this computation is maximal and it is not client-successful. In the second case, we infer the computation

$$C_0 \,||\, p_2 \Longrightarrow C_n \,||\, p_2' \stackrel{\tau}{\not\longrightarrow}$$

where the fact that $p_2' \,||\, C_n$ is stable follows from three things: $p_2' \stackrel{\tau}{\not\longrightarrow}$, $C_n \stackrel{\tau}{\not\longrightarrow}$, and by construction no action performed by C_n can interact with the actions offered by p_2'. It follows that the computation above is maximal. What argued so far proves that $p_2 \not\text{MUST}\ C_0$.

We have to prove that p_1 MUST C_0. Definition 3.1.1 requires us to prove that all the maximal computations of $C_0 \,||\, p_1$ are client-successful. Fix a maximal computation of $p_1 \,||\, C_0$. The only traces that C_0 performs are the prefixes of $\overline{s}$; since $p \Downarrow s$, and no C_k is stable but C_n, every maximal computation of $C_0 \,||\, p$ must contain a state $C_n \,||\, p'$, where $p \stackrel{s}{\Longrightarrow} p'$ and $p' \stackrel{\tau}{\not\longrightarrow}$.

Definition 4.1.11 ensures that $S(p') \in \text{ACC}(p_1, s)$, so the assumption in (4.7) and the construction of C_n imply that there exists an $\alpha_i \in S(p')$ such that $C_n \stackrel{\overline{\alpha}_i}{\longrightarrow} 1$. It follows that p' and C_n can interact. Since the computations we are discussing are maximal, and the only reduction of $C_n \,||\, p'$ is due to an interaction, all the maximal computations of $C_0 \,||\, p$ contain a state $1 \,||\, p''$; this means that the computations are client-successful.

□

We need to prove one more property of $\sqsubseteq_{\text{SVR}}$. Its importance will become evident in Example 4.1.18.

Lemma 4.1.16. Let $p_1 \sqsubseteq_{\text{SVR}} p_2$. For every $u \in Act^\infty$, if $p_1 \Downarrow u$ and $p_2 \stackrel{u}{\Longrightarrow}$ then $p_1 \stackrel{u}{\Longrightarrow}$.

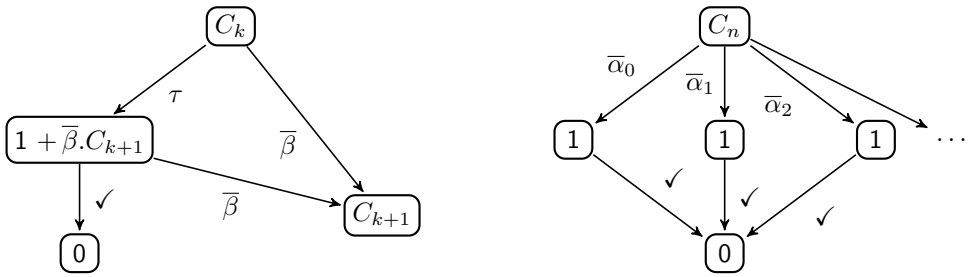


Figure 4.3: Tests to distinguish servers (see Lemma 4.1.15)

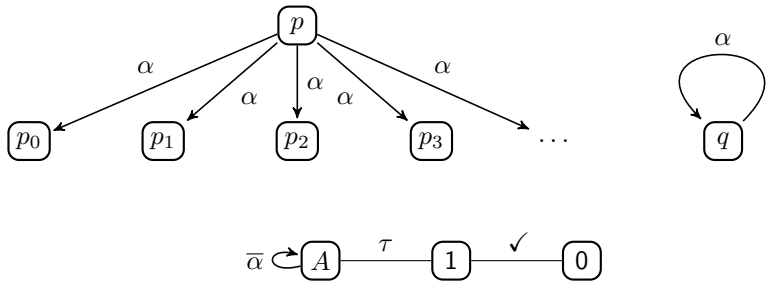


Figure 4.4: Infinite traces

Proof. Let $u = \alpha_1\alpha_2\alpha_3\ldots$, and for every $n \in \mathbb{N}$ let $A_n \stackrel{\texttt{def}}{=} (1 \oplus 1) + \overline{\alpha}_n.A_{n+1}$. No A_n is successful, so the infinite computation of $A_0 \,||\, p_2$ due to the interactions on the α's, proves that $p_2 \not\text{MUST}\ A_0$. The hypothesis $p_1 \sqsubseteq_{\text{SVR}} p_2$ implies that $p_1 \not\text{MUST}\ A_0$. If $p_1 \stackrel{u}{\not\Longrightarrow}$ then, thanks to the hypothesis $p \Downarrow u$, we can prove that all the maximal computations of $A_0 \,||\, p_1$ are client-successful; this contradicts $p_1 \not\text{MUST}\ A_0$, so it follows that $p_1 \stackrel{u}{\Longrightarrow}$. □

The next definition is a minor generalisation of the well-known behavioural characterisation of the must testing pre-order [De Nicola and Hennessy, 1984; Hennessy, 1985]. The only novelty is the condition on infinite traces.

Definition 4.1.17. [Semantic MUST server pre-order]
Let $p_1 \precsim_{\text{SVR}} p_2$ if

(1) for every $s \in Act^\star$ such that $p_1 \Downarrow s$,

  (a) $p_2 \Downarrow s$

  (b) for every $B \in \text{ACC}(p_2, s)$ there exists some $A \in \text{ACC}(p_1, s)$ such that $A \subseteq B$

(2) for every $w \in Act^\star \cup Act^\infty$ such that $p_1 \Downarrow w$, $p_2 \stackrel{w}{\Longrightarrow}$ implies $p_1 \stackrel{w}{\Longrightarrow}$. □

The condition on the existence of infinite computations in point (2) is already present for finite computations in point (1b). For suppose $p_1 \Downarrow s$. Then if $p_1 \stackrel{s}{\Longrightarrow}$ we know that $p_2 \stackrel{s}{\Longrightarrow}$. However in general, in particular in LTSs which are not finite branching, point (2) on infinite computations does not follow automatically from condition point (1b).

Example 4.1.18. [Infinite traces]
Consider the processes in Figure 4.4. There p_k denotes a process which performs a sequence of k α actions and then becomes $\mathsf{0}$; so the process p performs every finite sequence of α's. On the contrary, the process q performs an infinite sequence of α. Then $p \Downarrow s$ and $q \Downarrow s$ for every s, and the pair (p, q)

satisfies condition (1b) of $\precsim_{\text{SVR}}$. However condition (2) is not satisfied; let u denotes the infinite sequence of αs. On the one hand the self loop of q let us prove that $q \stackrel{u}{\Longrightarrow}$. On the other hand p does not perform the infinite trace u, that is $p \stackrel{u}{\not\Longrightarrow}$.

In fact $p \not\precsim_{\text{SVR}} q$. For consider the process $A \stackrel{\text{def}}{=} (1 \oplus 1) + \overline{\alpha}.A$. When A is run as test on p, or as a *client* using the *server* p, every computation is finite and successful; p MUST A. However when q is run as a server, there is the possibility of an infinite computation, the indefinite synchronisation on α, which is not successful; $q \not\text{MUST}\ A$. □

The previous example can be adapted to prove that the co-inductive technique used in [Laneve and Padovani, 2007], which is sound in the setting used there, is not sound with respect to the relation $\sqsubseteq_{\text{SVR}}$ in our setting.

Example 4.1.19. [Standard co-inductive characterisation not sound]
Let $\preccurlyeq_{\mathsf{bad}}$ be the greatest relation such that $p_1 \preccurlyeq_{\mathsf{bad}} p_2$ if and only if

1. if $p_2 \xrightarrow{\tau} p_2'$ then $p_1 \preccurlyeq_{\mathsf{bad}} p_2'$
2. if $B \in \text{ACC}(p_2, \varepsilon)$ then there exists a $A \in \text{ACC}(p_1, \varepsilon)$ such that $A \subseteq B$
3. if $p_2 \xrightarrow{\alpha} p_2'$ then $p_1 \stackrel{\alpha}{\Longrightarrow}$ and $\bigoplus(p_1 \text{ AFTER } \alpha) \preccurlyeq_{\mathsf{bad}} p_2'$

In this example we prove that the relation $\preccurlyeq_{\mathsf{bad}} \not\subseteq \sqsubseteq_{\text{SVR}}$. To this end, we show two processes p and q such that $p \preccurlyeq_{\mathsf{bad}} q$ and $p \not\precsim_{\text{SVR}} q$. Consider the processes p and q that we used in Example 4.1.18. As we have already argued that $p \not\precsim_{\text{SVR}} q$, it is enough to show that $p \preccurlyeq_{\mathsf{bad}} q$. Let α^n denote the string of n α's, and let p^n denote the process that performs n actions α and then becomes $\mathsf{0}$, with $p^0 = \mathsf{0}$. One can prove the that for every $n > 1$ following equality is true,

$$\bigoplus(p \text{ AFTER } \alpha^n) = \bigoplus_{i=0} p^i$$

Consider now the relation $\mathcal{R} = \{\, (p,\, q), (\bigoplus(p \text{ AFTER } \alpha), q) \,\}$. Thanks to the equality above one can prove that $\mathcal{R} \subseteq \preccurlyeq_{\mathsf{bad}}$, so $p \preccurlyeq_{\mathsf{bad}} q$. □

The relation $\precsim_{\text{SVR}}$ is complete with respect to $\sqsubseteq_{\text{SVR}}$.

Proposition 4.1.20. [Completeness]
If $p_1 \sqsubseteq_{\text{SVR}} p_2$ then $p_1 \precsim_{\text{SVR}} p_2$.

Proof. The proposition follows from Lemma 4.1.8, Lemma 4.1.15 and Lemma 4.1.16. □

Theorem 4.1.21. [Alternative characterisation $\sqsubseteq_{\text{SVR}}$]
For every $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, $p_1 \sqsubseteq_{\text{SVR}} p_2$ if and only if $p_1 \precsim_{\text{SVR}} p_2$.

Proof. For every two processes p_1 and p_2, we are required to prove two implications:

i) if $p_1 \sqsubseteq_{\text{SVR}} p_2$ if and only if $p_1 \precsim_{\text{SVR}} p_2$,

ii) if $p_1 \precsim_{\text{SVR}} p_2$ then $p_1 \sqsubseteq_{\text{SVR}} p_2$

The implication (i) is proven in Proposition 4.1.20, so we prove only why the implication in (ii) is true.

Let $p_1 \precsim_{\text{SVR}} p_2$, and suppose that r_1 MUST r. We have to explain why p_2 MUST r. Definition 3.1.1 requires us to show that all the maximal computations of $r \,||\, p_2$ are client-successful.

Fix a maximal computation of $r \,||\, p_2$,

$$r \,||\, p_2 = r^0 \,||\, p_2^0 \xrightarrow{\tau} r^1 \,||\, p_2^1 \xrightarrow{\tau} r^3 \,||\, p_2^3 \xrightarrow{\tau} r^3 \,||\, p_2^3 \xrightarrow{\tau} \ldots \tag{4.8}$$

The computation in (4.8) is finite or infinite. We discuss the two cases separately.

Suppose that the computation in (4.8) is finite. Let us unzip the maximal computation at hand; since it is finite, we get two finite contributions

$$r \stackrel{s}{\Longrightarrow} r', \qquad p_2 \stackrel{\overline{s}}{\Longrightarrow} p_2'$$

and the state $r' \parallel p_2'$ is stable. If $p_1 \not\Downarrow \overline{s}$, then p_1 reaches a state p_1' by performing a prefix of $\overline{s}$ and p_1' diverges. By zipping $r \stackrel{s}{\Longrightarrow} r'$ with the action sequence of p_1' we obtain a maximal computation of $r \parallel p_1$; since p_1 MUST r, the new computation is client-successful, and so is the one unzipped.

Suppose now that $p_1 \Downarrow s$. The definition of Definition 4.1.11 implies that $S(p_2') \in \text{ACC}(p_2, \overline{s})$, and so point (1b) of Definition 4.1.17 implies that there exists a set $A \in \text{ACC}(p_1, \overline{s})$ such that $A \subseteq S(p_2')$. The assumption that $p_1 \Downarrow s$ and Definition 4.1.11 ensures that there exists a p_1' such that $p_1 \stackrel{\overline{s}}{\Longrightarrow} p_1' \stackrel{\tau}{\not\longrightarrow}$ such that $S(p_1') \subseteq S(p_2')$. It follows that there exists the maximal computation

$$r \parallel p_1 \Longrightarrow r' \parallel p_1' \stackrel{\tau}{\not\longrightarrow}$$

Since p_1 MUST r, the finite computation above is client-successful, and so also the computation in (4.8) is client-successful.

We have proven that if the maximal computation in (4.8) is finite, then it is client-successful. Now we prove that if that computation is infinite then it is client-successful.

Suppose that he computation in (4.8) is infinite. This may be true because the processes engage in infinite traces, or because (at least) one of the two diverges.

Unzip the computation in (4.8) and suppose that obtain infinite contributions

$$r \stackrel{u}{\Longrightarrow}, \qquad p_2 \stackrel{\overline{u}}{\Longrightarrow}$$

We have to prove that one of the states reached by r is successful. Point (2) of Definition 4.1.17 implies that $p_2 \stackrel{\overline{u}}{\Longrightarrow}$. By zipping together the infinite contribution of r and the infinite contribution of p_1 we obtain a maximal computation of $r \parallel p_1$. The assumption that p_1 MUST r ensures that the new computation is client-successful. Since the derivatives of r in this computation are the same that appear in (4.8), we have proven that the computation we unzipped is client-successful.

Now we discuss the case of divergence. Unzip the computation in (4.8) and suppose that obtain finite contributions

$$r \stackrel{s}{\Longrightarrow} r', \qquad p_2 \stackrel{\overline{s}}{\Longrightarrow} p_2'$$

Since the computation is infinite r' diverges or p_2' diverges (or both diverge).

Suppose that p_2' diverges. This assumption implies that $p_2 \not\Downarrow \overline{s}$. Point (1a) of Definition 4.1.17 ensures that $p_1 \not\Downarrow \overline{s}$; in turn this means that there exists a prefix s' of s such that $p_1 \stackrel{\overline{s'}}{\Longrightarrow} p_1'$ and p_1' diverges. By zipping $r \stackrel{s'}{\Longrightarrow} r'$ with $p_1 \stackrel{\overline{s'}}{\Longrightarrow} p_1'$ we obtain an infinite maximal computation of $r \parallel p_1$. The assumption p_1 MUST r implies that r reaches a successful state, so the computation we unzipped is client-successful.

Suppose that r' diverges. Either $p_1 \not\Downarrow \overline{s}$ or $p_1 \Downarrow \overline{s}$. If $p_1 \not\Downarrow \overline{s}$ then we reason as explain above. If $p_1 \Downarrow \overline{s}$, then point (2) of Definition 4.1.17 implies $p_1 \stackrel{\overline{s}}{\Longrightarrow} p_1'$. By zipping this action sequence of p_1 with $r \stackrel{s}{\Longrightarrow} r'$ we obtain an infinite maximal computation of $r \parallel p_1$. The assumption that p_1 MUST r implies that r reaches a successful state, so the computation we unzipped is client-successful.

□

Theorem 4.1.21 gives a proof method for the relation $\sqsubseteq_{\text{SVR}}$. We discuss this fact in the next example.

Example 4.1.22. [Proof method] In this example we wish a) to use Theorem 4.1.21 to prove that

two processes are related by $\sqsubseteq_{\text{SVR}}$, and b) to compare the proof method with the one given by the definition of $\sqsubseteq_{\text{SVR}}$.

We prove $p_1 \sqsubseteq_{\text{SVR}} p_2$, where $p_1 = \alpha.\,\mathsf{0} \oplus \beta.\,\mathsf{0}$ and $p_2 = \alpha.\,\mathsf{0} + \beta.\,\mathsf{0}$. Thanks to Theorem 4.1.21, $p_1 \sqsubseteq_{\text{SVR}} p_2$ will follow if we show that $p_1 \precsim_{\text{SVR}} p_2$. Definition 4.1.17 requires us to show the following properties,

(1) for every $s \in Act^\star$ such that $p_1 \Downarrow s$,

   (a) $p_2 \Downarrow s$

   (b) for every $B \in \text{ACC}(p_2, s)$ there exists some $A \in \text{ACC}(p_1, s)$ such that $A \subseteq B$

(2) for every $w \in Act^\star \cup Act^\infty$ such that $p_1 \Downarrow w$, $p_2 \stackrel{w}{\Longrightarrow}$ implies $p_1 \stackrel{w}{\Longrightarrow}$.

Since each state reached by p_1 or p_2 converges, (1a) above is true. Now observe that the traces performed by p_1 and p_2 are the same, namely ε, α, β. This ensures that (2) above is true.

To prove condition (1b), let us compare the non-empty acceptance sets of p_2 with the acceptance sets of p_1,

$$\text{ACC}(p_2, \varepsilon) = \{\{\alpha, \beta\}\} \quad \text{ACC}(p_1, \varepsilon) = \{\{\alpha\}, \{\beta\}\}$$
$$\text{ACC}(p_2, \alpha) = \{\emptyset\} \qquad \text{ACC}(p_1, \alpha) = \{\emptyset\}$$
$$\text{ACC}(p_2, \beta) = \{\emptyset\} \qquad \text{ACC}(p_1, \beta) = \{\emptyset\}$$

For each ready set in the non-empty acceptance sets of p_2, there exists a ready set in the acceptance sets of p_1, so also (1b) above is satisfied; this concludes the proof that $p_1 \precsim_{\text{SVR}} p_2$.

Now let us sketch a proof of $p_1 \sqsubseteq_{\text{SVR}} p_2$ that relies on the definition of $\sqsubseteq_{\text{SVR}}$. We are required to show that that *for every* r if p MUST r, then q MUST r. In this example, thanks to the very simple behaviour of p_1, it is possible to prove the implication above. In general, though, it is not clear that the implication can be proven, because of the universal quantification on all the rs. □

Theorem 4.1.21 is essentially the characterisation of the MUST pre-order of [Hennessy, 1985]. Our server pre-order $\sqsubseteq_{\text{SVR}}$, though, is bigger than the MUST pre-order; this is because $\sqsubseteq_{\text{SVR}}$ relates also terms that contains $\mathsf{1}$ (i.e. processes whose LTS performs $\checkmark$), whereas $\sqsubseteq$ relates only terms that cannot perform $\checkmark$. The following inclusion is true, $\sqsubseteq \subseteq \sqsubseteq_{\text{SVR}}$.

The next result will be crucial in Section 4.3.

Lemma 4.1.23. For every process p, $p \eqsim_{\text{SVR}} p + \sum_{i \in I} \mathsf{1}$ for every set I.

Proof. If I is empty then the lemma is trivially true, so suppose that $I \neq \emptyset$. Thanks to Theorem 4.1.21, it is enough to prove that

i) $p \precsim_{\text{SVR}} p + \sum_{i \in I} \mathsf{1}$

ii) $p + \sum_{i \in I} \mathsf{1} \precsim_{\text{SVR}} p$

For every $s \in Act^\star$, $p \Downarrow s$ if and only if $p + \sum_{i \in I} \mathsf{1} \Downarrow s$; and $\text{ACC}(p, s) = \text{ACC}(p + \sum_{i \in I} \mathsf{1}, s)$. Moreover for every $u \in Act^\infty$, $p \stackrel{u}{\Longrightarrow}$ if and only if $p + \sum_{i \in I} \mathsf{1} \stackrel{u}{\Longrightarrow}$. In view of these facts, one can prove both i) and ii). □

In this section we have defined and characterised the server pre-order $\sqsubseteq_{\text{SVR}}$. We have seen that a server p_2 is better than another server p_1, if p_2 offers at least the interactions that p_1 offers, converges as p_1 does, and up-to convergence, p_1 performs the traces that p_2 performs. To lay bare the way whereby $\sqsubseteq_{\text{SVR}}$ relates the behaviours of processes, we have had to introduce the notions to reason about convergence, ready sets, and acceptance sets. These ideas will turn out to be paramount, as their variations will let us characterise many more pre-orders. The next pre-order that we study is the client pre-order.

4.2 Client pre-order

In the previous section we have studied when a server p_2 is better than a server p_1, in the sense that p_2 satisfies more clients than p_1 with respect to the MUST relation. In this section we take the dual stance; we study when a client process r_2 is satisfied by more servers than a client r_1. To this end we introduce a pre-order for clients (Definition 4.2.1), and show examples of refinements take can (not) take place. The examples show that the client pre-order differs from the server pre-order (4.9), thus we have to devise an alternative characterisation for the client pre-order. Throughout this section we put forth the notions that we need to give this alternative characterisation (Theorem 4.2.37).

Definition 4.2.1. [MUST-client pre-order]
We write $r_1 \sqsubseteq_{\text{CLT}} r_2$ if and only if p MUST r_1 implies p MUST r_2 for every process p. We refer to the relation $\sqsubseteq_{\text{CLT}}$ as MUST *client pre-order* . □

Notation Similarly what done for $\sqsubseteq_{\text{SVR}}$, also to reason on $\sqsubseteq_{\text{CLT}}$ we are free to use the general summations $\sum$ and $\bigoplus$. This is justified by the fact that $\eqsim_{\text{CLT}}$ is commutative and associative with respect to $\oplus$ and $+$, where $\eqsim_{\text{CLT}}$ is the equivalence relation given in the obvious way by $\sqsubseteq_{\text{SVR}}$.

The relation $\sqsubseteq_{\text{CLT}}$ is indeed a pre-order, as the definition implies immediately that $\sqsubseteq_{\text{CLT}}$ is reflexive and transitive. In view of Lemma 3.1.6, the process $\mathsf{1}$ is a top element of $\sqsubseteq_{\text{CLT}}$, while the terms $\mathsf{0}$ and τ^∞ are bottom elements of $\sqsubseteq_{\text{CLT}}$. In our study the elements of $\sqsubseteq_{\text{CLT}}$ which are not bottom will play a crucial role.

Definition 4.2.2. [Usable client]
Let

$$\mathcal{U}_{\text{CLT}}^{\text{MUST}} = \{\, r \mid p \text{ MUST } r, \text{ for some server } p \,\}$$

If $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ we say that r is a *usable* client. □

Intuitively, a client is usable if there is at least one server which satisfies it. It is straightforward to see that every usable client is not a bottom element of $\sqsubseteq_{\text{CLT}}$: if $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ then $r \not\sqsubseteq_{\text{CLT}} \mathsf{0}$.

Intuitively, if $r_1 \sqsubseteq_{\text{CLT}} r_2$, then relation $\sqsubseteq_{\text{CLT}}$ ensures that the interactions *required* by r_2 to reach a successful state are fewer then the interaction *required* by r_1.

Example 4.2.3. In this example we discuss some pairs of clients (not) related by the client pre-order. Let r_1 denote the process depicted in Figure 4.6.

$$\begin{array}{lll} \alpha.\mathsf{0} & \sqsubseteq_{\text{CLT}} & \mathsf{0} \\ \alpha.\mathsf{1} & \sqsubseteq_{\text{CLT}} & \mathsf{1} \oplus \mathsf{1} \\ \mathsf{1} & \not\sqsubseteq_{\text{CLT}} & \mathsf{0} \\ r_1 & \sqsubseteq_{\text{CLT}} & \mathsf{0} \end{array}$$

The inequality $\alpha.\mathsf{0} \sqsubseteq_{\text{CLT}} \mathsf{0}$ is true because neither client ever report success, so they are not satisfied by any server. In fact, $\alpha.\mathsf{0} \eqsim_{\text{CLT}} \mathsf{0}$.

To prove the second inequality let p MUST $\alpha.\mathsf{1}$; we have to show that p MUST $\mathsf{1} \oplus \mathsf{1}$. Intuitively, the assumption p MUST $\alpha.\mathsf{1}$ implies that $p \Downarrow$, and this lets us prove that in all the maximal computations of $p \parallel \mathsf{1} \oplus \mathsf{1}$ the client reduces to $\mathsf{1}$ (i.e. reaches a successful state).

The third inequality follows from p MUST $\mathsf{1}$ and $p \not\text{MUST}\ \mathsf{0}$ for every p; this being true because $\mathsf{1}$ reports success immediately, whereas $\mathsf{0}$ can never report $\checkmark$.

The intuition behind the fourth inequality is that all the trace that lead r_1 to a successful state, can be performed by r_1 also never reaching a successful state and eventually getting stuck. □

Example 4.2.4. Here we discuss how sensitive $\sqsubseteq_{\mathrm{CLT}}$ is to divergence.

$$\begin{array}{rcl} 1 & \not\sqsubseteq_{\mathrm{CLT}} & 1 \oplus 1 \\ 1 & \sqsubseteq_{\mathrm{CLT}} & 1 + \tau^\infty \\ 1 & \not\sqsubseteq_{\mathrm{CLT}} & 1 \oplus \tau^\infty \\ \alpha.1 & \sqsubseteq_{\mathrm{CLT}} & (\alpha.1) \oplus ((1+\tau^\infty) \oplus (1+\tau^\infty)) \\ (1+\tau^\infty) \oplus (1+\tau^\infty) & \not\sqsubseteq_{\mathrm{CLT}} & \tau^\infty \end{array}$$

The divergence on the server side lets us explain the inequality $1 \not\sqsubseteq_{\mathrm{CLT}} 1 \oplus 1$. The process 1 reports success immediately, whereas $1 \oplus 1$ requires one reduction to reach a successful state. A server that does not allow the clients to reduce distinguishes the two processes.For instance

$$\tau^\infty \text{ MUST } 1, \quad \tau^\infty \not\text{MUST } 1 \oplus 1$$

The latter fact follows from $1 \oplus 1 \stackrel{\checkmark}{\not\longrightarrow}$ and the maximal computation

$$\tau^\infty \,||\, 1 \oplus 1 \stackrel{\tau}{\longrightarrow} \tau^\infty \,||\, 1 \oplus 1 \stackrel{\tau}{\longrightarrow} \ldots$$

To prove that $1 \not\sqsubseteq_{\mathrm{CLT}} 1 \oplus 1$ it is necessary to use a divergent server, and in Example 4.2.26 we will see that under the assumption of convergence one can prove that $1 \sqsubseteq_{\mathrm{CLT}} 1 \oplus 1$.

The inequality $1 \sqsubseteq_{\mathrm{CLT}} 1 + \tau^\infty$ is true because both processes perform $\checkmark$, so for every process p one can prove that p MUST 1 and p MUST $1 + \tau^\infty$. This shows that a client is free to diverge after it has reached a successful state.

The inequality $1 \not\sqsubseteq_{\mathrm{CLT}} 1 \oplus \tau^\infty$ is true because the client $1 \oplus \tau^\infty$ can diverge having reached no successful state.

We discuss the fourth inequality, $\alpha.1 \sqsubseteq_{\mathrm{CLT}} r$, where $r = (\alpha.1) \oplus ((1+\tau^\infty) \oplus (1+\tau^\infty))$. The reason for the inequality to be true is that if p MUST $\alpha.1$ then p converges; we prove this fact. Suppose that p MUST $\alpha.1$; this implies that p does not diverge, for otherwise there would exist a non client-successful computation of $p \,||\, \alpha.1$, namely

$$\alpha.1 \,||\, p \stackrel{\tau}{\longrightarrow} \alpha.1 \,||\, p \stackrel{\tau}{\longrightarrow} \alpha.1 \,||\, p \stackrel{\tau}{\longrightarrow} \alpha.1 \,||\, p \stackrel{\tau}{\longrightarrow} \ldots$$

We have proven that if p MUST $\alpha.1$ then p converges. Now we prove that if p MUST $\alpha.1$ then p MUST r. Suppose that p MUST $\alpha.1$; we show that all the maximal computations of $r \,||\, p$ are client-successful. The state r is defined by a top-most internal choice, so it does not perform any visible action. It follows that in any maximal computation of $r \,||\, p$, the process p is bound to reach a stable state p'; this being true because r does not communicate and p converges. After p has reached a stable state, r reduces to (a) $\alpha.1$ or to (b) $(1+\tau^\infty) \oplus (1+\tau^\infty)$, and this term reduces further to the successful state $1+\tau^\infty$. If (a), then p' MUST $\alpha.1$ follows from the assumption on p MUST $\alpha.1$, so the maximal computation at hand must be client-successful; if (b) then the maximal computation at hand is client-successful as $1 + \tau^\infty \stackrel{\checkmark}{\longrightarrow}$. We have shown that p MUST r.

The last inequality shows that unsuccessful divergence is not equivalent to successful divergence; for instance 0 MUST $(1+\tau^\infty) \oplus (1+\tau^\infty)$, and 0 $\not$MUST τ^∞. This lets us insist on the fact that it does not matter whether a client diverges or not; the important aspect is whether the divergent computations of a client reach a successful state or not. □

In view of the Example 4.1.3, and Example 4.2.3, one can show that $\sqsubseteq_{\mathrm{SVR}}$ is not comparable with $\sqsubseteq_{\mathrm{CLT}}$; moreover divergence is not necessary to prove this;

$$\sqsubseteq_{\mathrm{SVR}} \not\subseteq \sqsubseteq_{\mathrm{CLT}}, \qquad \sqsubseteq_{\mathrm{CLT}} \not\subseteq \sqsubseteq_{\mathrm{SVR}} \tag{4.9}$$

$$\frac{}{p \overset{\varepsilon}{\Longrightarrow}_{\not\checkmark} p} \quad p \overset{\checkmark}{\not\longrightarrow};\ [\text{UT-AX}]$$

$$\frac{p' \overset{s}{\Longrightarrow}_{\not\checkmark} q}{p \overset{s}{\Longrightarrow}_{\not\checkmark} q} \quad p \xrightarrow{\tau} p', p \overset{\checkmark}{\not\longrightarrow};\ [\text{UT-TAU}]$$

$$\frac{p' \overset{s}{\Longrightarrow}_{\not\checkmark} q}{p \overset{\alpha s}{\Longrightarrow}_{\not\checkmark} q} \quad p \xrightarrow{\alpha} p', p \overset{\checkmark}{\not\longrightarrow};\ [\text{UT-ALPHA}]$$

Figure 4.5: Inference rules for the functional $\mathcal{F}_{\Longrightarrow_{\not\checkmark}}$

To see why the negative inclusions (4.9) are true observe that

$$\begin{array}{rcl} \alpha.1 & \sqsubseteq_{\text{SVR}} & \alpha.0 \\ \alpha.1 + \alpha.0 & \not\sqsubseteq_{\text{SVR}} & 0 \end{array} \qquad \begin{array}{rcl} \alpha.1 & \not\sqsubseteq_{\text{CLT}} & \alpha.0 \\ \alpha.1 + \alpha.0 & \sqsubseteq_{\text{CLT}} & 0 \end{array}$$

It follows the characterisation of $\sqsubseteq_{\text{SVR}}$ given in Theorem 4.1.21 does not capture $\sqsubseteq_{\text{CLT}}$.

This section is devoted to the formulation of a behavioural characterisation of $\sqsubseteq_{\text{CLT}}$. To characterise the server pre-order we used the notions of convergence, trace and acceptance set. We follow a similar approach to characterise the client pre-order; throughout this section we show that, as they stand, those notions do not let us characterise $\sqsubseteq_{\text{CLT}}$, and we amend them so as to obtain a characterisation of $\sqsubseteq_{\text{CLT}}$.

Example 4.2.5. One can prove that $\beta.\alpha.1 \sqsubseteq_{\text{CLT}} r$, where r denotes $\beta.(\gamma.0 + 1)$. However the acceptance sets of those clients are not related as required by Theorem 4.1.21. For example $\{\gamma\} \in \text{ACC}(r, \beta)$ but there is no $B \in \text{ACC}(\beta.\alpha.1, \beta)$ satisfying $B \subseteq \{\gamma\}$. This follows since $\text{ACC}(\beta.\alpha.1, \beta)$ contains only the set $\{\alpha\}$. □

The characterisation of the server pre-order $p_1 \sqsubseteq_{\text{SVR}} p_2$ in Theorem 4.1.21 demands that every interaction offered by p_2 after performing a trace s, represented by a ready set in $\text{ACC}(p_2, s)$, be matched appropriately by a set of interactions of p_1 after performing s. However, the reasons for $r_1 \sqsubseteq_{\text{CLT}} r_2$ being true are different. We only require possible deadlocks in r_2 to be matched by r_1 so long as r_2 has not reported a success. So in Example 4.2.5 we should not require the ready set $\{\gamma\} \in \text{ACC}(r_2, \beta)$ to be matched by one in $\text{ACC}(\beta.\alpha.1, \beta)$ because r_2 can report success immediately after performing β.

To formalise this intuition we need some notation for ready sets of deadlocks[1] after *unsuccessful* sequences of actions.

Definition 4.2.6. [Unsuccesful traces]
Let $\mathcal{F}_{\Longrightarrow_{\not\checkmark}} : \mathcal{P}(\mathsf{CCS}_{\text{w}\tau} \times Act^\star \times \mathsf{CCS}_{\text{w}\tau}) \longrightarrow \mathcal{P}(\mathsf{CCS}_{\text{w}\tau} \times Act^\star \times \mathsf{CCS}_{\text{w}\tau})$ be the rule functional given by the inference rules in Figure 4.5. Lemma C.0.22 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_{\Longrightarrow_{\not\checkmark}}(X)$; we call this solution the *unsuccesful traces*, and we denote it $\Longrightarrow_{\not\checkmark}$: That is $\Longrightarrow_{\not\checkmark} = \mu X.\mathcal{F}_{\Longrightarrow_{\not\checkmark}}(X)$. The predicate $\Longrightarrow_{\not\checkmark}$ is extended to $u \in Act^\infty$ by letting $r \overset{u}{\Longrightarrow}_{\not\checkmark}$ if and only if there exists a $t \in \mathsf{Act}_\tau^\infty$ such that

- if $t = \alpha_\tau^1 \alpha_\tau^2 \alpha_\tau^3 \ldots$, then $r = r_0 \xrightarrow{\alpha_\tau^1} r_1 \xrightarrow{\alpha_\tau^2} r_2 \xrightarrow{\alpha_\tau^3} \ldots$, and for every $n \in \mathbb{N}$, $r_n \overset{\checkmark}{\not\longrightarrow}$;
- for every $n \in \mathbb{N}$ $u_n = \langle t_k \rangle_{\setminus\tau}$ for some $k \in \mathbb{N}$ □

Intuitively, we would like $p \overset{s}{\Longrightarrow}_{\not\checkmark} q$ to mean that p performs the sequence of external actions s ending up in state q without passing through any state which can report success; in particular that

[1] In this context we deem a state "deadlock" if it is stable.

neither p nor q can report success. As the relation $\Longrightarrow_{\not\checkmark}$ is not defined directly in terms of $\longrightarrow$, but it is a fixed point of a rule functional, a proof is in order.

Proposition 4.2.7. [Operational meaning of $\Longrightarrow_{\not\checkmark}$]
For every $p, q \in \mathsf{CCS}_{\mathrm{w}\tau}$, $p \stackrel{s}{\Longrightarrow}_{\not\checkmark} p_n$ if and only if $s = \langle \alpha_\tau^1 \alpha_\tau^2 \dots \alpha_\tau^n \rangle_{\setminus\tau}$, $p \stackrel{\alpha_\tau^1}{\longrightarrow} p_1 \stackrel{\alpha_\tau^2}{\longrightarrow} \dots \stackrel{\alpha_\tau^n}{\longrightarrow} p_n$ and for every $0 \le i \le n$, $p_i \stackrel{\checkmark}{\not\longrightarrow}$.

Proof. We have to prove two implications, namely

(i) if $p \stackrel{\langle \alpha_\tau^1 \alpha_\tau^2 \dots \alpha_\tau^n \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} p_n$ then $p \stackrel{\alpha_\tau^1}{\longrightarrow} p_1 \stackrel{\alpha_\tau^2}{\longrightarrow} \dots \stackrel{\alpha_\tau^n}{\longrightarrow} p_n$ and for every $0 \le i \le n$, $p_i \stackrel{\checkmark}{\not\longrightarrow}$;

(ii) if $p \stackrel{\alpha_\tau^1}{\longrightarrow} p_1 \stackrel{\alpha_\tau^2}{\longrightarrow} \dots \stackrel{\alpha_\tau^n}{\longrightarrow} p_n$ and for every $0 \le i \le n$, $p_i \stackrel{\checkmark}{\not\longrightarrow}$, then $p \stackrel{s}{\Longrightarrow}_{\not\checkmark} q$.

We prove the implication (i). The argument is by induction on the derivation of $p \stackrel{\langle \alpha_\tau^1 \alpha_\tau^2 \dots \alpha_\tau^n \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} q$.

Base case In this case, the last rule applied in the derivation is the axiom [UT-AX], that is

$$\frac{}{p \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark} p}\ p \stackrel{\checkmark}{\not\longrightarrow};\ [\text{UT-AX}]$$

so $p_n = p$. By letting $n = 0$ (i.e. $\alpha_\tau^1 \alpha_\tau^2 \dots \alpha_\tau^n = \varepsilon$), we see that $p \stackrel{\checkmark}{\not\longrightarrow}$.

Inductive case The last rule used to derive $p \stackrel{s}{\Longrightarrow}_{\not\checkmark} q$ is [UT-TAU] or [UT-ALPHA]. In the first case the derivation has form

$$\frac{\begin{array}{c}\vdots\\ p' \stackrel{s}{\Longrightarrow}_{\not\checkmark} p_n\end{array}}{p \stackrel{s}{\Longrightarrow}_{\not\checkmark} p_n}\ p \stackrel{\checkmark}{\not\longrightarrow}, p \stackrel{\tau}{\longrightarrow} p';\ [\text{UT-TAU}]$$

Since the derivation of $p' \stackrel{s}{\Longrightarrow}_{\not\checkmark} q$ is shorter than the derivation of $p \stackrel{s}{\Longrightarrow}_{\not\checkmark} q$, the inductive hypothesis ensures that $s = \langle \alpha_\tau^1 \alpha_\tau^2 \dots \alpha_\tau^n \rangle_{\setminus\tau}$, $p' = p_0 \stackrel{\alpha_\tau^1}{\longrightarrow} p_1 \stackrel{\alpha_\tau^2}{\longrightarrow} \dots \stackrel{\alpha_\tau^n}{\longrightarrow} p_n$ and for every $0 \le i \le n$, $p_i \stackrel{\checkmark}{\not\longrightarrow}$. By letting $p_0 = p$ and $p' = p_1$ we obtain the action sequence

$$p_0 \stackrel{\tau}{\longrightarrow} p_1 \stackrel{\alpha_\tau^1}{\longrightarrow} p_2 \stackrel{\alpha_\tau^2}{\longrightarrow} \dots \stackrel{\alpha_\tau^n}{\longrightarrow} p_n$$

and since $p \stackrel{\checkmark}{\not\longrightarrow}$, for every $0 \le i \le n$, $p_i \stackrel{\checkmark}{\not\longrightarrow}$.

If rule [UT-ALPHA] was applied then the derivation of $p \stackrel{s}{\Longrightarrow}_{\not\checkmark} q$ has form

$$\frac{\begin{array}{c}\vdots\\ p' \stackrel{s'}{\Longrightarrow}_{\not\checkmark} q\end{array}}{p \stackrel{\alpha s'}{\Longrightarrow}_{\not\checkmark} q}\ p \stackrel{\checkmark}{\not\longrightarrow}, p \stackrel{\alpha}{\longrightarrow} p' [\text{UT-ALPHA}]$$

It follows that $s = \alpha s'$. Since the derivation of $p' \stackrel{s'}{\Longrightarrow}_{\not\checkmark} q$ is shorter than the derivation of $p \stackrel{\alpha s'}{\Longrightarrow}_{\not\checkmark} q$, the inductive hypothesis ensures that $s = \langle \alpha_\tau^1 \alpha_\tau^2 \dots \alpha_\tau^n \rangle_{\setminus\tau}$, $p' = p_0 \stackrel{\alpha_\tau^1}{\longrightarrow} p_1 \stackrel{\alpha_\tau^2}{\longrightarrow} \dots \stackrel{\alpha_\tau^n}{\longrightarrow} p_n$ and for every $0 \le i \le n$, $p_i \stackrel{\checkmark}{\not\longrightarrow}$. By letting $p_0 = p$, $p' = p_1$, $\beta_1 = \alpha$ and for $i > 1$, $\beta_i = \alpha_\tau^{i-1}$ we obtain the action sequence

$$p_0 \stackrel{\beta_1}{\longrightarrow} p_1 \stackrel{\beta_2}{\longrightarrow} p_2 \stackrel{\beta_3}{\longrightarrow} \dots \stackrel{\beta_{n+1}}{\longrightarrow} p_n$$

and since $p_0 \stackrel{\checkmark}{\not\longrightarrow}$, for every $0 \le i \le n$, $p_i \stackrel{\checkmark}{\not\longrightarrow}$.

We have proven implication (i); now we prove implication (ii). Suppose that

a) $p \stackrel{\alpha_\tau^1}{\longrightarrow} p_1 \stackrel{\alpha_\tau^2}{\longrightarrow} \dots \stackrel{\alpha_\tau^n}{\longrightarrow} p_n$, and

b) for every $0 \leq i \leq n$, $p_i \overset{\checkmark}{\not\longrightarrow}$.

Let $t = \alpha_\tau^1 \alpha_\tau^2 \dots \alpha_\tau^n$, and $s = \langle r \rangle_{\setminus\tau}$. We have to exhibit a finite derivation of $p \overset{s}{\Longrightarrow}_{\not\checkmark} q$.

The argument is by induction on n.

(Base case $n = 0$) In this case $t = \varepsilon$, so $p_n = p$. The string s is empty as well, so we have to derive $p \overset{\varepsilon}{\Longrightarrow}_{\not\checkmark} p$. The derivation amounts in an application of the axiom,

$$\frac{}{p \overset{\varepsilon}{\Longrightarrow}_{\not\checkmark} p}\ p \overset{\checkmark}{\not\longrightarrow};\ [\text{UT-AX}]$$

(Inductive case $n = m + 1$) In this case $t = \alpha_1 t'$ or $t = \tau t'$ for some $t' \in Act^\star$ such that t' has length m.

In the first case, there exists a p_1 such that $p \xrightarrow{\alpha} p_1 \xrightarrow{t'} p_n$. Since for every $1 \leq i \leq n$, $p_i \overset{\checkmark}{\not\longrightarrow}$, and t' is shorter than t, the inductive hypothesis ensures that there exists a finite derivation of $p_1 \overset{\langle t' \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} p_n$. We extend this derivation as follows,

$$\frac{\dfrac{\vdots}{p_1 \overset{\langle t' \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} p_n}}{p \overset{\alpha\langle t' \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} p_n}\ p \overset{\checkmark}{\not\longrightarrow}, p \xrightarrow{\alpha} p_1\ [\text{UT-ALPHA}]$$

Since $\alpha\langle t' \rangle_{\setminus\tau} = \langle \alpha t' \rangle_{\setminus\tau} = \langle t \rangle_{\setminus\tau} = s$, we have derived $p \overset{s}{\Longrightarrow}_{\not\checkmark} q$.

In the second case, there exists a p' such that $p \xrightarrow{\tau} p' \xrightarrow{t'} q$. e, there exists a p_1 such that $p \xrightarrow{\tau} p_1 \xrightarrow{t'} p_n$. Since for every $1 \leq i \leq n$, $p_i \overset{\checkmark}{\not\longrightarrow}$, and t' is shorter than t, the inductive hypothesis ensures that there exists a finite derivation of $p_1 \overset{\langle t' \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} p_n$. Now we apply rule [UT-TAU],

$$\frac{\dfrac{\vdots}{p' \overset{\langle t' \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} p_n}}{p \overset{\langle t' \rangle_{\setminus\tau}}{\Longrightarrow}_{\not\checkmark} p_n}\ p \overset{\checkmark}{\not\longrightarrow}, p \xrightarrow{\tau} p_1\ [\text{UT-TAU}]$$

As $\langle t' \rangle_{\setminus\tau} = \langle \tau t' \rangle_{\setminus\tau} = \langle t \rangle_{\setminus\tau} = s$, we have derived $p \overset{s}{\Longrightarrow}_{\not\checkmark} q$. □

By using unsuccessful traces, we can define a notion of acceptance set that suits our aim (i.e. characterise $\sqsubseteq_{\text{CLT}}$).

Definition 4.2.8. [Unsuccesful acceptance sets]
For every process p and trace $s \in Act^\star$, let

$$\text{ACC}_{\not\checkmark}(p, s) = \{\, S(q) \mid p \overset{s}{\Longrightarrow}_{\not\checkmark} q \overset{\tau}{\not\longrightarrow} \,\}$$

We call the set $\text{ACC}_{\not\checkmark}(p, s)$ the *unsuccessful* acceptance set of p after s. □

We can now try to adapt the characterisation for servers in Theorem 4.1.21 to clients as follows:

Definition 4.2.9. Let $r_1 \preccurlyeq_{\mathsf{bad}} r_2$ if for every $s \in Act^\star$, if $r_1 \Downarrow s$ then

(i) $r_2 \Downarrow s$,

(ii) for every $B \in \text{ACC}_{\not\checkmark}(r_2, s)$, there exists some $A \in \text{ACC}_{\not\checkmark}(r_1, s)$ such that $A \subseteq B$. □

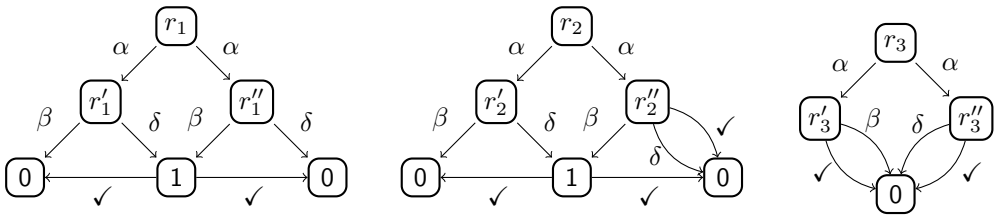


Figure 4.6: Unusable and usable processes (see Example 4.2.13)

Example 4.2.10. [Ready sets of deadlocks]
The difference between Definition 4.2.8 and Definition 4.1.11 is that unsuccessful acceptance sets are defined taking into the account only unsuccessful traces, and we have already motivated this difference. There is a second more subtle difference. While in Definition 4.1.11 it is not necessary to use deadlock states, in Definition 4.2.8 it is necessary. We explain why. Let $\preccurlyeq'_{\mathsf{bad}}$ be defined as $\preccurlyeq_{\mathsf{bad}}$, but omitting the requirement that $q \stackrel{\tau}{\not\longrightarrow}$ in the definition of unsuccessful acceptance sets. The relation $\preccurlyeq'_{\mathsf{bad}}$ is not complete: $\sqsubseteq_{\mathrm{CLT}} \not\subseteq \preccurlyeq'_{\mathsf{bad}}$. For instance, observe that $\alpha.1 \sqsubseteq_{\mathrm{CLT}} 1 \oplus \alpha.1$. We cannot prove $\alpha.1 \preccurlyeq'_{\mathsf{bad}} 1 \oplus \alpha.1$. The problem is that $\emptyset \in \mathrm{ACC}_{\not\checkmark}(1 \oplus \alpha.1, \varepsilon)$, that the set $\mathrm{ACC}_{\not\checkmark}(\alpha.1, \varepsilon)$ contains only the singleton $\{\alpha\}$, and that $\{\alpha\} \not\subseteq \emptyset$. □

Intuitively, only deadlock states have to be taken into the account in the unsuccessful acceptance sets, because the interactions they offer are necessary to reach a successful state. In Example 4.2.10, the term $1 \oplus \alpha.1$ does not need to perform α in order to reach success, as it is not stable, and, in particular, it can reduce to a successful state. On the other hand $1 \oplus \alpha.1 \stackrel{\tau}{\longrightarrow} \alpha.1$ and the client $\alpha.1$ has to perform α in order to report success.

Unfortunately, as the name suggests, there are still problems with the alternative pre-order $\preccurlyeq_{\mathsf{bad}}$.

Example 4.2.11. One can show that $r \sqsubseteq_{\mathrm{CLT}} \gamma.\alpha.1$ where r denotes the client $\gamma.(\alpha.1 + \beta.0)$. However they are not related by the proposed $\preccurlyeq_{\mathsf{bad}}$ in Definition 4.2.9, as the comparison of the acceptance sets fails. Obviously $r \Downarrow \gamma$ and $\{\alpha\} \in \mathrm{ACC}_{\not\checkmark}(\gamma.\alpha.1, c)$. But there is no $B \in \mathrm{ACC}_{\not\checkmark}(r, \gamma)$ such that $B \subseteq \{\alpha\}$; this is because $\mathrm{ACC}_{\not\checkmark}(r, \gamma)$ contains only one element, namely $\{\alpha, \beta\}$.

The problem is the presence of b in the ready set of $\alpha.1 + \beta.0$. □

To overcome this problem we need to develop even more notation. But first we give some intuition. No server that satisfies the client r in Example 4.2.11 can ever offer an interaction on β after an offer on the unsuccessful trace γ, because this would make the client fail. Intuitively the action β is *unusable* for r after having performed the unsuccessful trace γ; this is because performing β leads to a client, 0, which is *unusable*, because it can never be satisfied by any server. So when comparing ready sets after unsuccessful traces in Definition 4.2.9 we should ignore occurrences of *unusable* actions.

To formalise this notion of usable actions of a client after an unsuccessful trace s we use the notion of usable client (Definition 4.2.2). We also modify the definition of p AFTER s, which gives the set of residuals of p after any trace s, so that only the unsuccessful traces are considered

Definition 4.2.12. [Unsuccesful after]
For any process $r \in \mathsf{CCS}_{\mathrm{w}\tau}$ and $s \in Act^\star$ let $(r \text{ AFTER}_{\not\checkmark} s) = \{\, q \mid r \stackrel{s}{\Longrightarrow}_{\not\checkmark} q \,\}$. □

Example 4.2.13. In this example we use $\text{AFTER}_{\not\checkmark}$ to discuss the usability of the processes in Figure 4.6. Consider the left-most process, $r_1 = \alpha.(\beta.0 + \delta.1) + \alpha.(\beta.1 + \delta.0)$. The longest traces that r_1 are ab and ad. According to Definition 4.2.8,

$$\begin{aligned} (r_1 \text{ AFTER}_{\not\checkmark} \alpha\beta) &= \{0\} \\ (r_1 \text{ AFTER}_{\not\checkmark} \alpha\delta) &= \{0\} \end{aligned}$$

$$\frac{}{r \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon} \ r \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}; \ [\textsc{cconv-ax}]$$

$$\frac{}{r \ \mathsf{usbl}_{\not\checkmark} \ \alpha s} \ r \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}, r \stackrel{\alpha}{\not\Longrightarrow}_{\not\checkmark}; \ [\textsc{cconv-not}]$$

$$\frac{\bigoplus(r \ \textsc{after}_{\not\checkmark} \ \alpha) \ \mathsf{usbl}_{\not\checkmark} \ s}{r \ \mathsf{usbl}_{\not\checkmark} \ \alpha s} \ r \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}, r \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}; \ [\textsc{cconv-alpha}]$$

Figure 4.7: Inference rules for the functional $\mathcal{F}_{\mathsf{usbl}_{\not\checkmark}}$

The sets above contain an unusable client, namely $\mathsf{0}$. This means that while performing the traces $\alpha\beta$ and $\alpha\delta$, the process r_1 may reach $\mathsf{0}$ and we do not know, a priori, if this will happen or not. Indeed, one can prove that $r_1 \notin \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$.

The process $r_2 = \alpha.(\beta.\,\mathsf{0} + \delta.\,\mathsf{1}) \ + \ \alpha.(\mathsf{1} + \beta.\,\mathsf{1} + \delta.\,\mathsf{0})$, on the other hand is usable. Consider the following sets

$$\begin{array}{rcl}(r_2 \ \textsc{after}_{\not\checkmark} \ \alpha\beta) & = & \{\mathsf{0}\} \\ (r_2 \ \textsc{after}_{\not\checkmark} \ \alpha\delta) & = & \emptyset\end{array}$$

The process r_2 has the same issue as r_1 along $\alpha\beta$ (i.e.. it may non-deterministically fail), but it performs $\alpha\delta$ passing always via a successful state; this is the reason why $(r_1 \ \textsc{after}_{\not\checkmark} \ ad)$ is the empty set. Indeed, if $r_2 \xrightarrow{\alpha} r_2^1 \xrightarrow{\delta} r_2^2$ then either

- $r_2^1 = r_2'$, $r_2^2 = \mathsf{1}$ and so $r_2^2 \xrightarrow{\checkmark}$, or
- $r_2^1 = r_2''$ and so $r_2^1 \xrightarrow{\checkmark}$

Indeed one can prove $\overline{\alpha}.\overline{\delta}.\,\mathsf{0} \ \textsc{must} \ r_2$, so $r_1 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$.

Let us apply $\textsc{after}_{\not\checkmark}$ to the longest traces of $r_3 = \alpha.(\beta.\,\mathsf{0} + \mathsf{1}) \ + \ \alpha.(\mathsf{1} + \delta.\,\mathsf{0})$.

$$\begin{array}{rcl}(r_3 \ \textsc{after}_{\not\checkmark} \ \alpha\beta) & = & \emptyset \\ (r_3 \ \textsc{after}_{\not\checkmark} \ \alpha\delta) & = & \emptyset\end{array}$$

Similar to r_2 the trace $\alpha\delta$ always leads r_3 to a successful state, and so does the trace $\alpha\beta$; for instance one can show that $\overline{\alpha}.\overline{\beta}.\,\mathsf{0} \ \textsc{must} \ r_3$. It follows that also r_3 is a usable client, $r_3 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$. □

Now we define a convergence predicate for clients.

Definition 4.2.14. [Usability after unsuccesful traces]
Let $\mathcal{F}_{\mathsf{usbl}_{\not\checkmark}} : \mathcal{P}(\mathsf{CCS}_{\mathrm{w}\tau} \times Act^\star) \longrightarrow \mathcal{P}(\mathsf{CCS}_{\mathrm{w}\tau} \times Act^\star)$ be the rule functional given by the inference rules in Figure 4.7. Lemma C.0.23 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_{\mathsf{usbl}_{\not\checkmark}}(X)$; we call this solution the *client convergence predicate*, and we denote it $\mathsf{usbl}_{\not\checkmark}$: That is $\mathsf{usbl}_{\not\checkmark} = \mu X.\mathcal{F}_{\mathsf{usbl}_{\not\checkmark}}(X)$. We extend the relation $\mathsf{usbl}_{\not\checkmark}$ to infinite strings by letting for every $u \in Act^\infty$, $p \ \mathsf{usbl}_{\not\checkmark} \ u$ if and only if $p \ \mathsf{usbl}_{\not\checkmark} \ u_k$ for every finite prefix u_k of u. □

Although the predicate $\mathsf{usbl}_{\not\checkmark}$ seems to check only the usability of a client, in Lemma 4.2.21 we will establish that $\mathsf{usbl}_{\not\checkmark}$ enforces also a form of convergence. This justifies the symbol $\Downarrow$.

Intuitively $r \ \mathsf{usbl}_{\not\checkmark} \ s$ means that any state reachable from r by performing any subsequence of s is usable.

Lemma 4.2.15. For every $s, s' \in Act^\star$ and $r \in \mathsf{CCS}_{\mathrm{w}\tau}$, if $r \ \mathsf{usbl}_{\not\checkmark} \ s$, s' is a prefix of s and $r \stackrel{s'}{\Longrightarrow}_{\not\checkmark}$, then $\bigoplus(r \ \textsc{after}_{\not\checkmark} \ s') \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$.

Proof. First observe that $\bigoplus(r \ \textsc{after}_{\not\checkmark} \ \varepsilon) \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$. This is true because regardless of the string s, to derive $r \ \mathsf{usbl}_{\not\checkmark} \ s$ it is necessary that $r \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$.

We reason by induction on the length of s.

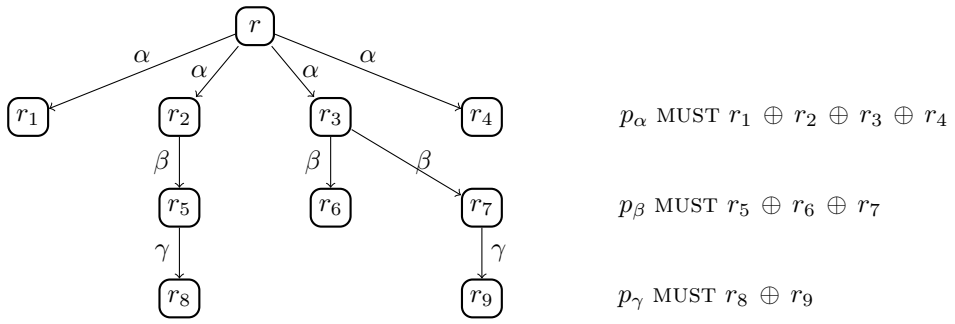


Figure 4.8: Suppose that r $\mathsf{usbl}\not\swarrow$ $\alpha\beta\gamma$. The existence of p_α, p_β, and p_γ is a consequence of Lemma 4.2.15.

(Base case, $\mathsf{len}(s) = 0$) In this case $s = \varepsilon$, so if s' is a prefix of s then it is empty. We have to prove that $\bigoplus(r \text{ AFTER}\not\swarrow \varepsilon)$; we have already seen that this is true.

(Inductive case, $\mathsf{len}(s) = n + 1$) In this case $s = \alpha s''$ for some $s'' \in Act^\star$; and we have to prove that for every s' prefix of s, $\bigoplus(r \text{ AFTER}\not\swarrow \alpha s') \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$.

Fix a s' that is a prefix of s. If $s' = \varepsilon$, then we have to show that $\bigoplus(r \text{ AFTER}\not\swarrow \epsilon)$; we have already proven this. Suppose now that $s' \neq \varepsilon$; then $s' = \alpha w$, where w is a prefix of s''. We have to show that $\bigoplus(r \text{ AFTER}\not\swarrow \alpha w) \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$. Definition 4.2.12 implies that

$$\bigoplus(r \text{ AFTER}\not\swarrow \alpha w) = \bigoplus(\bigoplus(r \text{ AFTER}\not\swarrow \alpha) \text{ AFTER}\not\swarrow w)$$

so, by letting $\hat{r} = \bigoplus(r \text{ AFTER}\not\swarrow \alpha)$, it is enough to to prove that $\bigoplus(\hat{r} \text{ AFTER}\not\swarrow w) \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$.

Since s'' is shorter than s, the inductive hypothesis states that

$$\text{for every } w' \in Act^\star \text{ and } r' \in \mathsf{CCS}_{\text{w}\tau}, \text{ if } r \;\mathsf{usbl}\not\swarrow\; w',\ w' \text{ is a prefix of } s'' \text{ and } r \overset{w'}{\Longrightarrow}\not\swarrow,$$
$$\text{then } \bigoplus(r \text{ AFTER}\not\swarrow w') \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}.$$

The hypothesis r $\mathsf{usbl}\not\swarrow$ αw implies that $\hat{r}$ $\mathsf{usbl}\not\swarrow$ w; the hypothesis $r \overset{\alpha w}{\Longrightarrow}\not\swarrow$ implies that $\hat{r} \overset{w}{\Longrightarrow}\not\swarrow$, and by assumption w is a prefix of s''.

The inductive hypothesis implies that $\bigoplus(\hat{r} \text{ AFTER } w) \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$. □

In Figure 4.8 we suppose that the client r $\mathsf{usbl}\not\swarrow$ $\alpha\beta\gamma$, and it performs the trace $\alpha\beta\gamma$ unsuccessfully. The existence of the processes p's on the right side of the the figure is the consequence of Lemma 4.2.15. We do not know how these servers p's are defined; nevertheless, Lemma 4.2.15 ensures that they exist so in our reasoning we can use them.

Now the set of usable actions for a client can be defined as follows.

Definition 4.2.16. [Usable actions after unsuccesful trace]
For every $r \in \mathsf{CCS}_{\text{w}\tau}$ and $s \in Act^\star$, let

$$\mathsf{ua}\not\swarrow(r, s) = \{\, \alpha \in Act \mid r \overset{s\alpha}{\Longrightarrow}\not\swarrow \text{ implies } r \;\mathsf{usbl}\not\swarrow\; s\alpha \,\}$$

be the set of *usable actions of r after s*. □

Thus if $\alpha \in \mathsf{ua}\not\swarrow(r, s)$ we know that the set of clients $(r \text{ AFTER}\not\swarrow s\alpha)$ is non-empty, and the client given by the internal choice among them is usable; that is, there is some server which satisfies it.

Example 4.2.17. Let us revisit Example 4.2.11. First note that although r can perform the sequence $\gamma\beta$, the action β is not in $\mathsf{ua}\not\swarrow(r, \gamma)$ because $(r \text{ AFTER}\not\swarrow \gamma\beta)$ is the singleton set containing $\mathsf{0}$, which is not in $\mathcal{U}_{\text{CLT}}^{\text{MUST}}$. Instead we have $\mathsf{ua}\not\swarrow(r, \gamma) = \{\alpha\}$.

Now suppose we were to amend Definition 4.2.9 so that instead of demanding $A \subseteq B$, we relaxed this to $A \cap \mathsf{ua}_{\not\checkmark}(r_1, s) \subseteq B$. It would then follow that $r \preccurlyeq_{\mathsf{bad}} \gamma.\alpha.\mathsf{1}$, thereby correctly reflecting the fact that $r \sqsubseteq_{\mathrm{CLT}} \gamma.\alpha.\mathsf{1}$. □

It is essential that in Definition 4.2.14 we consider only the unsuccessful traces s rather than all the traces.

Example 4.2.18. We explain the previous statement. Consider the client

$$r = \beta.((\mathsf{1} + \alpha.\mathsf{0}) \oplus \alpha.(\mathsf{1} \oplus \mathsf{1}))$$

and note that $\overline{\beta}.\overline{\alpha}.\mathsf{0}$ MUST r while $\overline{\beta}.\overline{\alpha}.\mathsf{0} \not\mathrm{MUST}\ \beta.\mathsf{0}$, and so $r \not\sqsubseteq_{\mathrm{CLT}} \beta.\mathsf{0}$

Now consider the consequences of using AFTER rather than AFTER$_{\not\checkmark}$ in Definition 4.2.14. The proposed amendment to the definition of $\preccurlyeq_{\mathsf{bad}}$ suggested in Example 4.2.17 would no longer be sound, as $r \preccurlyeq_{\mathsf{bad}} \beta.\mathsf{0}$ would be a consequence.

This is because $(r \text{ AFTER } \beta\alpha)$ is the set $\{\mathsf{0}, \mathsf{1}\}$ and so $\bigoplus(r \text{ AFTER } \beta\alpha)$ is the client $\mathsf{0} \oplus \mathsf{1}$ which is not in $\mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$. This would lead in turn to $\mathsf{ua}_{\not\checkmark}(r, \beta)$ being $\emptyset$, from which $r \preccurlyeq_{\mathsf{bad}} \beta.\mathsf{0}$ would follow. The incorrect reasoning involves the unsuccessful acceptances after the trace β. $\mathrm{ACC}_{\not\checkmark}(\beta.\mathsf{0}, \beta) = \{\emptyset\}$ and the unique ready set it contains, $\emptyset$, can be matched by $A \cap \emptyset$ for some set $A \in \mathrm{ACC}_{\not\checkmark}(r, \beta)$, namely $A = \{\, a \,\}$.

However, with the correct Definition 4.2.14 this reasoning no longer works, as $\mathsf{ua}_{\not\checkmark}(r, \beta) = \{\alpha\}$.
□

The amendment to Definition 4.2.9 suggested in Example 4.2.17 is still not sufficient to obtain a complete characterisation of the client pre-order.

Example 4.2.19. Consider the clients $r_1 = \alpha.(\beta.\delta.\mathsf{0} + \beta.\mathsf{1})$ and $r_2 = \alpha.\gamma.\delta.\mathsf{1}$. The term r_1 is not usable, so $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$, although $r_1 \not\preccurlyeq_{\mathsf{bad}} r_2$, even when $\preccurlyeq_{\mathsf{bad}}$ is amended as suggested in Example 4.2.17. To see this first note $\{\delta\} \in \mathrm{ACC}_{\not\checkmark}(r_2, \alpha\gamma)$, and $r_1 \Downarrow \alpha\gamma$, although r_1 can not actually perform the sequence of actions $\alpha\gamma$; $r_1 \Downarrow \alpha\gamma$ merely says that if r_1 can perform any prefix of the sequence ac to reach r' then r' must converge. Consequently $\mathrm{ACC}_{\not\checkmark}(r_1, \alpha\gamma)$ is empty and thus no ready set B can be found to match the ready set $\{\delta\}$. □

To fix this problem we need to relax the circumstances under which the ready sets in Definition 4.2.9 are matched. Note that there the predicate $\Downarrow s$ already moderates when the matching is required. For example $\alpha.(\tau^\infty + \beta.\mathsf{1}) \preccurlyeq_{\mathsf{bad}} \alpha.\gamma.\delta.\mathsf{1}$, where τ^∞ denotes some process which does not converge. This is because $\alpha.(\tau^\infty + \beta.\mathsf{1}) \Downarrow a$ is false and therefore the ready set $\{\gamma\} \in \mathrm{ACC}_{\not\checkmark}(\alpha.\gamma.\delta.\mathsf{1}, \alpha)$ does not have to be matched by $\alpha.(\tau^\infty + \beta.\mathsf{1})$. A convenient way to address the problem encountered in Example 4.2.19 is to strengthen this convergence predicate.

The client convergence predicate suits our aims, as it describes precisely when we expect ready sets and unsuccessful traces to be compared. We explain this in a series of lemmas.

First we deal with some technicalities. We introduce a predicate that ensures that if a client diverges, then it reaches a successful state.

Definition 4.2.20. [Convergence to success]
We say that r *converges to success*, denoted $r \Downarrow^{\checkmark}$, whenever if there exists an infinite reduction sequence as $r = r_0 \xrightarrow{\tau} r_1 \xrightarrow{\tau} r_2 \xrightarrow{\tau} r_3 \xrightarrow{\tau} \ldots$ then there exists $n \in N$ such that $r_n \xrightarrow{\checkmark}$. □

The predicate $\Downarrow^{\checkmark}$ ensures that if a process diverges, then along the diverging computations it reaches a successful state in a finite amount of internal moves. For instance $\tau^\infty \not\Downarrow^{\checkmark}$, whereas if we let

$$r = \mathsf{0} \oplus (\mathsf{0} \oplus (\mathsf{1} + \tau^\infty))$$

then $r \Downarrow^{\checkmark}$; this is true because the infinite computation

$$r \xrightarrow{\tau} \mathsf{0} \oplus (\mathsf{1} + \tau^{\infty}) \xrightarrow{\tau} \mathsf{1} + \tau^{\infty} \xrightarrow{\tau} \mathsf{1} + \tau^{\infty} \xrightarrow{\tau} \dots$$

is the only diverging computation of r, and it reaches a successful state after 2 reductions. Note that $\mathsf{0} \oplus \mathsf{0} \Downarrow^{\checkmark}$; this is true because $\mathsf{0} \oplus \mathsf{0}$ performs no infinite series of reductions, so the predicate $\Downarrow^{\checkmark}$ is trivially true.

Since for every $s \in Act^{\star}$, r $\mathsf{usbl}_{\not\checkmark}$ s ensures that $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$, there is relation between the convergence predicate for client, $\mathsf{usbl}_{\not\checkmark}$, and the predicate of convergence to success $\Downarrow^{\checkmark}$.

Lemma 4.2.21. If $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ then $r \Downarrow^{\checkmark}$.

Proof. As $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ there exists a p such that p MUST r. Fix a divergent computation of r, and zip it with p,

$$p \parallel r = p \parallel r_0 \xrightarrow{\tau} p \parallel r_1 \xrightarrow{\tau} p \parallel r_2 \xrightarrow{\tau} \dots$$

The definition of MUST ensures that one of the derivatives of r is successful. □

Note that r $\mathsf{usbl}_{\not\checkmark}$ s ensures that r converges only while performing *unsuccessful* traces; for instance $(\mathsf{1} + \tau^{\infty})$ $\mathsf{usbl}_{\not\checkmark}$ ε, and τ^{∞} $\not\mathsf{usbl}_{\not\checkmark}$ ε. We motivate this choice in Example 4.2.33.

We have enough material to expose the properties of $\sqsubseteq_{\text{SVR}}$ that we require in the characterisation.

Lemma 4.2.22. Suppose $r_1 \sqsubseteq_{\text{CLT}} r_2$ and $\alpha s \in Act^{\star}$. If r_1 $\mathsf{usbl}_{\not\checkmark}$ αs and $r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$. Then

(i) $r_1 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$

(ii) $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha)$ $\mathsf{usbl}_{\not\checkmark}$ s

(iii) $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha) \sqsubseteq_{\text{CLT}} \bigoplus(r_2 \text{ AFTER}_{\not\checkmark} \alpha)$

Proof. We divide the argument in three parts, which prove respectively point (i), point (ii) and point (iii).

We prove that $r_1 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$. The hypothesis r_1 $\mathsf{usbl}_{\not\checkmark}$ αs implies that $r_1 \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$, thus there exists a p such that p MUST r_1. Let $\hat{p} = p + \overline{\alpha}.\tau^{\infty}$. By hypothesis there exists a r_2' such that $r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark} r_2'$; it follows that the composition $r_2 \parallel \hat{p}$ performs the following maximal computation

$$r_2 \parallel \hat{p} \Longrightarrow r_2' \parallel \tau^{\infty} \Longrightarrow r_2' \parallel \tau^{\infty} \Longrightarrow r_2' \parallel \tau^{\infty} \Longrightarrow \dots$$

The computation above is due to an interaction (via α) and then to the divergence of τ^{∞}. The derivatives of r_2 that appear in the computation above also appear in $r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark} r_2'$, so they are not successful. It follows that the maximal computation we have depicted is not client-successful, and so $\hat{p}$ $\not$MUST r_2. The hypothesis $r_1 \sqsubseteq_{\text{CLT}} r_2$ ensures that $\hat{p}$ $\not$MUST r_1, so there exists a maximal computation of $\hat{p} \parallel r_1$ that is not client-successful. In view of the construction of p, this computation cannot be due to p, for otherwise p $\not$MUST r_1, so it must be due to $\overline{\alpha}.\tau^{\infty}$. This is possible only if $r_1 \stackrel{\alpha}{\Longrightarrow}$. Since the resulting computation is not client-successful it follows that $r_1 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark} r_1'$.

We prove point (ii); namely that $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha)$ $\mathsf{usbl}_{\not\checkmark}$ s. By hypothesis r_1 $\mathsf{usbl}_{\not\checkmark}$ αs, so if $r_1 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$ then $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha)$ $\mathsf{usbl}_{\not\checkmark}$ s. We have already proven that $r_1 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$, so $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha)$ $\mathsf{usbl}_{\not\checkmark}$ s.

Now we prove point (iii). First, note that point (i) and Definition 4.2.12 imply that $(r_1 \text{ AFTER}_{\not\checkmark} \alpha) \neq \emptyset$, so the term $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha)$ exists. We have to explain why $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha) \sqsubseteq_{\text{CLT}} \bigoplus(r_2 \text{ AFTER}_{\not\checkmark} \alpha)$. Let p' MUST $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha)$; we have to prove that p' MUST $\bigoplus(r_2 \text{ AFTER}_{\not\checkmark} \alpha)$. Let $\hat{p} = p + \overline{\alpha}.p'$, where p MUST r_1. The ensuing implication is true

$$\text{if } \hat{p} \text{ MUST } r_2 \text{ then } p' \text{ MUST } \bigoplus(r_2 \text{ AFTER}_{\not\checkmark} \alpha).$$

In view of the hypothesis that $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$, to show that $\hat{p}$ MUST r_2 it suffices to prove that $\hat{p}$ MUST r_1. This is what we prove. Definition 3.1.1 requires us to show that all the maximal computations of $r_1 \parallel \hat{p}$ are client-successful.

Fix a maximal computation of $r_1 \parallel \hat{p}$,

$$r_1 \parallel \hat{p} = r_1^0 \parallel \hat{p}_0 \xrightarrow{\tau} r_1^1 \parallel \hat{p}_1 \xrightarrow{\tau} r_1^2 \parallel \hat{p}_2 \xrightarrow{\tau} \dots \tag{4.10}$$

In the computation above either there are no interactions, or an interaction happens.

Suppose that no interaction happens. Then $\hat{p}_1 \xrightarrow{\tau} \hat{p}_2 \xrightarrow{\tau} \hat{p}_3 \xrightarrow{\tau}$ and $r_1^0 \xrightarrow{\tau} r_1^1 \xrightarrow{\tau} r_1^2 \xrightarrow{\tau} \dots$ If the computation reaches a stable state $r_1^i \parallel \hat{p}_i$, then there exists a p' such that $p \stackrel{\tau}{\Longrightarrow} p' \stackrel{\tau}{\not\longrightarrow}$, and $r_1^i \parallel p' \stackrel{\tau}{\not\longrightarrow}$; it follows that there exists also the maximal computation

$$r_1 \parallel p \stackrel{\tau}{\Longrightarrow} r_1^i \parallel p' \stackrel{\tau}{\not\longrightarrow}$$

The assumption p MUST r_1 implies that the computation is client-successful; this ensures that the computation in (4.10) is client-successful as well.

If the computation in (4.10) contains no stable state, then one of the processes diverge. If the client diverges, then there exists the infinite computation

$$r_1^0 \parallel p \xrightarrow{\tau} r_1^1 \parallel p \xrightarrow{\tau} r_1^2 \parallel p \xrightarrow{\tau} \dots$$

The assumption p MUST r_1 implies that the computation we unzipped is client-successful. If the server diverges then there exists the infinite computation

$$r_1 \parallel p \xrightarrow{\tau} r_1 \parallel p_1 \xrightarrow{\tau} r_1 \parallel p_2 \xrightarrow{\tau} \dots$$

The assumption p MUST r_1 implies that $r_1 \xrightarrow{\checkmark}$, so the computation we unzipped is client-successful.

Suppose now that the computation in (4.10) contains interactions. Let $r_1^i \parallel p_i$ be the state that performs the *first* interaction; that is $r_1^0 \stackrel{\tau}{\Longrightarrow} r_1^i$ and $\hat{p} \stackrel{\tau}{\Longrightarrow} p_i$. Moreover, let $r_1^{i+1} \parallel p_{i+1}$ be the state reached by the interaction that is

$$\frac{r_1^i \xrightarrow{\delta} r_1^{i+1} \quad p_i \xrightarrow{\overline{\delta}} p_{i+1}}{r_1^i \parallel p_i \xrightarrow{\tau} r_1^{i+1} \parallel p_{i+1}} \text{ [P-SYNCH]}$$

If $r_1^{i+1} \notin (r_1 \text{ AFTER}_{\not\checkmark} \delta)$, then there exists a $0 \leq k \leq i+1$ such that $r_1^k \xrightarrow{\checkmark}$, so the computation in (4.10) is client-successful. In the opposite case we have $r_1 \stackrel{\delta}{\Longrightarrow}_{\not\checkmark} r_1^{i+1}$. Our reasoning now depends on the process p_{i+1}.

- If $p_{i+1} = p'$, then $\delta = \alpha$. It follows that $r_1^{i+1} \in (r_1^0 \text{ AFTER}_{\not\checkmark} \alpha)$. By construction we know that p_{i+1} MUST $\bigoplus(r_1^0 \text{ AFTER}_{\not\checkmark} \alpha)$, so the computation in (4.10) must be client-successful.
- If $p_{i+1} \neq p'$, then $p \stackrel{\overline{\delta}}{\Longrightarrow} p_{i+1}$. The facts that p MUST r_1, and $r_1^0 \stackrel{\delta}{\Longrightarrow}_{\not\checkmark} r_1^{i+1}$ ensure that p_{i+1} MUST r_1^{i+1}, and so the computation in (4.10) is client-successful.

We have proven that a maximal computation of $r_1 \parallel \hat{p}$ is client-successful. Since we used no assumption on the maximal computation, we have shown that all the maximal computations of $r_1 \parallel \hat{p}$ are client-successful, and so Definition 3.1.1 ensures that $\hat{p}$ MUST r_1.

□

Lemma 4.2.23. Suppose that $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$. For every $s \in Act^\star$, if r_1 $\mathsf{usbl}_{\not\checkmark}$ s then r_2 $\mathsf{usbl}_{\not\checkmark}$ s.

Proof. We begin by proving that $r_2 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$. By hypothesis r_1 $\mathsf{usbl}_{\not\checkmark}$ ε, so there exists a p such that p MUST r_1. The hypothesis $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$ ensures that p MUST r_2; this implies that $r_2 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$.

Now we prove the lemma, reasoning by induction on the length of s.

Base case ($\mathsf{len}(s) = 0$) In this case $s = \varepsilon$ and we have to prove $r_2 \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon$. Since $r_2 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$ we use the axiom in Figure 4.7 to derive

$$\frac{}{r_1 \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon} \ r_2 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}} \ [\text{CCONV-AX}]$$

Inductive case ($\mathsf{len}(s) = n + 1$) In this case $s = \alpha s'$, and we have to prove that $r_2 \ \mathsf{usbl}_{\not\checkmark} \ \alpha s'$. If $r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$, then we can infer

$$\frac{}{r_1 \ \mathsf{usbl}_{\not\checkmark} \ \alpha s'} \ r_2 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}, r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark} \ [\text{CCONV-NOT}]$$

If $r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$, then we have to prove that $\bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha) \ \mathsf{usbl}_{\not\checkmark} \ s'$. The hypothesis $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$, $r_1 \ \mathsf{usbl}_{\not\checkmark} \ \alpha s'$ and the assumption $r_1 \stackrel{\alpha}{\Longrightarrow}$ allow us to use Lemma 4.2.22. That lemma implies

- $\bigoplus(r_1 \ \text{AFTER}_{\not\checkmark} \ \alpha) \sqsubseteq_{\mathrm{CLT}} \bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha)$
- $\bigoplus(r_1 \ \text{AFTER}_{\not\checkmark} \ \alpha) \ \mathsf{usbl}_{\not\checkmark} \ s'$

As s' has length n we are allowed to apply the inductive hypothesis, which implies that $\bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha) \ \mathsf{usbl}_{\not\checkmark} \ s'$. Now we derive

$$\frac{\bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha) \ \mathsf{usbl}_{\not\checkmark} \ s'}{r_1 \ \mathsf{usbl}_{\not\checkmark} \ \alpha s'} \ r_2 \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}, r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark} \ [\text{CCONV-ALPHA}]$$

□

Lemma 4.2.24. Suppose $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$, and $r_1 \ \mathsf{usbl}_{\not\checkmark} \ s$. If $r_2 \stackrel{s}{\Longrightarrow}_{\not\checkmark}$ then $r_1 \stackrel{s}{\Longrightarrow}_{\not\checkmark}$.

Proof. First we prove that $r_1 \stackrel{\checkmark}{\not\longrightarrow}$. By hypothesis we know that $r_2 \stackrel{s}{\Longrightarrow}_{\not\checkmark}$ for some s; this ensures that $r_2 \stackrel{\checkmark}{\not\longrightarrow}$. As the process τ^∞ diverges, we can infer the following maximal computation,

$$r_2 \parallel \tau^\infty \Longrightarrow r_2 \parallel \tau^\infty \Longrightarrow r_2 \parallel \tau^\infty \Longrightarrow \ldots$$

Since the computation is not client-successful, it follows that that $\tau^\infty \not\text{MUST} \ r_2$, and so the hypothesis $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$ implies that $\tau^\infty \not\text{MUST} \ r_1$; it follows that $r_1 \stackrel{\checkmark}{\not\longrightarrow}$, for otherwise $\tau^\infty \ \text{MUST} \ r_1$.

We are ready to prove the lemma. The argument is by induction on the length of s.

Base case ($|s| = 0$) In this case $s = \varepsilon$, so we have to prove $r_1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark}$. Consider the following inference tree,

$$\frac{}{r_1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark} r_1} \ r_1 \stackrel{\checkmark}{\not\longrightarrow} \ [\text{UT-AX}]$$

from which it follows $r_1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark}$.

Inductive case ($|s| = n+1$) In this case $s = \alpha s'$, so we know that (a) $r_1 \ \mathsf{usbl}_{\not\checkmark} \ \alpha s'$, and (b) $r_2 \stackrel{\alpha s'}{\Longrightarrow}_{\not\checkmark}$. We have to prove that $r_1 \stackrel{\alpha s'}{\Longrightarrow}_{\not\checkmark}$.

Point (b) above implies that $r_1 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$, so (a) and the hypothesis $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$ let us apply Lemma 4.2.22. It follows that

(1) $\bigoplus(r_1 \ \text{AFTER}_{\not\checkmark} \ \alpha) \sqsubseteq_{\mathrm{CLT}} \bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha)$

(2) $\bigoplus(r_1 \ \text{AFTER}_{\not\checkmark} \ \alpha) \ \mathsf{usbl}_{\not\checkmark} \ s'$

Since s' is shorter than s, we can use the inductive hypothesis, so in view of (1) and (2),

$$\bigoplus(r_2 \text{ AFTER}_{\not\checkmark} \alpha) \stackrel{s'}{\Longrightarrow}_{\not\checkmark} \text{ implies } \bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha) \stackrel{s'}{\Longrightarrow}_{\not\checkmark} \tag{4.11}$$

We prove that the premises of the implication above are true. The hypothesis $r_2 \stackrel{\alpha s'}{\Longrightarrow}_{\not\checkmark}$ ensures that there exists a r_2' such that $r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark} r_2' \stackrel{s'}{\Longrightarrow}_{\not\checkmark}$; Definition 4.2.12 implies that $r_2' \in (r_2 \text{ AFTER}_{\not\checkmark} \alpha)$. It follows that one can infer $\bigoplus(r_2 \text{ AFTER}_{\not\checkmark} \alpha) \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark} r_2' \stackrel{s'}{\Longrightarrow}_{\not\checkmark}$; in turn this implies that $\bigoplus(r_2 \text{ AFTER}_{\not\checkmark} \alpha) \stackrel{s'}{\Longrightarrow}_{\not\checkmark}$. The implication in (4.11) ensures that $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha) \stackrel{s'}{\Longrightarrow}_{\not\checkmark} r$ for some r. There exists a r_1' such that $r_1 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark} r_1' \stackrel{s'}{\Longrightarrow}_{\not\checkmark} r_1$. Proposition 4.2.7 and the transitivity of $\Longrightarrow$ imply that $r_1 \stackrel{s'}{\Longrightarrow}_{\not\checkmark} r$. □

Lemma 4.2.24 is true also for infinite unsuccessful traces.

Corollary 4.2.25. *Suppose* $r_1 \sqsubseteq_{\text{CLT}} r_2$ *and* $r_1 \text{ usbl}_{\not\checkmark} u$. *For every* $u \in Act^\infty$, *if* $r_2 \stackrel{u}{\Longrightarrow}_{\not\checkmark}$ *then* $r_1 \stackrel{u}{\Longrightarrow}_{\not\checkmark}$.

Proof. We have to prove that $r_1 \stackrel{u}{\Longrightarrow}_{\not\checkmark}$. Definition 4.2.6 requires us to show a $t \in \text{Act}_\tau^\infty$ such that

- if $t = \alpha_\tau^1\alpha_\tau^2\alpha_\tau^3 \ldots$, then $r = r_0 \stackrel{\alpha_\tau^1}{\longrightarrow} r_1 \stackrel{\alpha_\tau^2}{\longrightarrow} r_2 \stackrel{\alpha_\tau^3}{\longrightarrow} \ldots$, and for every $n \in \mathbb{N}$, $r_n \stackrel{\checkmark}{\not\longrightarrow}$;
- for every $n \in \mathbb{N}$ $u_n = \langle t_k \rangle_{\setminus\tau}$ for some $k \in \mathbb{N}$ □

Let $u = \alpha_1\alpha_2\alpha_3 \ldots$. The hypothesis $r_1 \text{ usbl}_{\not\checkmark} u$, Definition 4.2.14, and Lemma 4.2.15 imply that for every $k \in \mathbb{N}$ there exists a p_k such that $p_k \text{ MUST } \bigoplus(r_1 \text{ AFTER}_{\not\checkmark} u_k)$. For every $k \in \mathbb{N}$, let $A_k \stackrel{\text{def}}{=} p_k + \overline{\alpha_k}_{k+1}.A_{k+1}$.

The hypothesis $r_2 \stackrel{u}{\Longrightarrow}_{\not\checkmark}$ let us prove that $A_0 \not\text{MUST } r_2$, so the hypothesis $r_1 \sqsubseteq_{\text{CLT}} r_2$ implies that $A_0 \not\text{MUST } r_1$. There exists a maximal computation of $r_1 \mid\mid A_0$, say C, which is not client-successful. We prove that C is due to the trace u.

By hypothesis $r_1 \text{ usbl}_{\not\checkmark} u$, thus in C no derivatives of r_1 diverges, for otherwise C would be client-successful. Thanks to the construction of the p_k's, for every $k \in \mathbb{N}$, no interaction is due to the summand p_k, because otherwise the computation would be client-successful. It follows that the interactions that take place in C are due to the summands $\overline{\alpha_i}.A_i$. Let t be the contribution of r_1 in the computation C. What argued thus far implies that for every $n \in \mathbb{N}$ there exists a $k \in \mathbb{N}$ such that $\langle t_k \rangle_{\setminus\tau} = u_n$. Let $t = \alpha_\tau^1\alpha_\tau^2 \ldots$; in C the client r_1 performs the action sequence $r_1 = r_1^0 \stackrel{\alpha_\tau^1}{\longrightarrow} r_1^1 \stackrel{\alpha_\tau^2}{\longrightarrow} \ldots$; since C is not client-successful, it follows that for every $n \in \mathbb{N}$, $r_1^n \stackrel{\checkmark}{\not\longrightarrow}$. □

Lemma 4.2.24 is not true if there is no divergent term in the LTS.

Example 4.2.26. In Example 4.2.4 we have argued that $1 \not\sqsubseteq_{\text{CLT}} 1 \oplus 1$ because of the divergent server τ^∞. In this example we prove that convergent servers cannot distinguish 1 and $1 \oplus 1$. Suppose that $p \Downarrow$ for every p . Then $1 \sqsubseteq_{\text{CLT}} 1 \oplus 1$; to prove this we have to show that $p \text{ MUST } 1 \oplus 1$ for every p, under the assumption of convergence.

Take a maximal computation of $1 \oplus 1 \mid\mid p$:

$$1 \oplus 1 \mid\mid p \stackrel{\tau}{\longrightarrow} r_1 \mid\mid p_1 \stackrel{\tau}{\longrightarrow} \ldots \tag{4.12}$$

As $p \Downarrow$, any prefix of the computation due to internal moves of p is finite: for some $k \in \mathbb{N}$ the computation in (4.12) contains a state $p_k \mid\mid r_k$ such that $p_k \stackrel{\tau}{\not\longrightarrow}$ and $r_k = r_0$. Since the computation is maximal, and $r_k \stackrel{\tau}{\longrightarrow} 1$, the state $p_k \mid\mid r_k$ is followed by the state state $p_{k+1} \mid\mid r_{k+1}$ where $p_{k+1} = p_k$ and $r_{k+1} = 1$. Since $r_{k+1} \stackrel{\checkmark}{\longrightarrow}$ the computation in (4.12) is client-successful.

We have shown that $1 \sqsubseteq_{\text{CLT}} 1 \oplus 1$. To see why Lemma 4.2.24 is false, note that $1 \text{ usbl}_{\not\checkmark} \varepsilon$, and $1 \oplus 1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark}$. The lemma states that $1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark}$, but this is not true, because $1 \stackrel{\checkmark}{\longrightarrow}$.

In general $1 \oplus 1 \sqsubseteq_{\text{CLT}} 1$. What argued in the previous example implies that convergent servers cannot tell apart 1 from $1 \oplus 1$: $1 \eqsim_{\text{CLT}} 1 \oplus 1$. This means that under the assumption of convergence it is safe to postpone the action ✓ after any finite amount of internal moves. We depict this in Figure 4.9.

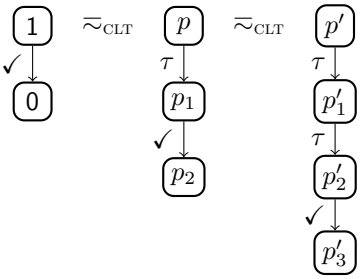


Figure 4.9: If servers converge, it is safe to perform internal computations before reaching success. See Example 4.2.26.

Lemma 4.2.27. Suppose $r_1 \sqsubseteq_{\overline{\approx}_{\text{CLT}}} r_2$. For every $s \in Act^\star$, if $r_1 \ \mathsf{usbl}_{\not\checkmark} \ s$ and $\text{ACC}_{\not\checkmark}(r_2, s) \neq \emptyset$, then $\text{ACC}_{\not\checkmark}(r_1, s) \neq \emptyset$.

Proof. The proof is by induction on the length of s.

Base case ($|s| = 0$) In this case $s = \varepsilon$, and we have to prove $\text{ACC}_{\not\checkmark}(r_1, \varepsilon) \neq \emptyset$. Definition 4.2.8 requires us to exhibit a r'_1 such that $r_1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark} r'_1 \stackrel{\tau}{\not\longrightarrow}$.

By hypothesis we know that $\text{ACC}_{\not\checkmark}(r_2, \varepsilon) \neq \emptyset$ and that $r_1 \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon$; the former fact ensures that for some r'_2, $r_2 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark} r'_2 \stackrel{\tau}{\not\longrightarrow}$. As the process $\mathsf{0}$ is stable and offers no actions, we can infer the following maximal computation,

$$r_2 \parallel \mathsf{0} \Longrightarrow r'_2 \parallel \mathsf{0} \Longrightarrow r'_2 \parallel \mathsf{0} \stackrel{\tau}{\not\longrightarrow}$$

The computation above is not client-successful, so $\mathsf{0} \not\text{MUST}\ r_2$, and so the hypothesis $r_1 \sqsubseteq_{\overline{\approx}_{\text{CLT}}} r_2$ implies that $\mathsf{0} \not\text{MUST}\ r_1$. Definition 3.1.1 implies that $r_1 \parallel \mathsf{0}$ performs a maximal computation which is not client-successful. The process $\mathsf{0}$ cannot interact with r_1, thus the computation is due either to (a) a divergence of r_1, or (b) a reduction sequence $r_1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark} r'_1 \stackrel{\tau}{\not\longrightarrow}$, for some r'_1. The hypothesis $r_1 \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon$ implies that $r_1 \Downarrow^{\checkmark}$, so the maximal computation at hand cannot be due a divergence of r_1, for otherwise it would be client-successful. It follows that (b) above is true, and so that $S(r_1) \in \text{ACC}_{\not\checkmark}(r_1, \varepsilon)$.

Inductive case ($|s| = n + 1$) In this case $s = \alpha s'$, so we know that

(a) $r_1 \ \mathsf{usbl}_{\not\checkmark} \ \alpha s'$,

(b) $\text{ACC}_{\not\checkmark}(r_2, \alpha s') \neq \emptyset$.

We have to prove that $\text{ACC}_{\not\checkmark}(r_1, \alpha s') \neq \emptyset$.

The hypothesis $\text{ACC}_{\not\checkmark}(r_2, \alpha s') \neq \emptyset$ implies that $r_2 \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$; the hypothesis $r_1 \sqsubseteq_{\overline{\approx}_{\text{CLT}}} r_2$, $r_1 \ \mathsf{usbl}_{\not\checkmark} \ \alpha s'$. Lemma 4.2.22 imply the ensuing statements

(1) $\bigoplus(r_1 \ \text{AFTER}_{\not\checkmark} \ \alpha) \sqsubseteq_{\overline{\approx}_{\text{CLT}}} \bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha)$

(2) $\bigoplus(r_1 \ \text{AFTER}_{\not\checkmark} \ \alpha) \ \mathsf{usbl}_{\not\checkmark} \ s'$

As s' is shorter than s, the inductive hypothesis guarantees that

$$\text{ACC}_{\not\checkmark}(\bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha), s') \neq \emptyset \text{ implies } \text{ACC}_{\not\checkmark}(\bigoplus(r_1 \ \text{AFTER}_{\not\checkmark} \ \alpha), s') \neq \emptyset \tag{4.13}$$

We show that the premises of the implication in (4.13) are true: $\text{ACC}_{\not\checkmark}(\bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha), s') \neq \emptyset$. This fact follows from the hypothesis that $\text{ACC}_{\not\checkmark}(r_2, s) \neq \emptyset$, (b), and the equality

$$\text{ACC}_{\not\checkmark}(r_2, \alpha s') = \text{ACC}_{\not\checkmark}(\bigoplus(r_2 \ \text{AFTER}_{\not\checkmark} \ \alpha), s')$$

Now the implication in (4.13) implies that $\text{ACC}_{\not\checkmark}(\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha), s') \neq \emptyset$, so the equality

$$\text{ACC}_{\not\checkmark}(r_1, \alpha s') = \text{ACC}_{\not\checkmark}(\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} \alpha), s')$$

ensures that $\text{ACC}_{\not\checkmark}(r_1, \alpha s') \neq \emptyset$. □

Lemma 4.2.27 uses Lemma 4.2.22, which requires a divergent term, τ^∞, to be in the LTS. Lemma 4.2.27, though, can be proven without using Lemma 4.2.22 and a divergent term τ^∞. The alternative proof relies on the property that (unsuccessful) acceptance sets contain the ready sets of stable states; we have used this fact in the base case of the proof, but not in the inductive case. Also the next lemma can be strengthened without requiring τ^∞ in the LTS.

Lemma 4.2.28. Suppose $r_1 \sqsubseteq_{\text{CLT}} r_2$. For every $s \in Act^\star$, if $r_1 \ \mathsf{usbl}_{\not\checkmark}\ s$ then for every $B \in \text{ACC}_{\not\checkmark}(r_2, s)$ there exists some $A \in \text{ACC}_{\not\checkmark}(r_1, s)$ such that $A \cap \mathsf{ua}_{\not\checkmark}(p, s) \subseteq B$.

Proof. Fix a string s such that $B \in \text{ACC}_{\not\checkmark}(r_2, s)$ for some set B, $r_1 \ \mathsf{usbl}_{\not\checkmark}\ s$, and let $s = \beta_1 \dots \beta_n$. The hypothesis let us use Lemma 4.2.27, which ensures that $\text{ACC}_{\not\checkmark}(r_1, s) = \{ A_i \mid i \in I \}$ for some non-empty set I. This implies that $r_1 \stackrel{s}{\Longrightarrow}_{\not\checkmark} r_1' \stackrel{\tau}{\not\longrightarrow}$, and so $r_1 \stackrel{s}{\Longrightarrow}$.

We reason by contradiction: suppose that

$$\text{for every } i \in I, \text{ the set } A_i \cap \mathsf{ua}_{\not\checkmark}(p, s) \text{ contains an action } \alpha_i \notin B \tag{4.14}$$

In the rest of the proof we use this supposition to define a process P such that

(a) $P \not\text{MUST}\ r_2$, and

(b) $P \text{ MUST } r_1$

For every $i \in I$, the assumption that $\alpha_i \in A_i \cap \mathsf{ua}_{\not\checkmark}(p, s)$ and Definition 4.2.16 ensure that there exists a process $\hat{p}_i$ such that

$$\hat{p}_i \text{ MUST } \bigoplus(r_1 \text{ AFTER}_{\not\checkmark} s\alpha_i) \tag{4.15}$$

Moreover, as $r_1 \ \mathsf{usbl}_{\not\checkmark}\ s$ and $r_1 \stackrel{s}{\Longrightarrow}_{\not\checkmark}$, for every $0 \leq k \leq n$ Lemma 4.2.15 implies that there exists a process p_k such that

$$p_k \text{ MUST } \bigoplus(r_1 \text{ AFTER}_{\not\checkmark} s_k) \tag{4.16}$$

where s_k is the prefix of s with length k. We are ready to define the process P. For $0 \leq k \leq n$ let

$$P_k \stackrel{\mathsf{def}}{=} \begin{cases} p_k + \overline{\beta}_{k+1}.P_{k+1} & \text{if } k < n \\ \sum_{i \in I} \overline{\alpha}_i.\hat{p}_i & \text{if } k = n \end{cases}$$

The P we are after is P_0.

(a) We prove that $P_0 \not\text{MUST}\ r_2$. It suffices to exhibit a maximal computation of $r_2 \parallel P_0$ that is not client-successful. Since $B \in \text{ACC}_{\not\checkmark}(r_2, s)$, there exists a derivative r_2' of r_2 such that $r_2 \stackrel{s}{\Longrightarrow}_{\not\checkmark} r_2' \stackrel{\tau}{\not\longrightarrow}$; by zipping this transition sequence with $P_0 \stackrel{\overline{s}}{\Longrightarrow} P_{n+1}$ we obtain the computation

$$r_1 \parallel P_0 \Longrightarrow r_2' \parallel P_n$$

As all the (co)actions offered by P_n are not in B, and both P_n and r_2' are stable, so it follows that $r_2' \parallel P_n \stackrel{\tau}{\not\longrightarrow}$, thus the computation is maximal and not client-successful. We have proven (a), that is $A_0 \not\text{MUST}\ r_2$.

(b) Now we show that $P_0 \text{ MUST } r_1$. Definition 3.1.1 requires us to show that all the maximal computations of $r_1 \parallel P_0$ are client-successful. Fix a maximal computation of $r_1 \parallel P_0$.

Figure 4.10: Tests to distinguish clients (see Lemma 4.2.28)

The contributions of r_1 and P_0 being with a possibly empty prefix of s (resp. $\overline{s}$), say

$$r_1 \stackrel{s_j}{\Longrightarrow} r', \qquad P_0 \stackrel{\overline{s}_j}{\Longrightarrow} P_j$$

Let s_j be the longest prefix of s that in the computation at hand leads P_0 to a P_i.

The state P_j converges. For suppose P_j diverges. Then $j < n$, $P_j = p_j + \overline{\beta}_{j+1}.P_{j+1}$, and p_j diverges. (4.16) ensures that p_j MUST $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} s_j)$. Pick a $r \in (r_1 \text{ AFTER}_{\not\checkmark} s_j)$, Definition 4.2.12 implies that $r \stackrel{\checkmark}{\not\longrightarrow}$, so we can show a maximal computation of $r \parallel p_j$ which is not client-successful. This contradicts p_j MUST $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} s_j)$. Since P_j converges, so does p_j, and the action sequence $P_0 \stackrel{\overline{s}_j}{\Longrightarrow} P_j$ can be extended to $P_0 \stackrel{\overline{s}_j}{\Longrightarrow} P_j \Longrightarrow P'$, where $P' \stackrel{\tau}{\not\longrightarrow}$ (where P' may be P_j itself, for instance if $j = n$).

If there is a successful state in $r_1 \stackrel{s_j}{\Longrightarrow} r'$ then the computation is client-successful. Suppose that $r_1 \stackrel{s_j}{\Longrightarrow}_{\not\checkmark} r'$. If r' diverges, then it converges to success, because of the hypothesis r_1 $\mathsf{usbl}_{\not\checkmark}$ s and Lemma 4.2.21, so the computation is client-successful. Let us suppose that r' converges, and that it is stable.[2] As the computation is maximal it contains the state $r' \parallel P'$.

If $i = n$, then $P' = P_n$, and as r' is stable, $S(r') \in \text{ACC}_{\not\checkmark}(r_1, s)$. The set $S(r')$ contains and action α_i such that $P_n \stackrel{\overline{\alpha}_i}{\longrightarrow}$. It follows that the computation contains a reduction $r' \parallel P' \stackrel{\tau}{\longrightarrow} r'' \parallel \hat{p}_i$; either $r'' \stackrel{\checkmark}{\longrightarrow}$, or, in view of (4.15) $\hat{p}_i$ MUST r''. In both cases the computation is client-successful.

If $i < n$, then $P' = p'_j + \overline{\beta}_{j+1}.P_{j+1}$. Since $r' \stackrel{\checkmark}{\not\longrightarrow}$, the state $r' \parallel P'$ is not stable, for otherwise p'_j MUST r', which implies that p_j MUST $\bigoplus(r_1 \text{ AFTER}_{\not\checkmark} s_j)$. It follows that $r' \parallel P' \stackrel{\tau}{\longrightarrow} r'' \parallel P''$; as both r' and P' are stable, this reduction is due to an interaction: $p'_j \stackrel{\delta}{\longrightarrow} P''$ for some $\delta \in Act$. Our assumption on s_j ensures that the reduction cannot be due to $\overline{\beta}_{j+1}.P_{j+1}$, so it is due to p'_k. If $r'' \stackrel{\checkmark}{\longrightarrow}$ the computation is client-successful. If $r'' \stackrel{\checkmark}{\not\longrightarrow}$, then $r_1 \stackrel{s_j}{\Longrightarrow} r''$ and (4.16) imply that P'' MUST r'', so the computation we unzipped is client-successful.

We have proven that the maximal computation of $r_1 \parallel P_0$ are client-successful, so P_0 MUST r_1.

By assuming the thesis false (see (4.14)), we have defined a process P which lets us contradict the hypothesis $r_1 \sqsubseteq_{\text{CLT}} r_2$; it follows that the thesis of the lemma is true. □

In the previous lemma the server P is defined using an external sum. This is necessary; we explain why in the next example.

Example 4.2.29. In Lemma 4.2.28 (resp. Figure 4.10) the test used to distinguish two clients is defined using an external sum. This contrasts with Lemma 4.1.15 (resp. Figure 4.3), in which the test used to distinguish servers is defined by an internal sum. If we defined the server P in Lemma 4.2.28 using an internal sum then the proof would not work. In this example we explain why.

[2]Since r' converges, if it is not stable, the computation contains a stable state r'' such that $r' \Longrightarrow r''$. If r' is not stable we use r'' in place of r'.

Consider the clients $r_1 = \beta.(1 \oplus 1) \oplus \alpha.\gamma.1$ and $r_2 = \alpha.\gamma.1$. One can prove that $r_1 \sqsubseteq_{\text{CLT}} r_2$; this is true because $r_1 \Longrightarrow_{\not\checkmark} r_2$. Moreover note that r_1 $\mathsf{usbl}_{\not\checkmark}$ a and $\{c\} \in \text{ACC}_{\not\checkmark}(r_2, a)$, so the clients r_1 and r_2 satisfy the hypothesis of Lemma 4.2.28.

Observe how P_0 is defined in Lemma 4.2.28, and replace the external sum with an internal sum,

$$P_0' = p \oplus \overline{\alpha}.P_1$$

where p MUST r_1. It is not true that P_0' MUST r_1, so one of they key steps of the proof fails. To see why $P_0' \not\text{MUST}\ r_1$, note that we can infer the computation

$$r_1 \mid\mid P_0 \xrightarrow{\tau} r_1 \mid\mid \overline{\alpha}.P_1 \xrightarrow{\tau} \beta.(1 \oplus 1) \mid\mid \overline{\alpha}.P_1 \xrightarrow{\tau}\!\!\!\!\!\!/\ \ \ $$

This computation is maximal and not client-successful. □

We have gathered enough results to state the alternative characterisation of $\sqsubseteq_{\text{CLT}}$.

Definition 4.2.30. [Semantic MUST client pre-order]
Let $r_1 \precsim_{\text{CLT}} r_2$ if

(1) for every $s \in Act^\star$ such that r_1 $\mathsf{usbl}_{\not\checkmark}$ s,

  (a) r_2 $\mathsf{usbl}_{\not\checkmark}$ s

  (b) for every $B \in \text{ACC}_{\not\checkmark}(r_2, s)$ there exists some $A \in \text{ACC}_{\not\checkmark}(r_1, s)$ such that

$$A \cap \mathsf{ua}_{\not\checkmark}(r_1, s) \subseteq B$$

(2) for every $w \in Act^\star \cup Act^\infty$ such that r_1 $\mathsf{usbl}_{\not\checkmark}$ w, $r_2 \overset{w}{\Longrightarrow}_{\not\checkmark}$ implies $r_1 \overset{w}{\Longrightarrow}_{\not\checkmark}$. □

We obtain the completeness of $\precsim_{\text{CLT}}$ immediately.

Proposition 4.2.31. [Completeness]
If $r_1 \sqsubseteq_{\text{CLT}} r_2$ then $r_1 \precsim_{\text{CLT}} r_2$.

Proof. This is true because of Lemma 4.2.23, Lemma 4.2.28, Lemma 4.2.24, and Corollary 4.2.25. □

Unsuccessful acceptance sets contain the ready sets only of stable states; we have motivated this in Example 4.2.10. This property of unsuccessful acceptance sets implies that in Definition 4.2.30 condition (1b) and condition (2) are independent, so we have to require both of them.

Example 4.2.32. First, observe that $1 \xrightarrow{\checkmark}$, so $1 \overset{\varepsilon}{\not\Longrightarrow}_{\not\checkmark}$; whereas $1 \oplus 1 \overset{\varepsilon}{\Longrightarrow}_{\not\checkmark}$, because

$$\frac{}{1 \oplus 1 \overset{\varepsilon}{\Longrightarrow}_{\not\checkmark} 1 \oplus 1}\ 1 \oplus 1 \xrightarrow{\checkmark}\!\!\!\!\!\!/\ \ \ ;\ [\text{UT-AX}]$$

This difference between 1 and $1 \oplus 1$ is essentially the reason why $1 \not\approx_{\text{CLT}} 1 \oplus 1$ (see Example 4.2.4).

Without point (2) in Definition 4.2.30, the relation $\precsim_{\text{CLT}}$ would not be a sound with respect to $\sqsubseteq_{\text{CLT}}$. Let $\preccurlyeq'_{\mathsf{bad}}$ be defined as Definition 4.2.30 but omitting point (2). One can prove that $1 \preccurlyeq'_{\mathsf{bad}} 1 \oplus 1$. This follows from the fact that for every $s \in Act^\star$ the set $\text{ACC}_{\not\checkmark}(1 \oplus 1, s)$ is empty, and $1 \oplus 1$ $\mathsf{usbl}_{\not\checkmark}$ s. It follows that the relation $\preccurlyeq'_{\mathsf{bad}}$ is not sound: $\preccurlyeq'_{\mathsf{bad}} \not\subseteq \sqsubseteq_{\text{CLT}}$.

Similar to point (2), also point (1b) is necessary to obtain soundness. Define $\preccurlyeq'_{\mathsf{bad}}$ according to Definition 4.2.30, but omitting point (1b), and let $r_1 = \alpha.(1 \oplus 1)$. One can prove that $r_1 \preccurlyeq'_{\mathsf{bad}}$ $\alpha.\beta.1$. Let us see why. The only unsuccessful traces of $\alpha.\beta.1$ are ε and α ; the client r_1 $\mathsf{usbl}_{\not\checkmark}$ ε, α, moreover $r_1 \overset{\varepsilon}{\Longrightarrow}_{\not\checkmark}$ and $r_1 \overset{\alpha}{\Longrightarrow}_{\not\checkmark}$ so $r_1 \preccurlyeq'_{\mathsf{bad}}$ $\alpha.\beta.1$. Note now that $r_1 \not\approx_{\text{CLT}}$ $\alpha.\beta.1$; this being true because $\overline{\alpha}.0$ MUST r_1 while $\overline{\alpha}.0 \not\text{MUST}\ \alpha.\beta.1$. □

Example 4.2.33. The predicate $\mathsf{usbl}_{\not\checkmark}$ enforces convergence only along unsuccessful traces; if it required all the traces to converge, as the original $\Downarrow$ does, then Definition 4.2.30 would not be complete. Let r denote $\alpha.(\beta.\mathbf{1} \oplus (\mathbf{1} + \tau^\infty))$, and observe that $\alpha.\beta.\mathbf{1} \sqsubsetneq_{\text{CLT}} r$; this is because if the right-hand term diverges, then it reaches a successful state. We cannot prove $\alpha.\beta.\mathbf{1} \precsim_{\text{CLT}} r$ by using $\Downarrow$ in place of $\mathsf{usbl}_{\not\checkmark}$, as $\alpha.\beta.\mathbf{1} \Downarrow a$, but $r \not\Downarrow \alpha$ because $(r \text{ AFTER}_{\not\checkmark} \alpha) \not\Downarrow$. The predicate $\mathsf{usbl}_{\not\checkmark}$, on the contrary, is insensible to the divergence of $(r \text{ AFTER}_{\not\checkmark} \alpha)$, so one can prove both $\alpha.\beta.\mathbf{1}\ \mathsf{usbl}_{\not\checkmark}\ \alpha$ and $r\ \mathsf{usbl}_{\not\checkmark}\ \alpha$. □

Example 4.2.34. Let us revisit the clients r_1, r_2, in Example 4.2.19. The client $\beta.\delta.\mathbf{0} + \beta.\mathbf{1}$ is not usable; that is $\beta.\delta.\mathbf{0} + \beta.\mathbf{1} \notin \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ because it cannot be satisfied by any server. Consequently $r_1\ \mathsf{usbl}_{\not\checkmark}\ \alpha\gamma$ does not hold, and therefore when checking whether $r_1 \precsim_{\text{CLT}} r_2$ the set $\{\delta\} \in \text{ACC}_{\not\checkmark}(r_2, \alpha\gamma)$ does not have to be matched by r_1.

Indeed it is now straightforward to check that $r_1 \precsim_{\text{CLT}} r_2$; the only $s \in Act^*$ for which $\text{ACC}_{\not\checkmark}(r_2, s)$ is non-empty and $r_1\ \mathsf{usbl}_{\not\checkmark}\ s$ is the empty sequence ϵ.

The use of the predicate $\mathsf{usbl}_{\not\checkmark}\ s$ in Definition 4.2.30 is very strong. As an example consider again the client $r = \alpha.(\beta.\mathbf{0} + \gamma.\mathbf{1}) + \alpha.(\beta.\mathbf{1} + \gamma.\mathbf{0})$. Note that $r \notin \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ as there is no server which can satisfy it. Consequently $r\ \mathsf{usbl}_{\not\checkmark}\ s$ is false for every trace s, from which it follows that $r \precsim_{\text{CLT}} r'$ for any other client r'. □

We prove two properties of the predicate $\mathsf{usbl}_{\not\checkmark}$ that we need.

Lemma 4.2.35. Let p MUST r.

(i) If $p \stackrel{\overline{s}}{\Longrightarrow}$, then $r\ \mathsf{usbl}_{\not\checkmark}\ s$.

(ii) If $p \stackrel{\overline{u}}{\Longrightarrow}$, then $r\ \mathsf{usbl}_{\not\checkmark}\ u$.

Proof. First note that the hypothesis p MUST r ensures that $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$.

The proof of point (i) is by induction on the length of s. In the base case $\mathsf{len}(s) = 0$, so $s = \varepsilon$ and we have to prove that $r\ \mathsf{usbl}_{\not\checkmark}\ \varepsilon$; the following derivation suffices

$$\frac{}{r\ \mathsf{usbl}_{\not\checkmark}\ \varepsilon}\ r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}\ [\text{CCONV-AX}]$$

In the inductive case $\mathsf{len}(s) = n + 1$, so $s = \alpha s'$ and we have to prove that $r\ \mathsf{usbl}_{\not\checkmark}\ \alpha s'$. If $r \stackrel{\alpha}{\not\Longrightarrow}_{\not\checkmark}$ then we can derive

$$\frac{}{r\ \mathsf{usbl}_{\not\checkmark}\ \alpha s'}\ r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}, r \stackrel{\alpha}{\not\Longrightarrow}_{\not\checkmark}\ [\text{CCONV-AX}]$$

If $r \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$ then Definition 4.2.12 ensures that $(r \text{ AFTER}_{\not\checkmark} \alpha)$ is non-empty. The hypothesis $p \stackrel{\overline{s}}{\Longrightarrow}$ implies that $p \stackrel{\alpha}{\Longrightarrow} p'$ for some p'. The hypothesis p MUST r and the assumption $r \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$ imply that p' MUST $\bigoplus(r \text{ AFTER}_{\not\checkmark} \alpha)$. Since s' is shorter than s, the inductive hypothesis ensures that $\bigoplus(r \text{ AFTER}_{\not\checkmark} \alpha)\ \mathsf{usbl}_{\not\checkmark}\ s'$. Now we can derive

$$\frac{\dfrac{\vdots}{\bigoplus(r \text{ AFTER}_{\not\checkmark} \alpha)\ \mathsf{usbl}_{\not\checkmark}\ s'}}{r\ \mathsf{usbl}_{\not\checkmark}\ \alpha s'}\ r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}, r \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}\ [\text{CCONV-ALPHA}]$$

We have proven point (i); now we explain why point (ii) is true. Suppose that $p \stackrel{\overline{u}}{\Longrightarrow}$; we have to show that $r\ \mathsf{usbl}_{\not\checkmark}\ u$. Definition 4.2.14 requires us to prove that for every $n \in N$, $r\ \mathsf{usbl}_{\not\checkmark}\ u_n$. Fix a number $n \in \mathbb{N}$; the hypothesis $p \stackrel{\overline{u}}{\Longrightarrow}$ implies that $p \stackrel{\overline{u_n}}{\Longrightarrow}$. Since $u_n \in Act^\star$, point (i) implies that $r\ \mathsf{usbl}_{\not\checkmark}\ u_n$. We have no assumption on n, hence we have proven that for every finite s prefix of the string u, $r\ \mathsf{usbl}_{\not\checkmark}\ s$. □

Lemma 4.2.36. For every $s \in Act^\star$, if $r\ \mathsf{usbl}_{\not\checkmark}\ s$ and $r \stackrel{s}{\Longrightarrow}_{\not\checkmark} r'$, then $r'\ \mathsf{usbl}_{\not\checkmark}\ \varepsilon$.

Proof. The argument is by induction on the length of s.

Base case ($|s| = 0$) In this case $s = \varepsilon$. By hypothesis $r \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark} r'$, so the definition of $\Longrightarrow_{\not\checkmark}$ implies that is $r \stackrel{\tau}{\longrightarrow}^{*} r'$, and none of the states in the reduction sequence is successful (see Proposition 4.2.7).

We have to prove that r' $\mathsf{usbl}_{\not\checkmark}$ ε. If $r = r'$ then this is trivially true, otherwise it suffices to show that $r' \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$, as this allows us to infer

$$\frac{}{r' \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon} \ r' \in \mathcal{U}_{\text{CLT}}^{\text{MUST}} \ [\text{CCONV-AX}]$$

We prove that $r' \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$. By hypothesis r $\mathsf{usbl}_{\not\checkmark}$ ε, so $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$. It follows that there exists a p such that p MUST r. We prove that p MUST r'. Fix a maximal computation of $r' \mid\mid p$; since $r \stackrel{\tau}{\longrightarrow}^{*} r'$ the chosen computation is a suffix of a maximal computation of $r \mid\mid p$. The computation of $r \mid\mid p$ is client-successful: there exists a state $\hat{r} \mid\mid \hat{p}$ wherein $\hat{r} \stackrel{\checkmark}{\longrightarrow}$. As no state along the reductions $r \stackrel{\tau}{\longrightarrow}^{*} r'$ is successful, the state $\hat{r} \mid\mid \hat{p}$ must appear after $r' \mid\mid p$, that is

$$r \mid\mid p \Longrightarrow r' \mid\mid p \Longrightarrow \hat{r} \mid\mid \hat{p}$$

It follows that the computation of $r' \mid\mid p$ is client-successful. As this argument applies to all the maximal computations of $r' \mid\mid p$, we have proven that p MUST r'. It follows that $r' \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$.

Inductive case ($|s| = n + 1$) In this case $s = \alpha s'$ for some $s' \in Act^{\star}$. The hypothesis $r \stackrel{\alpha s'}{\Longrightarrow}_{\not\checkmark} r'$ implies that $r \stackrel{\alpha}{\Longrightarrow}_{\not\checkmark}$, and so the hypothesis r $\mathsf{usbl}_{\not\checkmark}$ s now implies that $\bigoplus(r \text{ AFTER}_{\not\checkmark} \alpha)$ $\mathsf{usbl}_{\not\checkmark}$ s'. As s' is shorter than s we can use the inductive hypothesis:

$$\text{if } \bigoplus(r \text{ AFTER}_{\not\checkmark} \alpha) \stackrel{s'}{\Longrightarrow}_{\not\checkmark} r'' \text{ then } r'' \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon.$$

The hypothesis $r \stackrel{\alpha s'}{\Longrightarrow}_{\not\checkmark} r'$ implies that $\bigoplus(r \text{ AFTER}_{\not\checkmark} \alpha) \stackrel{s'}{\Longrightarrow}_{\not\checkmark} r'$, thus $r' \ \mathsf{usbl}_{\not\checkmark} \ \varepsilon$.

□

We are ready to prove the chief result of this section.

Theorem 4.2.37. [Alternative characterisation $\sqsubseteq_{\text{CLT}}$]
For every $r_1, r_2 \in \mathsf{CCS}_{\text{w}\tau}$, $r_1 \sqsubseteq_{\text{CLT}} r_2$ if and only if $r_1 \precsim_{\text{CLT}} r_2$.

Proof. We are required to prove two implications, namely

(i) if $r_1 \sqsubseteq_{\text{CLT}} r_2$ then $r_1 \precsim_{\text{CLT}} r_2$

(ii) if $r_1 \precsim_{\text{CLT}} r_2$ then $r_1 \sqsubseteq_{\text{CLT}} r_2$

The first implication is Proposition 4.2.31, so we discuss only the second implication.

Fix a pair $r_1 \precsim_{\text{CLT}} r_2$, and let p MUST r_1; we have to show that all the maximal computations of the composition $r_2 \mid\mid p$ are client-successful. Fix such a computation,

$$r_2 \mid\mid p = r_2^0 \mid\mid p^0 \stackrel{\tau}{\longrightarrow} r_2^1 \mid\mid p^1 \stackrel{\tau}{\longrightarrow} r_2^3 \mid\mid p^3 \stackrel{\tau}{\longrightarrow} r_2^4 \mid\mid p^4 \stackrel{\tau}{\longrightarrow} \ldots \tag{4.17}$$

The computation in (4.17) is finite or infinite. We discuss the two cases separately.

Suppose that the computation is finite, and unzip it; the resulting contributions of p and r_2 are

$$r_2 \stackrel{s}{\Longrightarrow} r_2^k, \qquad p \stackrel{\overline{s}}{\Longrightarrow} p^k$$

for some $s \in Act^{\star}$, and stable $r_2^k \mid\mid p^k$. The hypothesis p MUST r_1, $p \stackrel{\overline{s}}{\Longrightarrow}$, and Lemma 4.2.35 imply that r_1 $\mathsf{usbl}_{\not\checkmark}$ s. The argument is by contradiction: suppose that no state in the contribution of r_2 reports success. It follows $r_2 \stackrel{s}{\Longrightarrow}_{\not\checkmark} r_2^k$, and as $r_2^k \stackrel{\tau}{\not\longrightarrow}$, $S(r_2^k) \in \text{ACC}_{\not\checkmark}(r_2, s)$; so point (1b) of Definition 4.2.30 implies that $A \in \text{ACC}_{\not\checkmark}(r_1, s)$, for some A such that $A \cap \mathsf{ua}_{\not\checkmark}(r_1, s) \subseteq S(r_2^k)$. Definition 4.2.8 implies that there exists a r_1' such that $S(r_1') = A$ and $r_1 \stackrel{s}{\Longrightarrow}_{\not\checkmark} r_1' \stackrel{\tau}{\not\longrightarrow}$. Zip together

the contributions along s of p and r_1, the resulting computation reaches the state $r_1' \parallel p_k$; if this state is terminal, then the computation is maximal and *not* client-successful, so $p \not\,\textsc{must}\, r_1$. This contradicts our assumption that p MUST r_1.

Terminal state To prove the contradiction just described we have to show that $r_1^k \parallel p_k$ is stable; both p_k and r_1^k are stable, so we show only that if $p_k \xrightarrow{\overline{\alpha}}$ then $\alpha \notin S(r_1')$. To this end, we partition the set $S(r_1')$ according to the usability of the actions in it; let

- $U = S(r_1') \cap \mathsf{ua}_{\not\downarrow}(r_1, s)$ be the partition of usable actions
- $N = S(r_1') \setminus \mathsf{ua}_{\not\downarrow}(r_1, s)$ be the partition of unusable actions.

We prove that if $p_k \xrightarrow{\overline{\alpha}}$ then $\alpha \notin N$ and $\alpha \notin U$. Suppose that $p_k \xrightarrow{\overline{\alpha}}$.

- Since $p \overset{\overline{s}}{\Longrightarrow} p_k$, the assumption $p_k \xrightarrow{\overline{\alpha}}$ implies that $p \overset{\overline{s\alpha}}{\Longrightarrow}$. The assumption that p MUST r_1 and Lemma 4.2.15 imply that $r_1\ \mathsf{usbl}_{\not\downarrow}\ s\alpha$. If $r_1 \overset{s\alpha}{\Longrightarrow}_{\not\downarrow}$, then Definition 4.2.16 implies that $\alpha \in \mathsf{ua}_{\not\downarrow}(r_1, s)$. If $r_1^k \xrightarrow{\alpha}$ then $\alpha \in \mathsf{ua}_{\not\downarrow}(r_1, s)$, so $\alpha \notin N$; if $r_1^k \overset{\alpha}{\not\longrightarrow}$ then $\alpha \notin N$. We have proven that $\alpha \notin \mathbb{N}$
- The stability if $p_k \parallel r_2^k$, the set inclusion $A \cap \mathsf{ua}_{\not\downarrow}(r_1, s) \subseteq S(r_2')$, and the equality $S(r_1') = A$ imply that $\alpha \notin U$.

It follows that the state $r_1' \parallel p_k$ is stable.

We have discussed the case of a finite maximal computation of $r_2 \parallel r_p$. We turn our attention the argument for the infinite computations.

Suppose that the computation in (4.17) is infinite, and unzip it. Either p and r_2 perform infinite traces, or they perform finite traces and then (at least) one of them diverge.

If we are in the first case, then

$$r_2 \overset{u}{\Longrightarrow}, \qquad p \overset{\overline{u}}{\Longrightarrow}$$

The assumption p MUST r_1, the fact that $p \overset{\overline{u}}{\Longrightarrow}$, and Lemma 4.2.35 imply that $r_1\ \mathsf{usbl}_{\not\downarrow}\ u$. The proof that there is a successful term in $r_2 \overset{u}{\Longrightarrow}$ is by contradiction; for suppose that $r_2 \overset{u}{\Longrightarrow}_{\not\downarrow}$; then point (2) of Definition 4.2.30 implies that $r_1 \overset{u}{\Longrightarrow}_{\not\downarrow}$. By zipping $r_1 \overset{u}{\Longrightarrow}_{\not\downarrow}$ with $p \overset{\overline{u}}{\Longrightarrow}$ we obtain a maximal computation of $r_1 \parallel p$ which is not client-successful; this implies that $p \not\,\textsc{must}\, r_1$, which contradicts our original assumption on p.

Suppose now that p and r_2 engage in finite traces and then there is a divergence; by unzipping the computation in (4.17) we get the contributions

$$r_2 \overset{s}{\Longrightarrow} r_2^k, \qquad p \overset{\overline{s}}{\Longrightarrow} p^k$$

The assumption p MUST r_1, the fact that $p \overset{\overline{s}}{\Longrightarrow}$, and Lemma 4.2.35 imply that $r_1\ \mathsf{usbl}_{\not\downarrow}\ s$. Either p^k diverge or r_2^k diverge, or both diverge.

Suppose that p_k diverges. To prove that the computation in (4.17) is client-successful we reason by contradiction: suppose that there is no successful state among $r_2, \ldots, r_2^k$; this implies that r_2 performs the trace s unsuccessfully,

$$r_2 \overset{s}{\Longrightarrow}_{\not\downarrow}$$

Point (2) of Definition 4.2.30 ensures that $r_1 \overset{s}{\Longrightarrow}_{\not\downarrow} r_1'$. We zip the contribution of p with the unsuccessful transition of r_1; as p_k diverges the resulting computation is maximal,

$$p \parallel r_1 \Longrightarrow p_k \parallel r_1' \Longrightarrow p_k \parallel r_1' \Longrightarrow \ldots \tag{4.18}$$

All the derivatives of r_1 in the maximal computation above are in $r_1 \stackrel{s}{\Longrightarrow}\not\checkmark r_1'$, so they are not successful; it follows that the computation in Eq. (4.18) is not client-successful. This proves that $p \not\text{MUST}\ r_1$. As this contradicts our assumption on p, it follows that one of the states in $r_2, \ldots, r_2^k$ is successful.

Suppose that r_2^k diverges. If there is a successful state in $r_2 \stackrel{s}{\Longrightarrow} r_2^k$ then the maximal computation we unzipped is client-successful. Suppose that there is no successful state in the contribution of r_2, that is $r_2 \stackrel{s}{\Longrightarrow}\not\checkmark r_2^k$. As $r_1\ \mathsf{usbl}\not\checkmark\ s$, point (1a) of Definition 4.2.30 implies that $r_2\ \mathsf{usbl}\not\checkmark\ s$. Lemma 4.2.36 ensures that $r_2^k\ \mathsf{usbl}\not\checkmark\ \varepsilon$, and the definition of $\mathsf{usbl}\not\checkmark$ ensures that r_2^k converges to success (Definition 4.2.20), thus after r_2^k there is a successful state in the contribution of r_2^k. □

In Section 4.1 and in this section we have studied the pre-orders that arise from the MUST relation in a client/server environment. One pre-order $\sqsubseteq_{\text{SVR}}$, states when a server satisfies more clients than another server; the other pre-order, $\sqsubseteq_{\text{CLT}}$, establishes when a client is satisfied by more servers than another client. Theorem 4.1.21 and Theorem 4.2.37 are the chief results of our study; those theorems provide proof methods for the server and the client pre-orders. They have have a similar form, with Theorem 4.2.37 showing explicitly the role played by the notion of usable term. This does not appear in Theorem 4.2.37 because every server is usable (i.e. satisfies some client), and so are the actions performed by server.

In the next section we will carry out a work similar to what done so far, but in a peer to peer setting rather than a client/server setting.

4.3 Peer pre-order

So far we have studied the pre-orders given by the relation MUST. Since the relation is not symmetric, it gives rise to two natural pre-orders, one for the servers and one for clients. The investigation we have carried out has shown that servers are compared assessing the amount of interaction that they offer to the environment; while clients are compared assessing the interactions that they require to be satisfied.

In this section we move from a client/server setting to a peer to peer setting. Roughly speaking, this means that now our definitions (and our results) are no longer biased towards one side of the compositions $r \mid\mid p$.

Recall Definition 3.1.3.

Definition 4.3.1. [MUST-peer-pre-order]
We write $p \sqsubseteq_{\text{P2P}} q$ if and only if $p\ \text{MUST}^{\text{P2P}}\ r$ implies $q\ \text{MUST}^{\text{P2P}}\ r$ for every process r. We refer to the relation $\sqsubseteq_{\text{P2P}}$ as MUST *peer pre-order* . □

Notation Similarly what done for $\sqsubseteq_{\text{P2P}}$, also to reason on $\sqsubseteq_{\text{CLT}}$ we are free to use the general summations $\sum$ and $\bigoplus$. This is by justified to the fact that $\eqsim_{\text{P2P}}$ is commutative and associative with respect to $\oplus$ and $\oplus$, where $\eqsim_{\text{P2P}}$ is the equivalence relation generated in the obvious way from $\sqsubseteq_{\text{P2P}}$.

Since MUST^{P2P} is the symmetric version of MUST, one would expect that $\sqsubseteq_{\text{P2P}} = \sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}}$. This is not true.

Example 4.3.2. In this we prove that $\sqsubseteq_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}}$. It suffices to prove that $\sqsubseteq_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{SVR}}$; to this end we have to exhibit two processes p and q such that $p \sqsubseteq_{\text{P2P}} q$ and $p \not\sqsubseteq_{\text{SVR}} q$.

It is easy to see that $\alpha.0 \sqsubseteq_{\text{P2P}} \beta.0$. This is true because $\alpha.0$ can never be satisfied, for it offers no ✓ at all. However, $\alpha.0 \not\sqsubseteq_{\text{SVR}} \beta.0$, as the client $\overline{\alpha}.1$ is satisfied by $\alpha.0$, whereas $\beta.0 \not\text{MUST}\ \overline{\alpha}.1$. □

The proof of the inclusion $\sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}} \subseteq \sqsubseteq_{\text{P2P}}$ is straightforward.

Lemma 4.3.3. For every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p\ (\sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}})\ q$, then $p \sqsubseteq_{\text{P2P}} q$.

Proof. Fix two processes p and q that satisfy the hypothesis. We have to prove that if p MUST$^{\text{P2P}}$ r then q MUST$^{\text{P2P}}$ r, for every process r. We reason as follows,

p MUST$^{\text{P2P}}$ r	By assumption
p MUST r and r MUST p	Thanks to Definition 3.1.3
q MUST r and r MUST q	Thanks to the hypothesis $p\ (\precsim_{\text{SVR}} \cap \precsim_{\text{CLT}})\ q$
q MUST$^{\text{P2P}}$ r	Thanks to Definition 3.1.3

□

Example 4.3.4. It is necessary to use the intersection of $\precsim_{\text{CLT}}$ and $\precsim_{\text{SVR}}$ to prove the previous lemma. The following inequalities prove this.

$$\alpha.1 \precsim_{\text{SVR}} \alpha.0 \qquad \alpha.1 \not\precsim_{\text{P2P}} \alpha.0$$
$$1+\alpha.0 \precsim_{\text{CLT}} 1+\beta.0 \qquad 1+\alpha.0 \not\precsim_{\text{P2P}} 1+\beta.0$$

We explain why $\alpha.1 \not\precsim_{\text{P2P}} \alpha.0$. This is true because the peers $\alpha.1$ and $\alpha.0$ are distinguished by $\overline{\alpha}.1$: $\alpha.1$ MUST$^{\text{P2P}}$ $\overline{\alpha}.1$, while $\alpha.0 \not\text{MUST}^{\text{P2P}}$ $\overline{\alpha}.1$.

To prove that $1+\alpha.0 \not\precsim_{\text{P2P}} 1+\beta.0$ we use the peer $\overline{\alpha}.1$; it is easy to see that all the maximal computations of $1+\alpha.0 \mid\mid \overline{\alpha}.1$ are successful. On the contrary, no maximal computation of $1+\beta.0 \mid\mid \overline{\alpha}.1$ is successful, as the peers are stable, and $\overline{\alpha}.1$ is not successful. □

The inequalities proven in (4.3.4) imply that the relations Definition 4.1.17 and Definition 4.2.30 do not describe faithfully the peer pre-order. Moreover, Example 4.3.2 shows that neither does the intersection $\precsim_{\text{SVR}} \cap \precsim_{\text{CLT}}$ provide a complete description of $\precsim_{\text{P2P}}$. Our aim in this section is therefore to characterise $\precsim_{\text{P2P}}$; we want to understand under which conditions the behaviours of two processes p and q are related by $\precsim_{\text{P2P}}$.

Since peers are at the same time clients and servers, we expect them to be compared both as clients and as servers. In other words, we expect $\precsim_{\text{P2P}}$ to enjoy some properties of $\precsim_{\text{CLT}}$ and some properties of $\precsim_{\text{SVR}}$. We refer to the first properties as client-properties and to the second properties as server-properties.

The client-properties of the peer pre-order are due to the next result.

Proposition 4.3.5. The peer pre-order is contained in the client pre-order. Formally, $\precsim_{\text{P2P}} \subseteq \precsim_{\text{CLT}}$.

Proof. Fix a pair of processes in the peer pre-order, $r_1 \precsim_{\text{P2P}} r_2$. We are required to prove that $r_1 \precsim_{\text{CLT}} r_2$. Take a process p such that p MUST r_1; Lemma 4.1.23 implies that $p+1$ MUST r_1. Since $p+1 \xrightarrow{\checkmark}$ it follows that r_1 MUST $p+1$, so the assumption that p MUST r_1 and Definition 3.1.3 imply that r_1 MUST$^{\text{P2P}}$ $p+1$. The hypothesis $r_1 \precsim_{\text{P2P}} r_2$ implies that r_2 MUST$^{\text{P2P}}$ $p+1$; in turn this ensures that $p+1$ MUST r_2. Lemma 4.1.23 lets us conclude that p MUST r_2. □

Proposition 4.3.5 ensures that if $p \precsim_{\text{P2P}} q$ then q in the role of client is better than the client p. It follows that Lemma 4.2.23, Lemma 4.2.28, Lemma 4.2.24, and Corollary 4.2.25 are true also for the processes related by $\precsim_{\text{P2P}}$. Note though that in Example 4.3.4 we have seen that $\precsim_{\text{CLT}} \not\subseteq \precsim_{\text{P2P}}$, so the peer pre-order is more demanding than $\precsim_{\text{CLT}}$. In particular, $\precsim_{\text{P2P}}$ compares the processes also as servers.

The remaining work needed to characterise $\precsim_{\text{P2P}}$ amounts in proving the server-properties of that relation. In Example 4.3.2 we have highlighted that $\precsim_{\text{P2P}} \not\subseteq \precsim_{\text{SVR}}$, so the properties required by $\precsim_{\text{SVR}}$ differ from the server-properties guaranteed by $\precsim_{\text{P2P}}$. The set inclusion $\precsim_{\text{SVR}} \cap \precsim_{\text{CLT}} \subseteq \precsim_{\text{P2P}}$ suggests that the server-properties of $\precsim_{\text{P2P}}$ are a relaxed version of the ones required by $\precsim_{\text{SVR}}$. The intuitive reason why $\precsim_{\text{P2P}} \not\subseteq \precsim_{\text{SVR}}$ is the non trivial usability of peers, so we relax $\precsim_{\text{SVR}}$ taking the usability into account.

We define an amended version of $\precsim_{\text{SVR}}$ and of the server convergence predicate $\Downarrow$.

Definition 4.3.6. [Peer MUST convergence along trace]
For every $s \in Act^\star$ we write $p \Downarrow_{\text{P2P}} s$ if and only if $p \Downarrow s$ and $p\ \mathsf{usbl}\not\swarrow\ s$. We extend the predicate $\Downarrow_{\text{P2P}}$ to infinite strings in the obvious way. □

Definition 4.3.7. Let $p \precsim_{u\text{SVR}} q$ whenever

(1) for every $s \in Act^\star$, if $p \Downarrow_{\text{P2P}} s$ then

(a) $q \Downarrow s$

(b) for every $B \in \text{ACC}(q, s)$ there exists some $A \in \text{ACC}(p, s)$ such that $A \cap \mathsf{ua}\not\swarrow(p, s) \subseteq B$

(2) for every $w \in Act^\star \cup Act^\infty$, if $p \Downarrow_{\text{P2P}} w$, and $q \overset{w}{\Longrightarrow}$, then $p \overset{w}{\Longrightarrow}$ □

Definition 4.3.7 takes the usability into account in two ways; first, if $p \precsim_{u\text{SVR}} q$, then requirements of $\precsim_{\text{SVR}}$ after any trace are enforced by $\precsim_{u\text{SVR}}$ only if p is usable after that trace; and second, only the usable actions of p (after a trace) are used to compare ready sets.

In the next lemmas we prove that the relations $\precsim_{u\text{SVR}}$ is a complete description of $\sqsubseteq_{\simeq_{\text{P2P}}}$ (i.e.. $\sqsubseteq_{\simeq_{\text{P2P}}} \subseteq \precsim_{u\text{SVR}}$). Those lemmas (Lemma 4.3.8, Lemma 4.3.11 and Lemma 4.3.13) are indeed analogous to Lemma 4.1.8, Lemma 4.1.9, and Lemma 4.1.15.

Structure of the proofs Intuitively, during unsuccessful execution of traces, peers behave at the same time as clients and servers; whereas after reporting success they behave only as servers. The oncoming proofs witness this intuition, and amounts to a mixture of the proofs we gave in Section 4.1 and Section 4.2. In particular, the peers that prove the completeness of the characterisation of $\sqsubseteq_{\simeq_{\text{P2P}}}$ are build combining the clients and the servers used in the previous sections.

If p is a usable client, then the convergence of p implies the convergence of q.

Lemma 4.3.8. For every $s \in Act^\star$, and every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \sqsubseteq_{\simeq_{\text{P2P}}} q$, $p \Downarrow_{\text{P2P}} s$ and $q \overset{s}{\Longrightarrow} q'$, then $q' \Downarrow$.

Proof. Let $s = \alpha_1\alpha_2 \ldots \alpha_n$. Let s' be the longest prefix of s such that $p \overset{s'}{\Longrightarrow}\not\swarrow$. The proof is divided in two cases, which depend on the existence of s'. In both cases the argument has the same structure; we define a peer C such that

1) p MUST$^{\text{P2P}}$ C, and

2) q MUST$^{\text{P2P}}$ C lets us prove that if $q \overset{s}{\Longrightarrow} q'$ then $q' \Downarrow$

s' does not exist In this case one cannot infer $p \overset{\varepsilon}{\Longrightarrow}\not\swarrow$, thus $p \xrightarrow{\checkmark}$. Let the process C be defined as in Lemma 4.1.8. The proof of that lemma shows that p MUST C. Since $p \xrightarrow{\checkmark}$ we also know that C MUST p, and so Definition 3.1.3 implies that p MUST$^{\text{P2P}}$ P. The hypothesis $p \sqsubseteq_{\simeq_{\text{P2P}}}$ implies that q MUST$^{\text{P2P}}$ P. We prove that if $q \overset{s}{\Longrightarrow} q'$ then q' by reasoning as we did in Lemma 4.1.8.

s' exists In this case $p \overset{s'}{\Longrightarrow}\not\swarrow$ for some s'; let $s' = \alpha_1\alpha_2 \ldots \alpha_m$, with $m \leq n$. For every $0 \leq j \leq m$ the assumption $p \overset{s'}{\Longrightarrow}\not\swarrow$ ensures that $p \overset{s_j}{\Longrightarrow}\not\swarrow$. Lemma 4.2.15 ensures that for every $0 \leq j \leq m$ there exists a $\hat{r}_j$ such that

$$\hat{r}_j \text{ MUST } \bigoplus (p \text{ AFTER}\not\swarrow\ s_j)$$

For every $0 \leq k \leq n+1$ let

$$C_k \overset{\text{def}}{=} \begin{cases} ((\hat{r}_k + 1) \oplus (\hat{r}_k + 1)) + \overline{\alpha}_{k+1}.P_{k+1} & \text{if } 0 \leq k \leq m \\ (1 \oplus 1) + \overline{\alpha}_{k+1}.A_{k+1} & \text{if } m < k \leq n \\ (\hat{r}_n + 1) \oplus (\hat{r}_n + 1) & \text{if } k = n+1,\ m = n \\ 1 \oplus 1 & \text{if } k = n+1,\ m < n \end{cases}$$

The process C that we are after is C_0.

We prove that p MUST^{P2P} C_0. Definition 3.1.3 requires us to prove that the maximal computations of $p \mid\mid C_0$ are successful. Fix a maximal computation of $p \mid\mid C_0$,

$$p \mid\mid C_0 = p^0 \mid\mid C_0^0 \xrightarrow{\tau} p^1 \mid\mid C_0^1 \xrightarrow{\tau} p^2 \mid\mid C_0^2 \xrightarrow{\cdots} \tag{4.19}$$

Note that the computation in (4.19) may be infinite.

Because of the construction of C_0, the computation in (4.19) begins with moves due to a (possibly empty) prefix of s, say s_j for some $0 \leq j \leq n$, that leads C_0 to C_0^j (which is C_0 itself is s_j is empty). Let us consider the longest s_j that satisfies the condition just given. By unzipping the computation at hand we obtain the contributions of p and C_0, which begins with the ensuing prefixes,

$$p \stackrel{s_j}{\Longrightarrow} p', \qquad C_0 \stackrel{\overline{s}_j}{\Longrightarrow} C_0^j$$

Our assumption on s_j ensures that C_0^j and p' cannot interact, because the only action to synchronise is α_{j+1}, and $s_j\alpha_{j+1}$ cannot appear in the computation.

We explain why in the computation there is a successful derivative of C_0. If in the prefix of contribution of C_0 there is an internal move, then that contribution contain a successful state. In the opposite case, observe that the hypothesis $p \Downarrow_{\text{P2P}} s$ ensures that $p \Downarrow s$, and so $p' \Downarrow$. This fact, $C_0^j \xrightarrow{\tau} \xrightarrow{\checkmark}$, and the fact that the computation in (4.19) is maximal imply that the computation contains a derivative of C_0^j which is successful.

Now we discuss why p reaches a successful state. If the contribution $p \stackrel{s_j}{\Longrightarrow} p'$ contains a successful state, then we have nothing more to discuss. In the opposite case, $j \leq m$, and $p' \in (p \text{ AFTER}_{\not\checkmark} s_j)$. Since $j \leq m$, $C_0^j = ((\hat{r}_j + 1) \oplus (\hat{r}_j + 1))) + \overline{\alpha}_{j+1}.C_{k+1}$, and by construction $\hat{r}_j$ MUST $\bigoplus(p \text{ AFTER}_{\not\checkmark} s_j)$. It follows that $\hat{r}_j$ MUST p'. Since p' and C_0^j cannot interact, the computation in (4.19) after the state $p' \mid\mid C_0^j$ contains a maximal computation of $p' \mid\mid \hat{r}_j$. Since $\hat{r}_j$ MUST p' it follows that p' reaches a successful state.

We have proven that p MUST^{P2P} C_0, so the hypothesis $p \sqsubseteq_{\text{P2P}} q$ implies that q MUST^{P2P} C_0. We still have to prove that if $q \stackrel{s}{\Longrightarrow} q'$ then $q' \Downarrow$. This is true for otherwise there exist a maximal computation of $q \mid\mid C_0$ which is not successful, namely

$$q \mid\mid C_0 \Longrightarrow q' \mid\mid 1 \oplus 1 \xrightarrow{\tau} q' \mid\mid 1 \oplus 1 \xrightarrow{\tau} q' \mid\mid 1 \oplus 1 \xrightarrow{\tau} \ldots$$

This computations contradicts q MUST^{P2P} C_0. □

Corollary 4.3.9. *For every $s \in Act^\star$, and every $p, q \in \mathsf{CCS}_{w\tau}$, if $p \sqsubseteq_{\text{P2P}} q$ and $p \Downarrow_{\text{P2P}} s$, then $q \Downarrow s$.*

Proof. Thanks to Lemma 4.1.7, to prove that $q \Downarrow s$ we have to show that

$$\text{for every } s' \text{ prefix of } s, \text{ if } q \stackrel{s'}{\Longrightarrow} q' \text{ then } q' \Downarrow \tag{4.20}$$

Let s' be a prefix of s such that $q \stackrel{s'}{\Longrightarrow} q'$. The hypothesis $p \Downarrow_{\text{P2P}} s$ implies that $p \Downarrow_{\text{P2P}} s'$, so Lemma 4.3.8 and $q \stackrel{s'}{\Longrightarrow} q'$ ensure that $q' \Downarrow$. Since the only assumption on s' is that it is a prefix of s, we have proven the implication in (4.20). □

Lemma 4.3.10. If $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$, then there exists a p such that p MUST r, $p \stackrel{\checkmark}{\not\longrightarrow}$ and $p \stackrel{\tau}{\not\longrightarrow}$.

Proof. We prove that if $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ then there exists a process p such that $p \stackrel{\checkmark}{\not\longrightarrow}$. The assumption that $r \in \mathcal{U}_{\text{CLT}}^{\text{MUST}}$ implies that there exists a p such that p MUST r. If $p \stackrel{\checkmark}{\not\longrightarrow}$ there is nothing more to prove. If $p \xrightarrow{\checkmark}$, then $p = p' + \sum_{i \in I} 1$ for some non-empty set I and p' such that $p' \stackrel{\checkmark}{\not\longrightarrow}$. Lemma 4.1.23 implies that $p' \approx_S p$, so p' MUST r.

Now we prove that there exists a process p such that p MUST r and $p \overset{\tau}{\not\longrightarrow}$. The assumption that $r \in \mathcal{U}_{\mathrm{CLT}}^{\mathrm{MUST}}$ implies that there exists a p such that p MUST r. If $p \overset{\tau}{\not\longrightarrow}$, then there is nothing more to prove. If $p \overset{\tau}{\longrightarrow}$, then either p diverges or p converges. If p diverges, then Lemma 3.1.6 implies that $r \overset{\checkmark}{\longrightarrow}$. It follows that $\mathsf{0}$ MUST r; as $\mathsf{0} \overset{\tau}{\not\longrightarrow}$, the process $\mathsf{0}$ suit our aims.

If p converges then there exists a stable p' such that all the maximal computation of $r \parallel p'$ are suffix of the computation

$$r \parallel p \Longrightarrow r \parallel p'$$

Since p MUST r, it follows that all the maximal computations of $r \parallel p'$ are client-successful, and so p' MUST r. □

Lemma 4.3.11. Let $p \sqsubseteq_{\mathrm{P2P}} q$. For every $s \in Act^\star$, if $p \Downarrow_{\mathrm{P2P}} s$ and $q \overset{s}{\Longrightarrow}$, then $p \overset{s}{\Longrightarrow}$.

Proof. Under the hypothesis, we have to exhibit a p' such that $p \overset{s}{\Longrightarrow} p'$.

Let n be the length of s (i.e. $n = \mathsf{len}(s)$), $s = \alpha_1\alpha_2 \dots \alpha_n$, and let s' be the longest prefix of s such that $p \overset{s'}{\Longrightarrow}_{\not\checkmark}$. The argument depends on the existence of s', and has to following structure: we define a process C such that

1. $q \not\text{MUST}^{\mathrm{P2P}} C$
2. C MUST p
3. $p \not\text{MUST}\ C$ implies that $\mathrm{ACC}(p, s) \neq \emptyset$

Either s' exists or it does not exist.

s' does not exist In this case we cannot infer $p \overset{\varepsilon}{\Longrightarrow}_{\not\checkmark}$, thus $p \overset{\checkmark}{\longrightarrow}$. We define the process C as we did in Corollary 4.1.14. The argument in that lemma implies that $q \not\text{MUST}\ C$, and so Definition 3.1.3 implies that $q \not\text{MUST}^{\mathrm{P2P}} C$. The hypothesis $p \sqsubseteq_{\mathrm{P2P}} q$ ensures that $p \not\text{MUST}^{\mathrm{P2P}} C$. Since $p \overset{\checkmark}{\longrightarrow}$, it is clear that C MUST p, and therefore $p \not\text{MUST}^{\mathrm{P2P}} C$ implies that $p \not\text{MUST}\ C$. The hypothesis $p \Downarrow_{\mathrm{P2P}} s$ implies that $p \Downarrow s$, and so to prove that $p \overset{s}{\Longrightarrow} p'$ for some process p', we can reason as in Corollary 4.1.14.

s' exists Let $s' = \alpha_0\alpha_1 \dots \alpha_m$, with $m \leq n$. For every $0 \leq k \leq m$, the assumption $p \overset{s}{\Longrightarrow}_{\not\checkmark}$ ensures that $p \overset{s_k}{\Longrightarrow}_{\not\checkmark}$, and so Lemma 4.2.15 and Lemma 4.3.10 implies that there exists a $\hat{r}_k$ such that $\hat{r}_k$ MUST $\bigoplus(p \text{ AFTER}_{\not\checkmark} s_k)$, $\hat{r}_k \overset{\tau}{\not\longrightarrow}$ and $\hat{r}_k \overset{\checkmark}{\not\longrightarrow}$. For every $0 \leq i \leq n+1$, let

$$C_i \overset{\mathsf{def}}{=} \begin{cases} ((\hat{r}_i + \mathsf{1}) \oplus (\hat{r}_i + \mathsf{1})) + \overline{\alpha}_i.C_{i+1} & \text{if } 0 \leq i \leq m \\ (\mathsf{1} \oplus \mathsf{1}) + \overline{\alpha}_{i+1}.C_{i+1} & \text{if } m < i < n \\ \hat{r}_n & \text{if } i = n+1,\ m = n \\ \mathsf{0} & \text{if } i = n+1,\ m < n \end{cases}$$

The C we are after is C_0. Note that by construction C_n is either $\mathsf{0}$ or $\hat{r}_n$, so $C_n \overset{\tau}{\not\longrightarrow}$ and $C_n \overset{\checkmark}{\not\longrightarrow}$.

1. We prove that $q \not\text{MUST}^{\mathrm{P2P}} C_0$. By hypothesis $\mathrm{ACC}(q, s) \neq \emptyset$, so there exists a q' such that $q \overset{s}{\Longrightarrow} q'$. If q' diverges then we infer the ensuing maximal computation

$$q \parallel C_0 \Longrightarrow q' \parallel C_n \overset{\tau}{\longrightarrow} q'_1 \parallel C_n \overset{\tau}{\longrightarrow} q'_2 \parallel C_n \overset{\tau}{\longrightarrow} \dots$$

The computation above is not successful, because no C_i is successful. If q' converges, then there exists a stable q'' such that we can infer the following maximal computation

$$q \parallel C_0 \Longrightarrow q'' \parallel C_n \overset{\tau}{\not\longrightarrow}$$

The finite computation above is not successful. What we have argued so far proves that $q \not\textsc{must}^{\text{P2P}} C_0$.

2. We prove that C_0 MUST p. We are required to show that all the maximal computations of $p \,||\, C_0$ are client-successful. Fix a maximal computation of $p \,||\, C_0$ and unzip it; the contributions that we obtain begin with the following prefixes of s,

$$p \stackrel{s_j}{\Longrightarrow} p', \qquad C_0 \stackrel{\overline{s}_j}{\Longrightarrow} C_j$$

where we let $0 \leq j \leq m$ be greatest j such that a state C_j appears in the computation. If the action sequence $p \stackrel{s_j}{\Longrightarrow} p'$ contains a successful state, then the whole computation is client-successful. In the opposite case, $p \stackrel{s_j}{\Longrightarrow}\not\checkmark p'$, and so $j \leq m$. It follows that C_j contains a term $\hat{r}_j$ such that $\hat{r}_j$ MUST $\bigoplus(p \text{ AFTER } s_j)$; this implies that $\hat{r}_j$ MUST p'. Our assumption on s_j and C_j ensure that the remaining part of the computation we unzipped contain a maximal computation of $p' \,||\, \hat{r}_j$; it follows that the maximal computation at hand is client-successful.

We have explained why all the maximal computations of $p \,||\, C_0$ are client-successful, thus C_0 MUST p.

Since $q \not\textsc{must}^{\text{P2P}} C_0$ the hypothesis $p \sqsubseteq_{\text{P2P}} q$ implies that $p \not\textsc{must}^{\text{P2P}} C_0$. As C_0 MUST p, it must be the case that $p \not\textsc{must} C_0$. By reasoning as we did in Lemma 4.1.9 we prove that $p \stackrel{s}{\Longrightarrow} p'$ for some p'. □

Corollary 4.3.12. *Let $p \sqsubseteq_{\text{P2P}} q$. For every $s \in Act^\star$, if $p \Downarrow_{\text{P2P}} s$ and $\text{ACC}(q, s) \neq \emptyset$ then $\text{ACC}(p, s) \neq \emptyset$.*

Proof. The hypothesis $\text{ACC}(q, s) \neq \emptyset$ implies that $q \stackrel{s}{\Longrightarrow}$, Lemma 4.3.11 implies that $p \stackrel{s}{\Longrightarrow} p'_1$ for some p'_1. The hypothesis that $p \Downarrow_{\text{P2P}} s$ implies $p \Downarrow s$. In turn this ensures that $p' \Downarrow$, and so there exists a p'' such that $p \stackrel{s}{\Longrightarrow} p'' \stackrel{\tau}{\not\longrightarrow}$. It follows that $S(p'') \in \text{ACC}(p, s)$. □

The last lemma that we need to prove establishes how the ready sets of peers are related by $\sqsubseteq_{\text{P2P}}$. This lemma is analogous to Lemma 4.1.15, but the matching of ready sets is relaxed by focusing on usable actions.

Lemma 4.3.13. Let $p \sqsubseteq_{\text{P2P}} q$. For every $s \in Act^\star$, if $p \Downarrow_{\text{P2P}} s$, then for every $B \in \text{ACC}(q, s)$ there exists a set A such that $A \in \text{ACC}(p, s)$ and $A \cap \mathsf{ua}_{\not\checkmark}(p, s) \subseteq B$.

Proof. Fix a string s such that $B \in \text{ACC}(q, s)$ for some set B, that $p \Downarrow_{\text{CLT}} s$, $p \Downarrow s$, and let $s = \beta_1 \ldots \beta_n$. The hypothesis of this lemma let us use Corollary 4.3.12, which ensures that the set $\text{ACC}(p, s)$ is non-empty: $\text{ACC}(p, s) = \{ A_i \mid i \in I \}$ for some non-empty set I. The proof is by contradiction; we suppose that

$$\text{for every} i \in I \text{ there exists an action } \alpha_i \in A_i \cap \mathsf{ua}_{\not\checkmark}(p, s) \text{ such that } \alpha_i \notin B \tag{4.21}$$

Using this assumption we contradicts the hypothesis that $p \sqsubseteq_{\text{P2P}} q$; that is, we define a peer C that can distinguish p and q, in the sense that

a) $C \not\textsc{must}^{\text{P2P}} q$

b) C MUST$^{\text{P2P}}$ p

The argument depends on p being successful or not.

If $p \stackrel{\checkmark}{\longrightarrow}$, then we define the peer C as we did in Lemma 4.1.15; the same reasoning we used in that lemma implies that $C \not\textsc{must} q$ and that C MUST p. The former fact implies $C \not\textsc{must}^{\text{P2P}} q$. The latter fact and $p \stackrel{\checkmark}{\longrightarrow}$ imply that C MUST$^{\text{P2P}}$ p.

If $p \overset{\checkmark}{\not\longrightarrow}$, then there exists a longest s' such that $p \overset{s'}{\Longrightarrow}_{\not\checkmark}$ and that s' is a prefix of s. Let $s' = \beta_1\beta_2 \ldots \beta_m$, with $m \leq n$. We partition the set I in two subsets U and S by letting $U = \{\, i \in I \mid p \overset{s\alpha_i}{\Longrightarrow}_{\not\checkmark}$ and p usbl $s\alpha_i \,\}$ and $S = \{\, i \in I \mid p \overset{s\alpha_i}{\not\Longrightarrow}_{\not\checkmark} \,\}$.

For every $u \in U$, p usbl $s\alpha_u$, $p \overset{s\alpha_u}{\Longrightarrow}_{\not\checkmark}$, Lemma 4.2.15, and Lemma 4.3.10 ensure that there exists a $\hat{r}_u$ such that

- $\hat{r}_u$ MUST $\bigoplus(p$ AFTER$_{\not\checkmark}$ $s\alpha_u)$
- $\hat{r}_u \overset{\tau}{\not\longrightarrow}$

For every $0 \leq j \leq m$, the assumption $p \overset{s}{\Longrightarrow}_{\not\checkmark}$, the hypothesis $p \Downarrow_{\text{CLT}} s$ and Lemma 4.2.15 ensure that there exists a r_j such that r_j MUST $\bigoplus(p$ AFTER$_{\not\checkmark}$ $s_j)$. For every $0 \leq k \leq n$, let

$$C_k = \begin{cases} ((r_k + 1) \oplus (r_k + 1)) + \overline{\beta}_{k+1}.C_{k+1} & \text{if } 0 \leq k \leq m \\ (1 \oplus 1) + \overline{\beta}_{k+1}.C_{k+1} & \text{if } m < k < n \\ (\sum_{u \in U} \overline{\alpha}_u.(\hat{r}_u + 1)) + (\sum_{j \in S} \overline{\alpha}_j.1) & \text{if } k = n \end{cases}$$

Note that $C_k \Downarrow$ for every $0 \leq k \leq n$, and that $C_n \overset{\tau}{\not\longrightarrow}$ and $C_n \overset{\checkmark}{\not\longrightarrow}$.

a) We prove that $q \not\text{MUST}^{\text{P2P}}\ C_0$; it is enough to exhibit a maximal computation of $C_0 \mid\mid q$ that is not client-successful. Definition 4.1.11 implies that there exists a $q \overset{s}{\Longrightarrow} q'$ and $S(q') = B$. If q' diverges, we there exists the maximal computation

$$C_0 \mid\mid q \Longrightarrow C_n \mid\mid q' \overset{\tau}{\longrightarrow} C_n \mid\mid q'_1 \overset{\tau}{\longrightarrow} C_n \mid\mid q'_2 \overset{\tau}{\longrightarrow} \ldots$$

The computation above is not client-successful. If q' converges, then there exists a stable q'' such that $S(q'') \subseteq B$, and $q' \Longrightarrow q''$; hence there exists the maximal computation $C_0 \mid\mid q \Longrightarrow C_n \mid\mid q'' \overset{\tau}{\not\longrightarrow}$ where $C_n \mid\mid q''$ is stable because both processes are, and by construction the state q' cannot interact with C_n. The computation is not client-successful.

b) We prove that p MUST$^{\text{P2P}}$ C_0; Definition 3.1.3 requires us to prove that every maximal computation of $p \mid\mid C_0$ is successful.

Fix a maximal computation of $p \mid\mid C_0$, and unzip it. The contributions that we obtain begins with a (possibly empty) prefix of s,

$$C_0 \overset{\overline{s}_j}{\Longrightarrow} C_j, \qquad p \overset{s_j}{\Longrightarrow} p'$$

We can assume s_j to be the longest prefix of s such that in the computation at hand $C_0 \overset{\overline{s}_j}{\Longrightarrow} C_j$. The hypothesis that $p \Downarrow_{\text{P2P}} s$ implies that $p' \Downarrow$.

We explain why C_0 and p reaches a successful state.

- If $j = n$ then $C_i \overset{\tau}{\not\longrightarrow}$. Since $p' \Downarrow$, $C_n \overset{\tau}{\not\longrightarrow}$ implies that in the computation we unzipped there is p'' such that $p' \Longrightarrow p'' \overset{\tau}{\not\longrightarrow}$. Definition 4.1.11 and the construction of C_n imply that C_n and p'' can interact via some action α_i. As the unzipped computation is maximal, and both C_n and p'' are stable, the computation contains the state $r \mid\mid \hat{p}$ resulting from the interaction. By construction $r \overset{\checkmark}{\longrightarrow}$.

  Now we explain why p reaches a successful state. The argument depends on the action α_i. If $i \in S$ then $p \overset{s\alpha_i}{\not\Longrightarrow}_{\not\checkmark}$, thus there is a successful state in the action sequence $p \overset{s\alpha_i}{\Longrightarrow} \hat{p}$. If $i \in U$, then $r = \hat{r}_u + 1$, so the remaining part of the unzipped computation is a maximal computation of $\hat{p} \mid\mid \hat{r}_u$. The construction of $\hat{r}_u$ implies in the computation $\hat{p}$ reaches a successful state, and so does p.

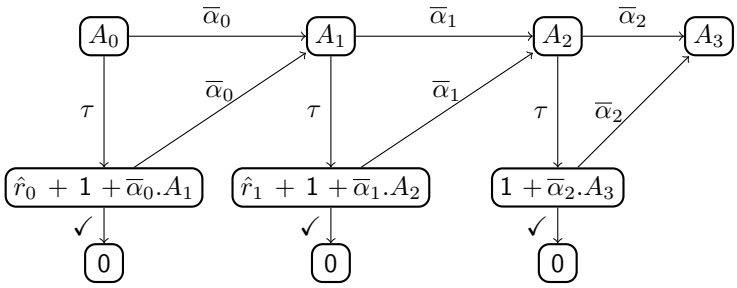


Figure 4.11: Peer used to prove Lemma 4.3.13, with $m = 1$ and $n = 3$

- If $j \leq n$, then $C_j = (r' \oplus r') + \overline{\beta}_{i+j}.C_{i+j}$ and $C_j \Downarrow$. Since $p' \Downarrow$ and p' cannot interact on $\overline{\beta}_{i+j}$, the computation contains a state $p'' \mid\mid C_j$ such that p'' is stable and $p' \Longrightarrow p''$. As the computation is maximal, the facts that $p'' \stackrel{\tau}{\not\longrightarrow}$ and that p'' and C_j cannot interact, imply that the computation contains a state reached by $C_j \mid\mid p''$ thanks to an internal move of C_j. This state is $r' + \overline{\beta}_{i+j}.C_{i+j} \mid\mid p''$. The construction of r' ensures that $r' \stackrel{\checkmark}{\longrightarrow}$.
  Now let us see why p reaches a successful state. If the action sequence $p \stackrel{s_j}{\Longrightarrow} p''$ contains a successful state, then we have nothing more to discuss. Let us suppose that $p \stackrel{s_j}{\Longrightarrow}_{\not\checkmark} p''$. It follows that $r' = r_j + 1$, and that r_j MUST $\bigoplus(p \text{ AFTER } s_j)$. In turn this ensures that r_j MUST p''. The remaining part of the unzipped computation must be a computation of $r_j \mid\mid p''$, so r_j MUST p'' implies that p'' reaches a successful state. It follows that p reaches a successful state in the computation that we unzipped.

We have proven that the maximal computations of $p \mid\mid C_0$ are successful, so p MUST$^{\text{P2P}}$ C_0.

We have proven that p MUST$^{\text{P2P}}$ C_0 and that $q \not\text{MUST}^{\text{P2P}}$ C_0; this contradicts the hypothesis $p \sqsubseteq_{\text{P2P}} q$, and so the assumption is (4.21). This proves the lemma. □

Lemma 4.3.14. Let $p \sqsubseteq_{\text{P2P}} q$. For every $u \in Act^{\infty}$, if $p \Downarrow_{\text{P2P}} u$ and $q \stackrel{u}{\Longrightarrow}$, then $p \stackrel{u}{\Longrightarrow}$.

Proof. If $p \stackrel{\checkmark}{\longrightarrow}$, then the argument is the same we used in Lemma 4.1.16. If $p \stackrel{\checkmark}{\not\longrightarrow}$, then we have other two cases. If $p \stackrel{u}{\Longrightarrow}_{\not\checkmark}$, then $p \stackrel{u}{\Longrightarrow}$. Suppose that $p \stackrel{u}{\not\Longrightarrow}_{\not\checkmark}$, then there exists the greatest $m \in \mathbb{N}$ such that $p \stackrel{u_m}{\Longrightarrow}_{\not\checkmark}$. The hypothesis $p \Downarrow_{\text{P2P}} u$ ensures that p $\mathsf{usbl}_{\not\checkmark}$ u, so for every $0 \leq i \leq m$ there exists a process r_i such that r_i MUST $\bigoplus(p \text{ AFTER } u_i)$. For every $n \in N$ let

$$A_n \stackrel{\text{def}}{=} \begin{cases} ((r_n + 1) \oplus (r_n + 1)) + \overline{\alpha}_n.A_{n+1} & \text{if } i \leq m \\ (1 \oplus 1) + \overline{\alpha}_n.A_{n+1} & \text{otherwise} \end{cases}$$

The proof now proceeds as the proof of Lemma 4.1.16, and relies on the fact that the hypothesis $p \Downarrow_{\text{P2P}} u$ implies that $p \Downarrow u$. □

The previous lemmas state that peers related by $\sqsubseteq_{\text{P2P}}$ are compared as servers, only if they are usable as clients. Indeed, this is the intuition behind Definition 4.3.7.

The peer pre-order turns out to be a nesting of the server pre-order inside the client pre-order, in the following sense.

Definition 4.3.15. [Semantic peer-preorder]
Let $p \precsim_{\text{P2P}} q$ if and only if $p \precsim_{\text{CLT}} q$ and $p \precsim_{u\text{SVR}} q$. □

Proposition 4.3.16. [Completeness]
If $p \sqsubseteq_{\text{P2P}} q$ then $p \precsim_{\text{P2P}} q$.

Proof. We have to prove that $\sqsubseteq_{\text{P2P}} \subseteq \precsim_{\text{CLT}}$ and that $\sqsubseteq_{\text{P2P}} \subseteq \precsim_{u\text{SVR}}$. The first inclusion follows from Proposition 4.3.5 and Theorem 4.2.37. The second set inclusion follows from Proposition 4.3.5, Corollary 4.3.9, Lemma 4.3.13, and Lemma 4.3.14. □

Theorem 4.3.17. [Alternative characterisation $\sqsubseteq_{\text{P2P}}$]
For every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, $p \sqsubseteq_{\text{P2P}} q$ if and only if $p \precsim_{\text{P2P}} q$.

Proof. We have to prove two implications:

- if $p \sqsubseteq_{\text{P2P}} q$ then $p \precsim_{\text{P2P}} q$
- if $p \precsim_{\text{P2P}} q$ then $p \sqsubseteq_{\text{P2P}} q$

In view of Proposition 4.3.16, we prove only the second implication. Fix two processes p and q such that $p \precsim_{\text{P2P}} q$; we are required to show that $p \sqsubseteq_{\text{P2P}} q$; that is, if p MUST^{P2P} r then q MUST^{P2P} r for every process r. Fix a process r such that p MUST^{P2P} r; we explain why q MUST^{P2P} r. Definition 3.1.3 requires us to prove that all the maximal computations of $q \,||\, r$ are successful.

We prove a preliminary result that will ease our task. The definition of $\sqsubseteq_{\text{P2P}}$ ensures that $p \precsim_{\text{CLT}} q$, so Theorem 4.2.37 implies that $p \sqsubseteq_{\text{CLT}} q$. Since p MUST^{P2P} r, it follows that r MUST p. The inequality $p \sqsubseteq_{\text{CLT}} q$ implies that r MUST q. It follows that all the maximal computations of $q \,||\, r$ are client-successful.

It follows that we have to prove only that the maximal computations of $q \,||\, r$ contain a state $q' \,||\, r'$ wherein $r' \xrightarrow{\checkmark}$.

Fix a maximal computation of $q \,||\, r$,

$$q \,||\, r = q_0 \,||\, r_0 \xrightarrow{\tau} q_1 \,||\, r_1 \xrightarrow{\tau} q_2 \,||\, r_2 \xrightarrow{\tau} \ldots \tag{4.22}$$

The computation is either finite or it is infinite.

If it is finite, then by unzipping the computation we obtain

$$q \stackrel{s}{\Longrightarrow} q_k, \quad r \stackrel{\overline{s}}{\Longrightarrow} r_k$$

where $q_k \,||\, r_k \stackrel{\tau}{\not\longrightarrow}$. Since $r \stackrel{\overline{s}}{\Longrightarrow} r_k$, and p MUST^{P2P} r implies r MUST p, Lemma 4.2.35 (point (i)) guarantees that $p \,\mathsf{usbl}_{\not\checkmark}\, s$. We are required to exhibit a successful state among the derivatives of r. Since $q \stackrel{s}{\Longrightarrow} q_k$, $S(q_k) \in \text{ACC}(q, s)$. As $p \,\mathsf{usbl}_{\not\checkmark}\, s$, point (1b) of Definition 4.3.7 implies that either $p \not\Downarrow s$, or there exists a set $A \in \text{ACC}(p, s)$ such that $A \cap \mathsf{ua}_{\not\checkmark}(p, s) \subseteq S(q_k)$. In the first case p performs a prefix of s, say s', and reaches a state p' that diverges: $p \stackrel{s'}{\Longrightarrow} p' \Longrightarrow \ldots$. By zipping this action sequence of p with a prefix of the action sequence $r \stackrel{\overline{s}}{\Longrightarrow} r_k$ we obtain a maximal computation of $p \,||\, r$ in which p diverges. As p MUST^{P2P} r, the computation of $p \,||\, r$ contains a successful derivative of r; this derivative appears also in (4.22), so the maximal computation in (4.22) contains a successful derivative of r.

In the second case, there exists a stable p' such that $S(p') = A$ and $p \stackrel{s}{\Longrightarrow} p'$. Consider the computation

$$p \,||\, r \Longrightarrow p' \,||\, r_k$$

If the state $p' \,||\, r_k$ is terminal (i.e. stable), then the computation is maximal, and so p MUST^{P2P} r ensures that one of the derivatives of r is successful. This implies that also the computation in (4.22) contains a successful derivative of r.

We have to prove that $p' \,||\, r_k \stackrel{\tau}{\not\longrightarrow}$. The argument is the same that we used in paragraph **Terminal state** of the proof of Theorem 4.2.37.

We have proven that if the computation in (4.22) is finite, then r reaches a successful state. Now we prove that also the infinite computations enjoy this property. If the computation at hand is infinite,

then there are three subcases to discuss: q and r engage in infinite traces, or q diverges along the computation, or r diverges.

Suppose that by unzipping the computation in (4.22) we obtain the infinite contributions

$$q \stackrel{u}{\Longrightarrow} \ldots, \quad r \stackrel{\overline{u}}{\Longrightarrow} \ldots \tag{4.23}$$

We have to show that one of the derivatives of r is successful.

As $r \stackrel{\overline{u}}{\Longrightarrow}$ and p MUST^{P2P} r, point (ii) of Lemma 4.2.35 implies that p $\mathsf{usbl}\not\swarrow$ u. Either $p \not\Downarrow u$ or $p \Downarrow u$. In the first case p performs some prefix s of u and reaches a state p' that diverges. Let us zip the action sequence $p \stackrel{s}{\Longrightarrow} p'$ with $r \stackrel{\overline{s}}{\Longrightarrow} r'$; we obtain an infinite computation that reaches a state $p' \,||\, r'$ and then let p' diverge. Since p MUST^{P2P} r the new computation must contain a successful derivative of r. This derivative appears also in the computation in (peer-max-comp).

In the second case, $p \Downarrow_{\text{P2P}} u$. Since $q \stackrel{u}{\Longrightarrow}$, point (2) of Definition 4.3.7 implies that $p \stackrel{u}{\Longrightarrow}$. By zipping this infinite trace of p with $r \stackrel{\overline{u}}{\Longrightarrow}$ we obtain a maximal computation of $p \,||\, r$. The assumption p MUST^{P2P} r ensures that r reaches a successful state.

We have shown that if the computation in (4.22) is due to two infinite traces, then r reaches a successful state.

We discuss the case of divergence of q or r. Suppose that the visible traces performed by q and r be finite:

$$q \stackrel{s}{\Longrightarrow} q_k, \quad r \stackrel{\overline{s}}{\Longrightarrow} r_k \tag{4.24}$$

The fact that $r \stackrel{\overline{s}}{\Longrightarrow} r_k$ implies p $\mathsf{usbl}\not\swarrow$ s. Either q_k diverges, or r_k diverges, or both states diverge.

q_k diverges We have to show a successful state among $r, \ldots, r_k$. As q_k diverges, $q \not\Downarrow s$. Point (1a) implies that $p \not\Downarrow_{\text{P2P}} s$. Since p $\mathsf{usbl}\not\swarrow$ s, the fact that $p \not\Downarrow_{\text{P2P}} s$ implies that $p \not\Downarrow s$. This implies that there exists a prefix s' of s such that $p \stackrel{s'}{\Longrightarrow} p'$ and p' diverges.

Zip this action sequence of p with the suitable prefix of the actions sequence of r, and let p' diverge. As p MUST^{P2P} r it follows that there is a successful state in the contribution of r; this state is reached by r also in the computation we unzipped.

r_k diverges We have to show a successful state among $r, \ldots, r_k$. Point (2) of Definition 4.3.7, $q \stackrel{s}{\Longrightarrow}$, and p $\mathsf{usbl}\not\swarrow$ s imply that either $p \not\Downarrow s$ or $p \stackrel{s}{\Longrightarrow}$. In the first case $p \stackrel{s'}{\Longrightarrow} p'$ for some prefix s' of s and p' diverges. By zipping this action sequence of p a prefix of the actions sequence of r, we obtain a maximal computation of $p \,||\, r$. The assumption p MUST^{P2P} r implies p MUST r, so the new computation contains a state in which the derivative of r is successful. The successful derivative of r appears also in Eq. (4.24).

In the second case, $p \stackrel{s}{\Longrightarrow}$. By zipping this action sequence of p with the actions sequence of r in Eq. (4.24), we obtain a maximal computation of $p \,||\, r$ in which the client side diverges. It follows that one of the derivatives of r in it is successful. As the derivatives of r in the new computation appear also in (4.24), the computation of $q \,||\, r$ that we unzipped contains a successful derivative of r. □

4.3.1 Relations between notions and pre-orders

Throughout the last three sections of this chapter we have studied the pre-orders given by the MUST relation. Our study has been driven by the need for alternative characterisations for these pre-orders, and we have shown the equalities

$$\sqsubseteq_{\text{SVR}} = \precsim_{\text{SVR}}, \qquad \sqsubseteq_{\text{CLT}} = \precsim_{\text{CLT}}, \qquad \sqsubseteq_{\text{P2P}} = \precsim_{\text{P2P}}$$

where the relations $\precsim_{\text{SVR}}, \precsim_{\text{CLT}}$, and $\precsim_{\text{P2P}}$ are defined using the following behavioural notions:

Server:	$\Longrightarrow$	AFTER	$\Downarrow$	ACC	–	–
Client:	$\Longrightarrow\!\!\!\!\!\!\diagup$	AFTER$\!\!\!\!\!\!\diagup$	$\Downarrow^{\checkmark}$	ACC$\!\!\!\!\!\!\diagup$	$\mathsf{ua}\!\!\!\!\!\!\diagup$	$\mathsf{usbl}\!\!\!\!\!\!\diagup$
Peer:	–	–	$\Downarrow_{\text{P2P}}$	–	–	$\Downarrow_{\text{P2P}}$

Figure 4.12: Predicates to characterise the server, the client, and the peer pre-orders.

$$\sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}} \quad \subset \quad \sqsubseteq_{\text{P2P}} \quad \subset \quad \sqsubseteq_{\text{CLT}}$$

Figure 4.13: Relations among the MUST pre-orders

- (unsuccessful) traces
- (unsuccessful) acceptance sets
- convergence (to success)
- usability

Figure 4.12 contains the symbols that we have used to formalise the notions listed above. Roughly speaking, for each server-side predicate, we had to define an analogous client-side predicate, by restricting our attention to the unsuccessful traces/actions; and we had to make explicit the role of usable clients/actions. Note that if Definition 4.1.17 does not mention explicitly the usability of the servers or their actions, it is because every server is usable: for every process p there exists a client r such that p MUST r; the client $\mathsf{1}$ is an example. The characterisation of the peer pre-order relies an the predicates used to reason on $\sqsubseteq_{\text{SVR}}$ and $\sqsubseteq_{\text{CLT}}$; the only feature typical of $\sqsubseteq_{\text{P2P}}$ is the combination of $\Downarrow$ and $\mathsf{usbl}\!\!\!\!\!\!\diagup$ that is denoted by $\Downarrow_{\text{P2P}}$.

In Figure 4.13 we have depicted how the MUST pre-orders are related with each other. The set inclusions are proven respectively in Lemma 4.3.3 and Proposition 4.3.5. The set inclusions are strict because of the following facts

$$\sqsubseteq_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{SVR}}, \quad \sqsubseteq_{\text{CLT}} \not\subseteq \sqsubseteq_{\text{P2P}}$$

which are proven by the ensuing inequalities,

$$\alpha.\mathsf{0} \quad \not\sqsubseteq_{\text{SVR}} \quad \beta.\mathsf{0} \qquad\qquad \alpha.\mathsf{0} \quad \sqsubseteq_{\text{P2P}} \quad \beta.\mathsf{0}$$
$$\alpha.(\mathsf{1}+\beta.\mathsf{0}) \quad \not\sqsubseteq_{\text{P2P}} \quad \alpha.(\mathsf{1}+\gamma.\mathsf{0}) \qquad\qquad \alpha.(\mathsf{1}+\beta.\mathsf{0}) \quad \sqsubseteq_{\text{CLT}} \quad \alpha.(\mathsf{1}+\gamma.\mathsf{0})$$

Syntax free proofs of completeness To prove the completeness of the alternative pre-orders $\precsim_{\text{SVR}}$, $\precsim_{\text{CLT}}$ and $\precsim_{\text{P2P}}$, we defined ad-hoc processes A's, C's, and so forth. In general, these processes allow us to distinguish, in some sense, the other two processes that we are comparing, say p_1 and p_2, or r_1 and r_2. Although we used the syntax of $\mathsf{CCS}_{\text{w}\tau}$ to define the processes A's and C's, the arguments do not depend on the syntax. The arguments for the server, client and peer pre-order, can be used in any LTS that contains the graphs depicted respectively in Figure 4.3, Figure 4.10, and Figure 4.11.

In this chapter we have investigated the three pre-orders that are naturally given the MUST testing relation: $\langle \sqsubseteq_{\text{SVR}}, \sqsubseteq_{\text{CLT}}, \sqsubseteq_{\text{P2P}} \rangle$. In particular we have unravelled the behavioural properties that two processes have to enjoy in order to be related by one of the pre-orders. While Theorem 4.2.37 and Theorem 4.3.17 are novel, Theorem 4.1.21 is very similar to the characterisation of the standard MUST pre-order [De Nicola and Hennessy, 1984, see Theorem 6.4.5].

4.4 Related Work

We compare our results with the relevant literature.

The behavioural characterisation of our MUST server pre-order is the same of the well-known MUST pre-order $\sqsubseteq_{\text{MUST}}$; to see why it is enough to compare Definition 4.1.17 with [De Nicola and Hennessy, 1984, Definition 6.4.1]. Nevertheless, the axioms for $\sqsubseteq_{\text{MUST}}$ on finite terms that are proposed by [De Nicola and Hennessy, 1984, Table 1] and are not complete for $\sqsubseteq_{\text{SVR}}$. This is a consequence of having defined $\sqsubseteq_{\text{SVR}}$ on terms that can perform $\checkmark$; indeed, with the original axioms we cannot prove $\mathsf{1} = \mathsf{0}$, which is true, in the sense that $\mathsf{1} \eqsim_{\text{SVR}} \mathsf{0}$.

To the best of our knowledge, the client and the peer pre-orders based on the MUST testing are original.

As for the client pre-order, the standard axiomatisation of the MUST pre-order is not sound with respect to it (on the general LTS of processes). Consider the axiom at the bottom of [Hennessy, 1985, Figure 3.6],

$$p \leq p \oplus p \qquad \textbf{(w)}$$

This axiom lets us prove that $\mathsf{1} \leq \mathsf{1} \oplus \mathsf{1}$; in Example 4.2.4, though, we have shown that $\mathsf{1} \not\sqsubseteq_{\text{CLT}} \mathsf{1} \oplus \mathsf{1}$.

The usability of clients and actions play a crucial role in the behavioural characterisation of $\sqsubseteq_{\text{CLT}}$. Condition (1b) of Definition 4.2.30 ensures that if $r_1 \sqsubseteq_{\text{CLT}} r_2$ then the non-usable actions of r_1 need not to be performed by r_2. This fact was already pointed out in [Laneve and Padovani, 2007, Section 4], but in the setting of compliance.

Fair theories Refinements for peers in the context of compliance and behavioural contracts for web-services have been investigated; one of these theories is presented in [Bravetti and Zavattaro, 2009]. Regardless of the differences between our framework and the framework of that paper, a comparison is in order. First we swiftly introduce the formalism used in [Bravetti and Zavattaro, 2009], and then compare our peer pre-order MUST with their refinement.

In the rest of this section let us denote output actions as $\overline{\alpha}, \overline{\beta}$ and input actions without any decoration, $\alpha, \beta, \ldots$.

Bravetti and Zavattaro use an LTS denoted by contracts, which are a sublanguage of recursive CCS, and are *output persistent*. A contract C is output persistent if given $C \stackrel{w}{\Longrightarrow} C'$ with $C' \stackrel{\overline{\alpha}}{\longrightarrow}$ then: $C' \stackrel{\checkmark}{\not\longrightarrow}$ and if $C' \stackrel{\alpha}{\longrightarrow} p''$ with $\alpha \neq \overline{\alpha}$ then also $C'' \stackrel{\overline{\alpha}}{\longrightarrow}$. For instance the term $\overline{\alpha}.\mathsf{1} + \beta.\mathsf{1}$ is a process in our theory, that is ruled out in their setting, because it is not output persistent. Systems can have any finite number of parties, as general compositions of contracts are allowed

$$[C_1] \,||\, [C_2] \,||\, \ldots \,||\, [C_n]$$

The notion of successful state differs from our presentation: for a composition to be successful (i.e. perform $\checkmark$), *all* its components have to be successful (i.e. be able of performing $\checkmark$) *at the same time.*

Bravetti and Zavattaro defined the satisfaction following the fair testing of Rensink and Vogler: a system P is a *correct contract composition*, denoted $P \downarrow$ if for every P' such that $P \stackrel{\tau}{\longrightarrow}^* P'$ there exists a P'' such that $P' \stackrel{\tau}{\longrightarrow}^* P'' \stackrel{\checkmark}{\longrightarrow}$.

Definition 12 of that paper introduces the subcontract relations $\preceq_O$ on output persistent contracts, where the parameter O is a the set of *output* actions that the compositions used as tests can show.

The comparison between the peer pre-order $\sqsubseteq_{\text{P2P}}$ and the pre-orders $\preceq_O$ is complicated by two aspects,

- if $C' \preceq_O C$ then it is safe to use C' in place of C; in view of this, we will compare our peer pre-order with *the inverse* of the pre-orders $\preceq_O$;

- a priori, it is not clear how to choose the parameter O. To solve this complication we treat $\preceq_O$ as a function of O, and briefly discuss its monotonicity.

The function $\preceq_O$ is not monotonically increasing, as $\emptyset \subseteq \{\overline{\alpha}\}$, while

$$\begin{array}{lll} \alpha.\mathsf{0} & \preceq_{\emptyset} & \alpha.\mathsf{1} \\ \alpha.\mathsf{0} & \not\preceq_{\{\overline{\alpha}\}} & \alpha.\mathsf{1} \end{array}$$

On the other hand $\preceq_O$ is monotonically *decreasing* in O.

Proposition 4.4.1. If $O \subseteq O'$ then $\preceq_{O'} \subseteq \preceq_O$.

This proposition gives us two criteria to reason on all the pre-orders $\preceq_O$:

- for every O the pairs in $\preceq_{\mathcal{N}}$ are in $\preceq_O$, where $\mathcal{N}$ is the set of output actions
- for every O, O', if $O \subseteq O'$, then the pairs *not* in $\preceq_O$ are *not* in $\preceq_{O'}$.

As for the restriction on output persistent contracts, in the oncoming examples we will use only terms that enjoy that property; thus our arguments are sound.

For every action α, the following inequalities are true

$$\begin{array}{lll} \alpha.\mathsf{1} & \sqsubseteq_{\mathrm{P2P}} & \mathsf{1} + \alpha.\mathsf{0} \\ \alpha.\mathsf{1} & \not\preceq^{-1}_{\{\overline{\alpha}\}} & \mathsf{1} + \alpha.\mathsf{0} \end{array}$$

where the peer used to prove the second fact is $\overline{\alpha}.\mathsf{1}$.

Also the ensuing facts are true,

$$\begin{array}{lll} \mathsf{1} & \preceq^{-1}_{\mathcal{N}} & \tau.\mathsf{1} + \tau.\mathsf{1} \\ \mathsf{1} & \not\sqsubseteq_{\mathrm{P2P}} & \mathsf{1} \oplus \mathsf{1} \end{array}$$

Our arguments show that the pre-order $\sqsubseteq_{\mathrm{P2P}}$ and the pre-orders $\preceq^{-1}_O$ are not comparable, except when O is trivial:

Proposition 4.4.2. For every set of output actions O, the following statements are true,

- if O is non-empty, then $\sqsubseteq_{\mathrm{P2P}} \not\subseteq \preceq^{-1}_O$
- $\preceq^{-1}_O \not\subseteq \sqsubseteq_{\mathrm{P2P}}$

Chapter 5

Compliance pre-orders

In Chapter 4 we have studied the pre-orders given by the MUST relation, if we use it to establish when a process q satisfies more clients, servers or peers than a process p. The definition of MUST (Definition 3.1.1), implies that in the MUST setting, what we mean by "clients" are really *tests*, so the relation $\sqsubseteq_{\approx_{\text{CLT}}}$ tells us when a test r_2 is passed by more processes than a test t_1. It makes sense to use the theory we unravelled in Chapter 4, only if the notion of client coincides with that of test.

Example 5.0.3. Let us define two processes

$$\begin{array}{lcl} \mathit{Plane} & \stackrel{\text{def}}{=} & !\texttt{flying}.\mathit{Plane} \\ \mathit{publican} & \stackrel{\text{def}}{=} & !\texttt{stout}.?\texttt{cash}.!\texttt{chat}.\mathit{publican} \end{array}$$

The term *Plane* describes the simplest interaction that we expect a flying plane to perform: communicate (to some control tower) that it is indeed flying. Notwithstanding how simple the communication described by *Plane* is, it is useful. For if such a communication is disrupted and the plane is not landed, then there exists some problem, either in the plane or in the control towers. Note that a priori it is not know how long a plane may fly, thus the communication can go on forever.

The second process, *publican*, represents the interaction that a bar tender may carry out with a typical customer: pour (output) a pint, receive (input) some money, and then do a bit of chat.

In the MUST theory, one can prove that *publican* is a better client that *Plane*, and vice-versa,

$$\mathit{Plane} \sqsubseteq_{\approx_{\text{CLT}}} \mathit{publican}, \qquad \mathit{publican} \sqsubseteq_{\approx_{\text{CLT}}} \mathit{Plane}$$

This proves that according to the MUST theory there is no difference between the client *Plane* and the client *publican*, $\mathit{Plane} \approx_{\text{CLT}} \mathit{publican}$. Indeed, it is true that there is no difference between *Plane* and *publican*, for in the MUST setting they are both tests, and neither of them can be satisfied. □

The previous examples shows that the pre-orders given by MUST, in particular $\sqsubseteq_{\approx_{\text{CLT}}}$, should not be used if we think of clients not as tests, but as software whose requests have to be answered by servers. For instance, if we reason in the setting of web-services, one would like the client *Plane* to be distinguished from *publican*. This can be achieved by using the pre-orders that arise from the compliance relation (Definition 3.2.1).

In this chapter we investigate the compliance pre-orders. As in Chapter 4, we define three pre-orders,

$$\sqsubseteq_{\text{SVR}}, \qquad \sqsubseteq_{\text{CLT}}, \qquad \sqsubseteq_{\text{P2P}} \tag{5.1}$$

and we study under which conditions two processes are related by each one of them. In other words, we show the alternative characterisations of the pre-orders in (5.1),

$$\preceq_{\text{SVR}}, \qquad \preceq_{\text{CLT}}, \qquad \preceq_{\text{P2P}} \tag{5.2}$$

The work we carry out to define the relations in (5.1) is necessary, for the compliance pre-orders are not comparable with the MUST pre-orders. The intuitions and the notions that we have put forth in Chapter 4, turn out to be useful also in reasoning on the compliance relation, and let us define the pre-orders in (5.1).

After having studied the server and the client compliance pre-orders, we compare them with their MUST counterparts. The MUST pre-orders are not comparable with the compliance pre-orders. One natural question then is to check whether in sub-LTSs of $\langle \mathsf{CCS}_{\mathrm{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$, there is some relation between the MUST and the compliance pre-orders.

In the case of the server pre-orders, this task is eased by a property of their alternative characterisations. The alternative relations given by Definition 4.1.17 and Definition 5.1.7 characterise the server pre-orders; these characterisation remain sound and complete also in certain sub-LTSs of the general one $\langle \mathsf{CCS}_{\mathrm{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$. It is thus relatively easy to compare the server pre-orders while restricting the LTS at hand.

In the case of the client pre-orders, on the contrary, the alternative characterisations are tightly related to the LTS at hand. In general, when we change the LTS we should also change the alternative characterisations of $\underline{\sqsubseteq}_{\mathrm{CLT}}$ and $\sqsubseteq_{\mathrm{CLT}}$. It follows that the alternative relations $\precsim_{\mathrm{CLT}}$ and $\preceq_{\mathrm{CLT}}$ do give us a straightforward way to compare the client pre-orders, if we restrict the LTS at hand.

Structure of the chapter. In this chapter we study first the server pre-order (in Section 5.1), for it is the simplest of the relations in (5.1). The characterisation of $\sqsubseteq_{\mathrm{SVR}}$ is not too different from the characterisation of $\precsim_{\mathrm{SVR}}$; in fact, in certain LTSs the two relations coincide (see Section 5.1.1). In Section 5.2 we study the client pre-order, $\sqsubseteq_{\mathrm{CLT}}$. Its characterisation is reminiscent of the of $\sqsubseteq_{\mathrm{SVR}}$, although it requires us to use the notion of usability, and to adjust the definition of ready set. As we anticipated, the characterisations of $\sqsubseteq_{\mathrm{CLT}}$ and $\underline{\sqsubseteq}_{\mathrm{CLT}}$ do not aid us when it comes to comparing these two pre-orders on sub-LTSs of $\langle \mathsf{CCS}_{\mathrm{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$. We leave this as an open problem, that we partly address in the following chapters.

In Section 5.3 we study the peer pre-order $\sqsubseteq_{\mathrm{P2P}}$. In view of the structure of Definition 4.3.15, which is a nesting of $\precsim_{\mathrm{SVR}}$ into $\precsim_{\mathrm{CLT}}$, we directly define the alternative characterisation of $\sqsubseteq_{\mathrm{P2P}}$, and prove it sound and complete.

5.1 Server pre-order

In this section we study when, according to the compliance relation, a server p_2 is better than a server p_1. We define a compliance-based pre-order for servers, $\sqsubseteq_{\mathrm{SVR}}$, and we expose its characteristic properties.

Definition 5.1.1. [Compliance server pre-order]
We write $p_1 \sqsubseteq_{\mathrm{SVR}} p_2$ if and only if $r \dashv p_1$ implies $r \dashv p_2$ for every process r. We refer to the relation $\sqsubseteq_{\mathrm{SVR}}$ as the *compliance server pre-order*. □

Notation As usual, the operations $+$ and $\oplus$ are commutative and associative with respect to $=_{\mathrm{SVR}}$, so we are allows to use the notations $\sum$ and $\bigoplus$.

We already know a server pre-order, namely the MUST server pre-order (Definition 4.1.1). The alternative characterisation of $\underline{\sqsubseteq}_{\mathrm{SVR}}$, that is $\precsim_{\mathrm{SVR}}$, provides a touchstone to devise the characterisation of $\sqsubseteq_{\mathrm{SVR}}$. Indeed, we have to provide a characterisation of $\sqsubseteq_{\mathrm{SVR}}$, because the pre-orders $\underline{\sqsubseteq}_{\mathrm{SVR}}$ and $\sqsubseteq_{\mathrm{SVR}}$ are **not** comparable.

In the next two examples we expose the differences between $\underline{\sqsubseteq}_{\mathrm{SVR}}$ and $\sqsubseteq_{\mathrm{SVR}}$.

Example 5.1.2. [Convergence of servers]
In this example we prove that $\underline{\sqsubseteq}_{\mathrm{SVR}} \not\subseteq \sqsubseteq_{\mathrm{SVR}}$. Let $p_1 = \tau^\infty + \alpha.\beta.\,\mathsf{0}$ and $p_2 = \alpha.(\tau^\infty + \beta.\,\mathsf{0})$. The LTS

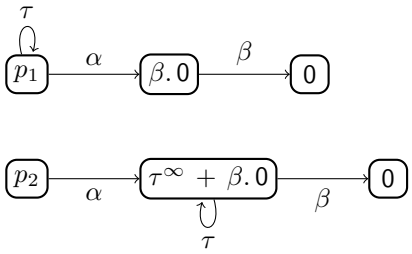


Figure 5.1: Processes used in Example 5.1.2; p_1 and p_2 are related by $\sqsubseteq_{\text{SVR}}$, but not by $\sqsubseteq_{\text{SVR}}$.

of these processes is depicted in Figure 5.1

We prove $p_1 \sqsubseteq_{\text{SVR}} p_2$. Since $p_1 \not\Downarrow$, for every $r \in \mathsf{CCS}_{\text{w}\tau}$, if p_1 MUST r then $r \xrightarrow{\checkmark}$; it follows that if p_1 MUST r then p_2 MUST r. However $p_1 \not\sqsubseteq_{\text{SVR}} p_2$. To prove this we have to exhibit a client r such that $r \dashv p_1$ and $r \not\dashv p_2$. Let $r = \tau^\infty + \overline{\alpha}.1$. To prove that $r \dashv p_1$ Definition 3.2.1 requires us to show a co-inductive compliance that contains the pair (r, p_1). The following relation will do, $\mathcal{R}= \{(r, p_1), (1, \beta.0)\}$. It is routine work to check that $\mathcal{R} \subseteq \mathcal{F}^{\dashv}(\mathcal{R})$. Will still have to prove that $r \not\dashv p_2$. Consider the ensuing computation

$$r \,||\, p_1 \Longrightarrow 1 \,||\, \tau^\infty + \beta.0$$

Since $1 \Downarrow$ and $\tau^\infty + \beta.0 \not\Downarrow$, Definition 3.2.1 ensures that $1 \not\dashv \tau^\infty + \beta.0$. It follows that $r \not\dashv p_2$. □

Example 5.1.2 exhibits a first difference between $\sqsubseteq_{\text{SVR}}$ and $\sqsubseteq_{\text{SVR}}$. The mismatch is of course due to the definitions of MUST and $\dashv$.

Let us discuss MUST. If *at any point* in a computation a server p diverges, then in that point the clients that p passes have to be successful (Lemma 3.1.6). This means that convergence has to be accounted for *along* the execution of every trace. The requirement of checking the converges of a server *along* traces is expressed by the predicate $\Downarrow$ in Definition 4.1.17.

The relation $\dashv$ imposes a weaker requirement; for for $\dashv$ the convergence of servers matters only as long as clients converge. So if $r \dashv p$ and r converges only after a trace s, then p is required to converge only *after* $\overline{s}$; the converge of p *along* $\overline{s}$ does not matter.

In Example 5.1.2 the server p_1 diverges *along* the trace $\alpha\beta$, because $p_1 \not\Downarrow$, but not *after*, for $p \xrightarrow{\alpha} \beta.0 \xrightarrow{\beta} 0$, and $0 \Downarrow$. The divergence along $\alpha\beta$ implies that $p \not\Downarrow s$ for every s, so in the MUST setting p_1 can be replaced by any server. This is not true in the compliance setting, because if $r \dashv p_1$, then the requests of r have to be satisfied by p_1 also after α and after $\alpha\beta$.

The second difference between $\sqsubseteq_{\text{SVR}}$ and $\sqsubseteq_{\text{SVR}}$ is how infinite traces as treated.

Example 5.1.3. [Infinite traces]
In this example we prove that the processes in Figure 4.4 are related by $\sqsubseteq_{\text{SVR}}$: $p \sqsubseteq_{\text{SVR}} q$.

We have to prove that if $r \dashv p$ then $r \dashv q$. Suppose that $r \dashv p$. Definition 3.2.1 requires us to exhibit a relation $\mathcal{R}$ such that (a) $r \,\mathcal{R}\, q$ and (b) $\mathcal{R}$ is a prefixed point of the rule functional $\mathcal{F}^{\dashv}$. Let

$$\mathcal{R} = \{\, (r', q') \mid r \stackrel{\overline{u}_k}{\Longrightarrow} r',\, q \stackrel{u_k}{\Longrightarrow} q' \text{ for every } k \in \mathbb{N} \,\}$$

Since $r \stackrel{\varepsilon}{\Longrightarrow} r$ and $q \stackrel{\varepsilon}{\Longrightarrow} q$, by definition $r \,\mathcal{R}\, q$; this proves (a).

To prove (b) we have to show that if $r' \,||\, q'$ then the ensuing properties are true,

i) if $r' \Downarrow$ then $q' \Downarrow$

ii) if $r' \,||\, q' \stackrel{\tau}{\not\longrightarrow}$ then $r \xrightarrow{\checkmark}$

iii) if $r' \parallel q' \xrightarrow{\tau} r'' \parallel q''$ then $r'' \mathcal{R} q''$

Pick a pair (r', q') from the relation $\mathcal{R}$.

We prove i). The definition of $\mathcal{R}$ implies that $q \stackrel{u_k}{\Longrightarrow} q'$ for some $k \in \mathbb{N}$, so Lemma 4.1.7 implies that $q' \Downarrow$. Since the consequences of the implication in i) are true, the whole implication is true.

We prove ii). Suppose that $r' \parallel q' \stackrel{\tau}{\not\longrightarrow}$. The definition of $\mathcal{R}$ implies that for some $k \in \mathbb{N}$, $r \stackrel{\overline{u}_k}{\Longrightarrow} r'$, and the assumption $r' \parallel q' \stackrel{\tau}{\not\longrightarrow}$ implies that $r' \stackrel{\tau}{\not\longrightarrow}$. By hypothesis $p \stackrel{u_k}{\Longrightarrow} \mathsf{0}$, so there exists the computation

$$r \parallel p \Longrightarrow r' \parallel \mathsf{0} \stackrel{\tau}{\not\longrightarrow}$$

The assumption that $r \dashv p$ and Corollary 3.2.7 imply that $r' \dashv \mathsf{0}$. Now $r' \parallel \mathsf{0}$ and point (b) of Definition 3.2.1 implies that $r' \xrightarrow{\checkmark}$. This proves ii).

We prove iii). Suppose that $r' \parallel q' \xrightarrow{\tau} r'' \parallel q''$; we show why $r'' \mathcal{R} q''$. By construction of $\mathcal{R}$ we know that $r \stackrel{\overline{u}_k}{\Longrightarrow} r'$ and $q \stackrel{u_k}{\Longrightarrow} q''$ for some $k \in \mathbb{N}$. The argument is a case analysis on the rule used to infer the reduction. If rule [P-Left] was applied then $r' \xrightarrow{\tau} r''$ and $q' = p''$; as $r \stackrel{\overline{u}_k}{\Longrightarrow} r''$ the definition of $\mathcal{R}$ implies that $r'' \mathcal{R} q''$. If rule [P-Right] was applied then $q' \xrightarrow{\tau} q''$ and $r' = r''$. We infer $q \stackrel{u_k}{\Longrightarrow} q''$, so the definition of $\mathcal{R}$ implies that $r'' \mathcal{R} q''$. If rule [P-Synch], then the reduction is due to an interaction. The only actions that the q' offer are in u, so it must be the case that $q \stackrel{u_k\alpha}{\Longrightarrow} q''$, $r \stackrel{\overline{u}_k\overline{\alpha}}{\Longrightarrow} r''$ and $u_k\alpha = u_{k+1}$. The definition of $\mathcal{R}$ implies that $r'' \mathcal{R} q''$.

We have proven that $\mathcal{R}$ is a prefixed point of $\mathcal{F}^{\dashv}$, so Definition 3.2.1 and the Knaster-Tarski theorem imply that $\mathcal{R} \subseteq \dashv$. Now (a) implies that $r \dashv q$. □

Example 5.1.3 shows that the relation $\sqsubseteq_{\text{SVR}}$ does not compare infinite traces as $\sqsubseteq_{\text{SVR}}$ does.

Again, the difference between $\sqsubseteq_{\text{SVR}}$ and $\sqsubseteq_{\text{SVR}}$ stems from the definitions of $\dashv$ and of MUST. To check that p MUST r, one has to prove that *every* maximal computation of $r \parallel p$, is client-successful. To check that $r \dashv p$, on the other hand, one has to check only that *finite* computations either end in a client-successful state or can be extended.

We laid bare the differences between server pre-orders $\sqsubseteq_{\text{SVR}}$ and $\sqsubseteq_{\text{SVR}}$, thereby proving that these pre-orders are not comparable,

$$\sqsubseteq_{\text{SVR}} \not\subseteq \sqsubseteq_{\text{SVR}}, \qquad \sqsubseteq_{\text{SVR}} \not\subseteq \sqsubseteq_{\text{SVR}} \tag{5.3}$$

These differences show that in the compliance setting

i) the convergence of servers is compared only after traces have been performed

ii) the divergence of a server does not imply that it is worse than the other servers

iii) the infinite traces of servers do not matter

Our task now is to adapt Definition 4.1.17 so as to characterise the relation $\sqsubseteq_{\text{SVR}}$. The properties we listed previously sheds light on what to do. Point (i) above calls for the definition of a new predicate to check convergence. Point (ii) suggests that we drop the requirement $p_1 \Downarrow w$ from condition (2) of Definition 4.1.17. Point (iii) suggests that we drop infinite traces from condition (2) of Definition 4.1.17.

We introduce the novel predicate to check the convergence of servers.

Definition 5.1.4. [Convergence after trace]
Let $\mathcal{F}_{\Downarrow\Downarrow} : \mathcal{P}(\mathsf{CCS}_{\text{w}\tau} \times Act^\star) \longrightarrow \mathcal{P}(\mathsf{CCS}_{\text{w}\tau} \times Act^\star)$ be the rule functional given by the inference rules in Figure 5.2. Lemma C.0.24 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_{\Downarrow\Downarrow}(X)$; we call this solution the *weak convergence predicate*, and we denote it $\Downarrow\Downarrow$: That is $\Downarrow\Downarrow = \mu X.\mathcal{F}_{\Downarrow\Downarrow}(X)$. □

$$\frac{}{p \Downarrow\!\Downarrow \varepsilon}\ p \Downarrow\ ;\ [\text{WCONV-AX}]$$

$$\frac{}{p \Downarrow\!\Downarrow \alpha s}\ p \overset{\alpha}{\Longrightarrow\!\!\!\!\!/}\ ;\ [\text{WCONV-AX-NOT}]$$

$$\frac{\bigoplus(p \text{ AFTER } \alpha) \Downarrow\!\Downarrow s'}{p \Downarrow\!\Downarrow \alpha s'}\ p \overset{\alpha}{\Longrightarrow}\ ;\ [\text{WCONV-ALPHA}]$$

Figure 5.2: Inference rules for the functional $\mathcal{F}_{\Downarrow\!\Downarrow}$

As the name suggests the predicate $\Downarrow\!\Downarrow$ is weaker than $\Downarrow$, in the sense that $p \Downarrow\!\Downarrow s$ does not imply $p \Downarrow s$. We show this in Example 5.1.5.

By using $\Downarrow\!\Downarrow$ in place of $\Downarrow$ we relax the condition under which the comparison between ready sets has to be performed; the result is that the amended definition checks more ready sets than the wrong definition.

Example 5.1.5. In this example we show a process p such that $p \Downarrow\!\Downarrow \alpha\beta$ and $p \not\Downarrow \alpha\beta$. Let $p' = \tau^\infty \oplus \beta.\,\mathsf{0}$ and $p = \tau^\infty + \alpha.p'$. The proof that $p \Downarrow\!\Downarrow \alpha\beta$ is the following inference tree

$$\dfrac{\dfrac{\dfrac{}{\mathsf{0} \Downarrow \varepsilon}\ \mathsf{0} \Downarrow\ ;\ [\text{WCONV-AX}]}{p' \Downarrow\!\Downarrow \beta}\ p' \overset{\beta}{\Longrightarrow}\ ;\ [\text{WCONV-ALPHA}]}{p \Downarrow\!\Downarrow \alpha\beta}\ p \overset{\alpha}{\Longrightarrow}\ ;\ [\text{WCONV-ALPHA}]$$

To derive $p \Downarrow \alpha\beta$ we need an inference tree like the above one. The tree for $\Downarrow$, though, does not exist because $p' \not\Downarrow$, so after the axiom [CONV-AX] the derivation cannot proceed. □

The previous example shows the difference between $\Downarrow\!\Downarrow$ and $\Downarrow$. Suppose that $p \overset{s}{\Longrightarrow}$. If $p \Downarrow\!\Downarrow s$ then p converges after the string s. If $p \Downarrow s$ then all the states in all the branches encountered while performing s must converge.

We prove a technicality.

Lemma 5.1.6. For every $s \in Act^\star$ and every $p \in \mathsf{CCS}_{\mathrm{w}\tau}$, if $p \Downarrow\!\Downarrow s$ and $p \xrightarrow{\tau} p'$, then $p' \Downarrow\!\Downarrow s$.

Proof. The argument is by induction on s.

Base case ($s = \varepsilon$) In this case we want to prove that $p' \Downarrow\!\Downarrow \varepsilon$. In view of [WCONV-AX], it is enough to prove that $p' \Downarrow$. By hypothesis there exists the derivation of $p \Downarrow\!\Downarrow \varepsilon$, we can be only the axiom [WCONV-AX]. The side conditions of the axiom imply that $p \Downarrow$. The hypothesis that $p \Longrightarrow p'$ implies that $p' \Downarrow$.

Inductive case ($s = \alpha s'$) We have to prove that $p' \Downarrow\!\Downarrow \alpha s'$. If $p' \overset{\alpha}{\Longrightarrow\!\!\!\!\!/}$, the we use the second axiom of $\Downarrow\!\Downarrow$,

$$\frac{}{p' \Downarrow\!\Downarrow \alpha s'}\ p' \overset{\alpha}{\Longrightarrow\!\!\!\!\!/}\ ;\ [\text{WCONV-AX-NOT}]$$

If $p' \overset{\alpha}{\Longrightarrow}$, then $p \overset{\alpha}{\Longrightarrow}$. The last fact and the hypothesis $p \Downarrow\!\Downarrow \alpha s'$ imply that $\bigoplus(p \text{ AFTER } \alpha) \Downarrow\!\Downarrow s'$. Since $(p' \text{ AFTER } \alpha) \subseteq (p \text{ AFTER } \alpha)$, it follows that $\bigoplus(p' \text{ AFTER } \alpha) \Downarrow\!\Downarrow s'$, and so we derive

$$\frac{\bigoplus(p' \text{ AFTER } \alpha) \Downarrow\!\Downarrow s'}{p' \Downarrow\!\Downarrow \alpha s'}\ p' \overset{\alpha}{\Longrightarrow}\ ;\ [\text{WCONV-ALPHA}]$$

□

We are ready to define a relation that characterises $\sqsubseteq_{\text{SVR}}$.

Definition 5.1.7. [Semantic compliance server pre-order]
Let $p_1 \preceq_{\text{SVR}} p_2$ whenever for every $s \in Act^\star$,

(1) if $p_1 \Downarrow\!\Downarrow s$ then

(a) $p_2 \Downarrow\!\Downarrow s$

(b) for every $B \in \text{ACC}(p_2, s)$ there exists some $A \in \text{ACC}(p_1, s)$ such that $A \subseteq B$

(2) if $p_2 \stackrel{s}{\Longrightarrow}$, then $p_1 \stackrel{s}{\Longrightarrow}$ □

To prove that $\preceq_{\text{SVR}}$ is a complete description of $\sqsubseteq_{\text{SVR}}$, we have to explain why $\sqsubseteq_{\text{SVR}}$ satisfied all the properties required in Definition 5.1.7. We carry out this task in a series of lemmas.

For every process p there exists a client that is not satisfied by p, namely $\mathsf{0}$. This implies the following result.

Lemma 5.1.8. [Finite trace simulation]
For every $s \in Act^\star$, and every $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, if $p_1 \sqsubseteq_{\text{SVR}} p_2$ and $p_2 \stackrel{s}{\Longrightarrow}$ then $p_1 \stackrel{s}{\Longrightarrow}$.

Proof. By hypothesis there exists a p_2' such that $p_2 \stackrel{s}{\Longrightarrow} p_2'$. Let $s = \alpha_1\alpha_2 \ldots \alpha_n$, and let

$$C_i \stackrel{\text{def}}{=} \begin{cases} \tau^\infty + \overline{\alpha}_i.C_{i+1} & \text{if } i < n \\ \mathsf{0} & \text{if } i = n \end{cases}$$

The definition of $\dashv$ lets us prove that $\mathsf{0} \not\dashv p_2'$, and also that $C_0 \not\dashv p_2$. The hypothesis $p_1 \sqsubseteq_{\text{SVR}} p_2$ implies that $C \not\dashv p_1$. If $p_1 \stackrel{s}{\not\Longrightarrow}$, then the divergence of all the C_i but C_n let us prove that $C_0 \dashv p_1$. It follows that $p_1 \stackrel{s}{\Longrightarrow}$. □

does not extend to traces that involve infinite states; we have proven this in Example 5.1.3.

The relation between the convergence of the servers in $\sqsubseteq_{\text{SVR}}$ is proven in the next lemma.

Lemma 5.1.9. For every $s \in Act^\star$ and $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, if $p_1 \sqsubseteq_{\text{SVR}} p_2$ and $p_1 \Downarrow\!\Downarrow s$ then $p_2 \Downarrow\!\Downarrow s$.

Proof. Fix a string $s \in Act^\star$ such that $p_1 \Downarrow\!\Downarrow s$. We have to show a finite derivation of $p_2 \Downarrow\!\Downarrow s$. the proof is by induction on s.

Base case ($s = \varepsilon$) We have to derive $p_1 \Downarrow\!\Downarrow \varepsilon$. The hypothesis that $p_1 \Downarrow\!\Downarrow \varepsilon$ implies that $p_1 \Downarrow$. Lemma 3.2.4 ensures that $1 \dashv p_1$, so the hypothesis $p_1 \sqsubseteq_{\text{SVR}} p_2$ implies that $1 \dashv p_2$. Definition 3.2.1 ensures that $p_2 \Downarrow$, thus we derive

$$\frac{}{p_2 \Downarrow\!\Downarrow \varepsilon}\ p_2 \Downarrow\ ;[\text{WCONV-AX}]$$

Inductive case ($s = \alpha s'$) We have to derive $p_2 \Downarrow\!\Downarrow \alpha s'$. If $p_2 \stackrel{\alpha}{\not\longrightarrow}$ then the derivation is the following one

$$\frac{}{p_2 \Downarrow\!\Downarrow \alpha s'}\ p_2 \stackrel{\alpha}{\not\Longrightarrow};[\text{WCONV-AX-NOT}]$$

Suppose that $p_2 \stackrel{\alpha}{\Longrightarrow}$. In this case if we knew that $\bigoplus(p_2 \text{ AFTER } \alpha) \Downarrow\!\Downarrow s'$, then we could apply [WCONV-ALPHA] to obtain the derivation we are after.

We prove that $\bigoplus(p_2 \text{ AFTER } \alpha) \Downarrow\!\Downarrow s'$. Since s' is smaller than s, the inductive hypothesis states that

$$\text{for every } \hat{p}_1 \sqsubseteq_{\text{SVR}} \hat{p}_2, \text{ if } \hat{p}_1 \Downarrow\!\Downarrow s' \text{ then } \hat{p}_2 \Downarrow\!\Downarrow s'$$

Let $\hat{p}_2 = \bigoplus(p_2 \text{ AFTER } \alpha)$; to show that $\hat{p}_1 \Downarrow\!\Downarrow s'$ it suffices to exhibit a $\hat{p}_1$ such that $\hat{p}_1 \sqsubseteq_{\text{SVR}} \hat{p}_2$ and $\hat{p}_1 \Downarrow\!\Downarrow s'$. The assumption $p_2 \stackrel{\alpha}{\Longrightarrow}$, the hypothesis $p_1 \sqsubseteq_{\text{SVR}} p_2$ and Lemma 5.1.8 imply that $p_1 \stackrel{\alpha}{\Longrightarrow}$, so the set ($p1$ AFTER α) is non-empty. Let $\hat{p}_1 = \bigoplus(p_1 \text{ AFTER } \alpha)$. The hypothesis $p_1 \Downarrow\!\Downarrow \alpha s'$ and $p_1 \stackrel{\alpha}{\Longrightarrow}$

imply that $\hat{p}_1 \Downarrow s'$. We still have to prove that $\hat{p}_1 \sqsubseteq_{\text{SVR}} \hat{p}_2$. Fix a process r such that $r \dashv \hat{p}_1$. It is relatively easy to prove that

$$\tau^\infty + \overline{\alpha}.r \dashv p_1 \tag{5.4}$$

so the hypothesis $p_1 \sqsubseteq_{\text{SVR}} p_2$ implies that $\tau^\infty + \overline{\alpha}.r \dashv p_2$. For every $p_2' \in (p_2 \text{ AFTER } \alpha)$, the computation $\tau^\infty + \overline{\alpha}.r \mid\mid p_2 \Longrightarrow r \mid\mid p_2'$ and Corollary 3.2.7 let us prove that $r \dashv p_2'$. It follows that $r \dashv \hat{p}_2$.

We have proven enough facts to use the inductive hypothesis, which implies that $\hat{p}_2 \Downarrow s'$. Now we derive

$$\dfrac{\dfrac{\vdots}{\hat{p}_2 \Downarrow s'}}{p_2 \Downarrow \alpha s'}\ p_2 \overset{\alpha}{\Longrightarrow};\ [\text{WCONV-ALPHA}]$$

□

Corollary 5.1.10. *For every $s \in Act^\star$, and $p_1, p_2 \in \mathsf{CCS}_{w\tau}$, if $p_1 \sqsubseteq_{\text{SVR}} p_2$, $p_1 \Downarrow s$ and $\text{ACC}(p_2, s) \neq \emptyset$ then $\text{ACC}(p_1, s) \neq \emptyset$.*

Proof. The hypothesis $\text{ACC}(p_2, s) \neq \emptyset$ implies that $p_2 \overset{s}{\Longrightarrow}$. Lemma 5.1.8 implies that $p_1 \overset{s}{\Longrightarrow} p_1'$ for some p_1'. The hypothesis $p_1 \Downarrow s$ implies that $p_1' \Downarrow$, and so there exists a p_1'' such that $p_1 \overset{s}{\Longrightarrow} p_1'' \overset{\tau}{\not\longrightarrow}$. This implies that $S(p_1'') \in \text{ACC}(p_1, s)$. □

Lemma 5.1.11. For every $s \in Act^\star$, and every $p_1, p_2 \in \mathsf{CCS}_{w\tau}$, if $p_1 \sqsubseteq_{\text{SVR}} p_2$, $p_1 \Downarrow s$ and $B \in \text{ACC}(p_2, s)$, then there exists a set $A \in \text{ACC}(p_1, s)$ such that $A \subseteq B$.

Proof. Fix a $s \in Act^\star$ and a set $B \in \text{ACC}(p_2, s)$. We have to exhibit a set $A \in \text{ACC}(p_1, s)$ that is a subset of B. Corollary 5.1.10 and the hypothesis of this lemma imply that the set is $\text{ACC}(p_1, s)$ is non-empty, that is $\text{ACC}(p_1, s) = \{ A_i \mid i \in I \}$ where I is some non-empty set.

The proof is by contradiction; we assume the following

$$\text{for every } i \in I, \text{ there exists a } \alpha_i \in A_i \text{ such that } \alpha_i \notin B.$$

By using this assumption we define a client C that contradicts the hypothesis $p_1 \sqsubseteq_{\text{SVR}} p_2$. Let $s = \beta_1 \beta_2 \ldots \beta_n$. Let

$$C_i \overset{\text{def}}{=} \begin{cases} \tau^\infty + \overline{\beta}_{i+1}.C_{i+1} & \text{if } i < n, \\ \sum_{i \in I} \overline{\alpha}_i.\tau^\infty & \text{if } i = n \end{cases}$$

We prove that $C_0 \not\dashv p_2$. Definition 4.1.11 and the hypothesis $B \in \text{ACC}(p_2, s)$ imply that there exists a p_2' such that $p_2 \overset{s}{\Longrightarrow} p_2' \overset{\tau}{\not\longrightarrow}$ and $S(p_2') = B$. Consider the computation, $C_0 \mid\mid p_2 \Longrightarrow C_n \mid\mid p_2' \overset{\tau}{\not\longrightarrow}$. As $C_n \overset{\checkmark}{\not\longrightarrow}$, $C_n \not\dashv p_2$; it follows that $C_0 \not\dashv p_2$.

We prove that $C_0 \dashv p_1$. Let

$$\mathcal{R} = \{ (C, p) \mid C_0 \overset{\overline{s'}}{\Longrightarrow} C, p_1 \overset{s'}{\Longrightarrow} p \text{ with } s' \text{ prefix of } s \} \cup \{ (\tau^\infty, p) \mid p_1 \overset{s\alpha_i}{\Longrightarrow} p \}$$

By construction $C_0 \mathcal{R} p_1$. We prove that the relation $\mathcal{R}$ is a co-inductive compliance. Fix a pair in $\mathcal{R}$, say (C, p); Definition 3.2.1 requires us to prove three properties

a) if $C \Downarrow$ then $p \Downarrow$

b) if $C \mid\mid p \overset{\tau}{\not\longrightarrow}$ then $C \overset{\checkmark}{\longrightarrow}$

c) if $C \mid\mid p \overset{\tau}{\longrightarrow} C' \mid\mid p'$ then $C' \mid\mid p'$

If $C = \tau^\infty$ and $p_1 \overset{s\alpha_i}{\Longrightarrow} p$, then the pair (C, p) satisfies all the points above. This is true because $C \not\Downarrow$, $C \mid\mid p \overset{\tau}{\not\longrightarrow}$, and if $C \mid\mid p \overset{\tau}{\longrightarrow} C' \mid\mid'$, then $C' = \tau^\infty$ and $p_1 \overset{s\alpha_i}{\Longrightarrow} p'$.

Now suppose that C and p are in $\mathcal{R}$ because for some s' prefix of s, $C_0 \stackrel{\overline{s'}}{\Longrightarrow} C$ and $p_1 \stackrel{s'}{\Longrightarrow} p$.

Suppose that $C \Downarrow$; then $C = C_n$ and $s' = s$; it follows that $p_1 \stackrel{s}{\Longrightarrow} p$. The hypothesis $p_1 \downdownarrows s$ implies that $p \Downarrow$. This proves a).

To prove that b) is true we show that $C \parallel p \stackrel{\tau}{\longrightarrow}$. If s' is shorter than s then $C \stackrel{\tau}{\longrightarrow}$ because C diverges. If $s' = s$ then $C = \sum_{i \in I} \overline{\alpha}_i.\tau^\infty$; since $p_1 \stackrel{s}{\Longrightarrow} p$, the hypothesis $\bigoplus(p_1 \text{ AFTER } s) \Downarrow$ implies that $p \Downarrow$, and so Definition 4.1.11 ensures that $S(p) \in \text{ACC}(p_1, s)$. It follows that for some $i \in I$, there is an action $\alpha_i \in S(p)$ such that $C \stackrel{\overline{\alpha}}{\longrightarrow}$. It follows that C and p can interact, so $C \parallel p \stackrel{\tau}{\longrightarrow}$.

We have to discuss c). Suppose that $C \parallel p \stackrel{\tau}{\longrightarrow} C' \parallel p'$. The argument depends on the rule used to infer the reduction. If [P-LEFT] or [P-RIGHT] was used, then $C \stackrel{\overline{s'}}{\Longrightarrow} C'$ or $p_1 \stackrel{s'}{\Longrightarrow} p'$. If [P-SYNCH] was applied then there exists the inference

$$\frac{C \stackrel{\overline{\delta}}{\longrightarrow} C' \quad p \stackrel{\delta}{\longrightarrow} p'}{C \parallel p \stackrel{\tau}{\longrightarrow} C' \parallel p'} \text{ [P-SYNCH]}$$

If $\delta = \beta_i$ for some β_i in s, then $C' \mathcal{R} p'$. If $\delta = \alpha_i$ for some α_i, then $C' = \tau^\infty$ and $p_1 \stackrel{s\alpha_i}{\Longrightarrow} p'$, so $C' \mathcal{R} p'$. □

The proof that the relation $\preceq_{\text{SVR}}$ is a complete description of $\sqsubseteq_{\text{SVR}}$ is now easy.

Proposition 5.1.12. [Completeness]
For every $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, $p_1 \sqsubseteq_{\text{SVR}} p_2$ implies $p_1 \preceq_{\text{SVR}} p_2$.

Proof. It follows from Lemma 5.1.9, Lemma 5.1.11, and Lemma 5.1.8. □

In order to prove the soundness of $\preceq_{\text{SVR}}$ with respect to $\sqsubseteq_{\text{SVR}}$, we need the following lemmas.

Lemma 5.1.13. For every $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, if $p_1 \preceq_{\text{SVR}} p_2$ and $p_2 \stackrel{\tau}{\longrightarrow} p_2'$, then $p_1 \preceq_{\text{SVR}} p_2'$.

Proof. Definition 5.1.7 requires us to prove three properties of the pair (p_1, p_2'), namely that for every $s \in Act^\star$,

a) if $p_1 \downdownarrows s$ then

   i) $p_2 \downdownarrows s$

   ii) if $B \in \text{ACC}(p_2', s)$ then there exists a set $A \in \text{ACC}(p_1, s)$ such that $A \subseteq B$

b) if $p_2' \stackrel{s}{\Longrightarrow}$, then $p_1 \stackrel{s}{\Longrightarrow}$

We prove these properties one by one. Fix a $s \in Act^\star$.

a) Suppose that $p_1 \downdownarrows s$.

   i) We have to show that $p_2' \downdownarrows s$. The hypothesis $p_1 \preceq_{\text{SVR}} p_2$, the assumption $p_1 \downdownarrows s$ and point (1a) of Definition 5.1.7 imply that $p_2 \downdownarrows s$. Lemma 5.1.6 implies that $p_2' \downdownarrows s$.

   ii) Suppose that $B \in \text{ACC}(p_2', s)$. Definition 4.1.11 ensures that $B \in \text{ACC}(p_2, s)$, and so the hypothesis $p_1 \preceq_{\text{SVR}} p_2$ and Definition 4.1.17 imply that there exists a set $A \in \text{ACC}(p_1, s)$ such that $A \subseteq B$.

b) Suppose that $p_2' \stackrel{s}{\Longrightarrow} p_2''$. It follows that $p_2 \stackrel{s}{\Longrightarrow} p_2''$, and so the hypothesis $p_1 \preceq_{\text{SVR}} p_2$ and point (2) of Definition 5.1.7 imply that $p_1 \stackrel{s}{\Longrightarrow}$. □

□

Lemma 5.1.14. For every $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, if $p_1 \downdownarrows \alpha$, $p_1 \preceq_{\text{SVR}} p_2$ and $p_2 \stackrel{\alpha}{\longrightarrow} p_2'$ then

i) the set $(p_1 \text{ AFTER } \alpha)$ is non-empty

ii) $\bigoplus(p_1 \text{ AFTER } \alpha) \preceq_{\text{SVR}} p_2'$.

Proof. Suppose that for some $\alpha \in Act$ and p_2', $p_2 \xrightarrow{\alpha} p_2'$. The hypothesis $p_1 \preceq_{\text{SVR}} p_2$ and point (2) of Definition 5.1.7 implies that $p \xrightarrow{\alpha}$. Definition 4.1.5 implies that $p_1' \in (p_1 \text{ AFTER } \alpha)$.

We prove point (ii). Point (i) guarantees that the process $\bigoplus(p_1 \text{ AFTER } \alpha)$ exists, and we have to show that $\bigoplus(p_1 \text{ AFTER } \alpha) \preceq_{\text{SVR}} p_2'$. Let $\hat{p} = \bigoplus(p_1 \text{ AFTER } \alpha)$. Definition 5.1.7 requires us to prove that for every $s \in Act^\star$, the processes $\hat{p}$ and p_2' enjoy the ensuing properties,

a) if $\hat{p} \Downarrow s$ then

  i) $p_2' \Downarrow s$

  ii) if $B \in \text{ACC}(p_2', s)$ then there exists a set $A \in \text{ACC}(\hat{p}, s)$ such that $A \subseteq B$

b) if $p_2' \stackrel{s}{\Longrightarrow} p_2''$, then $\hat{p} \stackrel{s}{\Longrightarrow}$

Fix a string $s \in Act^\star$ such that $\hat{p} \Downarrow s$. To prove point (ai), we have to explain why $p_2'' \Downarrow s$. The assumption on $\hat{p}$ ensures that we can derive

$$\frac{\hat{p} \Downarrow s}{p_1 \Downarrow \alpha s} \; p_1 \stackrel{\alpha}{\Longrightarrow}; \; [\text{WCONV-ALPHA}]$$

Since $p_1 \preceq_{\text{SVR}} p_2$, point (1a) of Definition 5.1.7 implies that $p_2 \Downarrow \alpha s$. Since $p_2 \stackrel{\alpha}{\Longrightarrow}$, it follows that $\bigoplus(p_2 \text{ AFTER } \alpha) \Downarrow s$. Since $\{p_2'\} \subseteq (p_2 \text{ AFTER } \alpha)$, it follows that $p_2' \Downarrow s$.

We prove point (aii). Fix a $B \in \text{ACC}(p_2', s)$; we have already proven that $p_1 \Downarrow \alpha s$, so $p_2 \xrightarrow{\alpha} p_2'$ and Definition 4.1.11 ensure that $B \in \text{ACC}(p_2, \alpha s)$. Now point (1b) of Definition 5.1.7 implies that there exists a set $A \in \text{ACC}(p_1, \alpha s)$ such that $A \subseteq B$. The equality $\text{ACC}(p_1, \alpha s) = \text{ACC}(\hat{p}, s)$ implies that $A \in \text{ACC}(\hat{p}, s)$.

To prove point (b) suppose that $p_2' \stackrel{s}{\Longrightarrow} p_2''$. The hypothesis $p_1 \preceq_{\text{SVR}} p_2$, $p_2 \stackrel{\alpha s}{\Longrightarrow} p_2''$ and point (2) of Definition 5.1.7 imply that $p_1 \stackrel{\alpha s}{\Longrightarrow}$. In turn this implies that $\hat{p} \stackrel{s}{\Longrightarrow}$.

□

Theorem 5.1.15. [Alternative characterisation $\sqsubseteq_{\text{SVR}}$]
For every $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, $p_1 \sqsubseteq_{\text{SVR}} p_2$ if and only if $p_1 \preceq_{\text{SVR}} p_2$.

Proof. We have to prove the ensuing implications,

i) if $p_1 \sqsubseteq_{\text{SVR}} p_2$ then $p_1 \preceq_{\text{SVR}} p_2$

ii) if $p_1 \preceq_{\text{SVR}} p_2$ then $p_1 \sqsubseteq_{\text{SVR}} p_2$

The first implication is shown in Proposition 5.1.12. We prove only the second implication.

We are required to show that for every r, if $r \dashv p_1$ then $r \dashv p_2$. To this end we define a suitable co-inductive compliance relation. Let

$$\mathcal{R} = \{ (r, p_2) \mid p_1 \preceq_{\text{SVR}} p_2, r \dashv p_1, \text{ for some } r \in \mathsf{CCS}_{\text{w}\tau} \}$$

The construction of $\mathcal{R}$ implies that if $r \dashv p_1$ and $p_1 \preceq_{\text{SVR}} p_2$, then $r \mathcal{R} p_2$. To show that $r \dashv p_2$ we have to prove that $\mathcal{R}$ is a co-inductive compliance. Definition 3.2.1 requires us to prove that if $r \mathcal{R} p$ then three properties are true,

(a) if $r \Downarrow$ then $p \Downarrow$

(b) if $r \mid\mid p \stackrel{\tau}{\not\longrightarrow}$ then $r \xrightarrow{\checkmark}$

(c) if $r \mid\mid p \xrightarrow{\tau} r' \mid\mid p'$ then $r' \mathcal{R} p'$

Fix some r and p such that $r \mathcal{R} p$. We prove the properties listed above. By definition of $\mathcal{R}$ there exists a process p_1 such that

$$p_1 \preceq_{\text{SVR}} p, \qquad r \dashv p_1$$

Suppose that $r \Downarrow$. Definition 3.2.1 implies that $p_1 \Downarrow$; in turn this means that $\bigoplus(p_1 \text{ AFTER } \varepsilon) \Downarrow$. point (1a) of Definition 5.1.7 and $p_1 \preceq_{\text{SVR}} p_2$ ensure that if $p \stackrel{\tau}{\Longrightarrow} p'$ then $p' \Downarrow$. By reflexivity, $p \stackrel{\varepsilon}{\Longrightarrow} p$, so $p \Downarrow$. We have proven (a).

Suppose that $r \parallel p \stackrel{\tau}{\not\longrightarrow}$. Then $S(p) \not\subseteq \overline{S(r)}$. Since $p \stackrel{\tau}{\not\longrightarrow}$, $p \Downarrow$, and so $S(p) \in \text{ACC}(p, \varepsilon)$. As r is stable, r is convergent, and so $r \dashv p_1$ implies that $p_1 \Downarrow$. It follows that $\bigoplus(p_1 \text{ AFTER } \varepsilon) \Downarrow$. point (1b) of $\preceq_{\text{SVR}}$ and the assumption $p_1 \preceq_{\text{SVR}} p$ imply that there exists a $A \in \text{ACC}(p_1, \varepsilon)$ such that $A \subseteq S(p)$. Definition 4.1.11 and Definition 4.1.6 imply that there exists a p_1' such that $p_1 \stackrel{\varepsilon}{\Longrightarrow} p_1' \stackrel{\tau}{\not\longrightarrow}$ and $S(p_1') \subseteq A$. It follows that $S(p_1') \subseteq S(p)$, and so $S(p_1') \not\subseteq \overline{S(r)}$. The last fact and r and p_1' are stable, imply that $r \parallel p_1' \stackrel{\tau}{\not\longrightarrow}$. Since $r \parallel p_1 \Longrightarrow r \parallel p_1'$ Corollary 3.2.7 implies that $r \dashv p_1'$; Definition 3.2.1 now ensures that $r \stackrel{\checkmark}{\longrightarrow}$.

Suppose that $r \parallel p \stackrel{\tau}{\longrightarrow} r' \parallel p'$; we have to explain why $r' \mathcal{R} p'$. The definition of $\mathcal{R}$ requires us to show a $\hat{p}$ such that

$$r' \dashv \hat{p}, \qquad \hat{p} \preceq_{\text{SVR}} p'$$

The argument depends on the rule used to infer the reduction.

If the reduction is due to rule [P-LEFT], then $r \stackrel{\tau}{\longrightarrow} r'$, and $p' = p$. Let $\hat{p} = p_1$. Definition 3.2.1 and $r \dashv p_1$ ensure that $r' \dashv \hat{p}$, and $\hat{p} \preceq_{\text{SVR}} p'$, so $r' \mathcal{R} p'$.

If the reduction is due to rule [P-RIGHT], then $p \stackrel{\tau}{\longrightarrow} p'$, and $r = r'$; the candidate $\hat{p}$ is p_1. We know that $p_1 \preceq_{\text{CLT}} p$, thus Lemma 5.1.13 implies that $p_1 \preceq_{\text{SVR}} p'$; and we already know that $r' \dashv p_1$, because $r = r'$.

If the reduction is due to rule [P-SYNCH], the $r \stackrel{\alpha}{\longrightarrow} r'$, $p \stackrel{\overline{\alpha}}{\longrightarrow} p'$. the fact that $p_1 \preceq_{\text{SVR}} p$ and point (2) of Definition 5.1.7 implies that $p_1 \stackrel{\overline{\alpha}}{\Longrightarrow}$. We can use Corollary 3.2.7 to prove that $r' \dashv \bigoplus(p_1 \text{ AFTER } \alpha)$, so let our candidate $\hat{p}$ be $\bigoplus(p_1 \text{ AFTER } \alpha)$. We have to prove that $\hat{p} \preceq_{\text{SVR}} p'$; this follows from Lemma 5.1.14. □

Comparison with other pre-orders

In Chapter 4 we have studied three pre-orders, and so far we have compared $\sqsubseteq_{\text{SVR}}$ only with one of them, namely $\sqsubseteq_{\text{SVR}}$. By exposing the differences between the server pre-orders, we have justified the characterisation of $\sqsubseteq_{\text{SVR}}$.

Now we swiftly prove that $\sqsubseteq_{\text{SVR}}$ is comparable neither with the other MUST pre-orders, nor with a well-known pre-order from the literature, the shd testing pre-order.

The following inequalities should not be surprising.

$$\begin{array}{rcl} \alpha.0 & \sqsubseteq_{\text{CLT}} & \beta.0 \\ 1 & \not\sqsubseteq_{\text{CLT}} & 0 \\ \alpha.1 + \beta.0 & \sqsubseteq_{\text{P2P}} & \alpha.1 \end{array} \qquad \begin{array}{rcl} \alpha.0 & \not\sqsubseteq_{\text{SVR}} & \beta.0 \\ 1 & \sqsubseteq_{\text{SVR}} & 0 \\ \alpha.1 + \beta.0 & \not\sqsubseteq_{\text{SVR}} & \alpha.1 \end{array}$$

The inequalities above are true because of the non trivial usability of clients (and of peers alike). Intuitively, non-usable clients are always equivalent, even if they offer different communication pattern. The last fact lets $\sqsubseteq_{\text{SVR}}$ distinguish the clients. Moreover, it is safe to remove from a peer a non-usable action and the derivation after it. This, though, is detected by $\sqsubseteq_{\text{SVR}}$, because the all the (co)actions offered by servers can be used by clients. Since $\sqsubseteq_{\text{SVR}} \not\subseteq \sqsubseteq_{\text{CLT}}$, the set inclusion $\sqsubseteq_{\text{P2P}} \subseteq \sqsubseteq_{\text{CLT}}$ implies that $\sqsubseteq_{\text{SVR}} \not\subseteq \sqsubseteq_{\text{P2P}}$.

Now we turn our attention to the setting of shd testing by [Rensink and Vogler, 2007]. Our aim is merely to prove that the pre-order $\sqsubseteq_{\text{SVR}}$ is not not comparable with the should pre-order. Let $\sqsubseteq_{\text{shd}}$ be defined as in [Rensink and Vogler, 2007, Section 3.2].

Example 5.1.16. [Should pre-order]
In this example we prove the following inequalities,

$$\sqsubseteq_{\text{SVR}} \not\subseteq \sqsubseteq_{\mathsf{shd}}, \qquad \sqsubseteq_{\mathsf{shd}} \not\subseteq \sqsubseteq_{\text{SVR}}$$

We show that $\sqsubseteq_{\text{SVR}}$ is not contained in the $\sqsubseteq_{\mathsf{shd}}$. Let $A \stackrel{\text{def}}{=} (\alpha.A \oplus \beta.A)$ and $B \stackrel{\text{def}}{=} \alpha.B$. It is relatively easy to prove that $A \sqsubseteq_{\text{SVR}} B$; this being true because all the clients that comply with A have to be ready to interact on α. To prove that $A \not\sqsubseteq_{\mathsf{shd}} B$ we use the client $C \stackrel{\text{def}}{=} \overline{\alpha}.C + \overline{\beta}.1$. We can prove that A shd C, whereas $B \not\mathsf{shd}\ C$, because B never allows C to reach a successful state.

Now we explain why $\sqsubseteq_{\mathsf{shd}}$ is not contained in $\sqsubseteq_{\text{SVR}}$. The proof that $0 \sqsubseteq_{\mathsf{shd}} \tau^\infty$ relies on two facts: if 0 shd r then if $r \Longrightarrow r'$ implies $r' \stackrel{\checkmark}{\Longrightarrow}$; and since τ^∞ offers no interaction, the states in the computations of $r \mid\mid \tau^\infty$ contain a derivative r' of r such that $r \Longrightarrow r'$. However, $0 \not\sqsubseteq_{\text{SVR}} \tau^\infty$ because $1 \dashv 0$ while $1 \not\dashv \tau^\infty$.

Essentially, the proof that $\sqsubseteq_{\text{SVR}} \not\subseteq \sqsubseteq_{\mathsf{shd}}$ follows from the fact that the compliance allows ever-lasting computation with not client-successful state; whereas the proof of $\sqsubseteq_{\mathsf{shd}} \not\subseteq \sqsubseteq_{\text{SVR}}$ follows from condition (a) of Definition 3.2.1, that is how the compliance deal with the divergence of servers.

5.1.1 Server pre-orders on restricted LTSs

The MUST server pre-order and the compliance server pre-order are not comparable (see 5.3). This result is true in the LTS of processes $\langle \mathsf{CCS}_{\text{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$, which contains

- infinite branching terms
- infinite states
- divergent terms

By dropping some of the properties listed above, we can restrict the LTS, thereby focusing on different LTSs. We study how the relations $\sqsubseteq\!\!\!\sim_{\text{SVR}}$ and $\sqsubseteq_{\text{SVR}}$ are related in some sub-LTSs of the general LTS of processes,

$$\langle \mathsf{CCS}_{\text{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$$

First we compare point (1a) of Definition 5.1.7 with point (1a) of Definition 4.1.17. What we saw in (Example 5.1.2) shows that point (1a) of Definition 4.1.17 does not imply point (1a) of Definition 5.1.7. The converse is true.

Lemma 5.1.17. For every $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$, if for every $s \in Act^\star$, $p_1 \downarrow\!\downarrow s$ implies $p_2 \downarrow\!\downarrow s$, then for every $s \in Act^\star$, $p_1 \Downarrow s$ implies $p_2 \Downarrow s$.

Proof. Fix two processes $p_1, p_2 \in \mathsf{CCS}_{\text{w}\tau}$ such that for every $s \in Act^\star$, if $p_1 \downarrow\!\downarrow s$ then $p_2 \downarrow\!\downarrow s$. Fix a string $s \in Act^\star$ and two processes such that $p_1 \Downarrow s$; we have to prove that $p_2 \Downarrow s$. To this end, it is enough to show that for every s' prefix of s, if $p_2 \stackrel{s'}{\Longrightarrow} p_2'$ then $p_2' \Downarrow$.

Fix a s' prefix of s. The hypothesis $p_1 \Downarrow s$ implies if $p_1 \stackrel{s'}{\Longrightarrow} p_1'$ then $p_1' \Downarrow$. In turn this ensures that $p_1 \downarrow\!\downarrow s'$. The hypothesis now imply that $p_2 \downarrow\!\downarrow s'$, so if $p_2 \stackrel{s'}{\Longrightarrow} p_2'$ then $p_2' \Downarrow$.

The only assumption on s' is that it is a prefix of s, so we have proven that for every s' prefix of s, if $p_1 \Downarrow s$ then $p_2 \Downarrow s$. As there is no assumption on s, the result is true for every $s \in Act^\star$. □

Lemma 5.1.17 implies that if $\preceq_{\text{SVR}}$ is not contained in $\precsim_{\text{SVR}}$, it is because of point (2) in the definition of $\precsim_{\text{SVR}}$.

Let us decorate the symbols $\mathsf{CCS}_{\text{w}\tau}$, $\sqsubseteq_{\text{SVR}}$ and $\sqsubseteq\!\!\!\sim_{\text{SVR}}$ in order to specify the LTS wherein we define the pre-orders. It is relatively easy to prove that in certain sub-LTS of $\langle \mathsf{CCS}_{\text{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$, the alternative characterisations $\precsim_{\text{SVR}}$ and $\preceq_{\text{SVR}}$ remains sound and complete. For instance, we let $\mathsf{CCS}^{\mathsf{fb}}_{\text{w}\tau}$

denote the finite branching processes, and $\sqsubseteq^{\mathsf{fb}}_{\text{SVR}}, \sqsubseteq^{\mathsf{fb}}_{\text{SVR}}$ the server pre-orders defined in the obvious way on the LTS $\langle\, \mathsf{CCS}^{\mathsf{fb}}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$. One can prove the following properties of $\precsim_{\text{SVR}}$.

- if $p_1 \sqsubseteq^{\mathsf{fb}}_{\text{SVR}} p_2$ then $p_1 \precsim_{\text{SVR}} p_2$
- if $p_1, p_2 \in \mathsf{CCS}^{\mathsf{fb}}_{\mathrm{w}\tau}$ and $p_1 \precsim_{\text{SVR}} p_2$, then $p_1 \sqsubseteq^{\mathsf{fb}}_{\text{SVR}} p_2$

Let $\langle\, \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$ be the LTS in which each state is convergent.

Proposition 5.1.18. For every $p_1, p_2 \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau}$, if $p_1 \sqsubseteq^{\Downarrow}_{\text{SVR}} p_2$ then $p_1 \sqsubseteq^{\Downarrow}_{\text{SVR}} p_2$.

Proof. The result follows from the next three set inclusions, which we prove;

a) $\sqsubseteq^{\Downarrow}_{\text{SVR}} \subseteq \precsim_{\text{SVR}}$

b) for every $p_1, p_2 \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau}$, if $p_1 \precsim_{\text{SVR}} p_2$ then $p_1 \preceq_{\text{SVR}} p_2$

c) for every $p_1, p_2 \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau}$, if $p_1 \precsim_{\text{SVR}} p_2$ then $p_1 \sqsubseteq^{\Downarrow}_{\text{SVR}} p_2$

The set inclusion in a) is true because the tests used in (the lemmas that imply) Proposition 4.1.20, are convergent, and so they are in the LTS $\langle\, \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$. This implies that Proposition 4.1.20 is true also in the LTS of convergent states.

We prove b). In a convergent LTS the requirements of $\precsim_{\text{SVR}}$ and $\preceq_{\text{SVR}}$ on the convergence of processes are trivially true, so for every $p_1, p_2 \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau}$, $p_1 \preceq_{\text{SVR}} p_2$ if and only if

- for every $s \in Act^\star$ and $B \in \text{ACC}(p_2, s)$ there exists a set $A \in \text{ACC}(p_2, s)$ such that $A \subseteq B$
- for every $w \in Act^\star$, if $p_2 \stackrel{w}{\Longrightarrow}$ then $p_1 \stackrel{w}{\Longrightarrow}$

The two conditions above follow from $p_1 \precsim_{\text{SVR}} p_2$, and this explains b). The third inequality follows from Theorem 5.1.15. □

We introduce a property of states that relates infinite traces to their finite prefixes,

$$p\,\mathsf{PrefInf} \text{ whenever for every } u \in Act^\infty,\ \text{if } p \stackrel{u_n}{\Longrightarrow} \text{ for every } n \in \mathbb{N},\ \text{then } p \stackrel{u}{\Longrightarrow}$$

Let $\langle\, \mathsf{CCS}^{\mathsf{pinf}},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$ be the LTS in which each state enjoys the property $\mathsf{PrefInf}$.

Proposition 5.1.19. For every $p_1, p_2 \in \mathsf{CCS}^{\mathsf{pinf}}$, if $p_1 \sqsubseteq^{\mathsf{pinf}}_{\text{SVR}} p_2$ then $p_1 \sqsubseteq^{\mathsf{pinf}}_{\text{SVR}} p_2$.

Proof. We process in a manner similar to Proposition 5.1.18, and prove three set inclusions that imply the proposition;

a) $\sqsubseteq^{\mathsf{pinf}}_{\text{SVR}} \subseteq \preceq_{\text{SVR}}$

b) for every $p_1, p_2 \in \mathsf{CCS}^{\mathsf{pinf}}$, if $p_1 \preceq_{\text{SVR}} p_2$ then $p_1 \precsim_{\text{SVR}} p_2$

c) for every $p_1, p_2 \in \mathsf{CCS}^{\mathsf{pinf}}$, if $p_1 \precsim_{\text{SVR}} p_2$ then $p_1 \sqsubseteq^{\mathsf{pinf}}_{\text{SVR}} p_2$

The first inclusion is true because all the clients used to prove the completeness of $\preceq_{\text{SVR}}$ (Proposition 5.1.12) enjoy $\mathsf{PrefInf}$, so they are in the LTS at hand.

The second inclusion follows from the definition of $\preceq_{\text{SVR}}$, Lemma 5.1.17, and the fact that if $p_1, p_2 \in \mathsf{CCS}^{\mathsf{pinf}}$, then they enjoy $\mathsf{PrefInf}$.

The third inequality follows form Theorem 5.1.15. □

Let $\mathsf{CCS}_{\mathsf{web}} = \{\, p \in \mathsf{CCS}_{\mathrm{w}\tau} \mid p\,\mathsf{PrefInf}, \text{for every } s \in Act^\star, p \stackrel{s}{\Longrightarrow} p' \text{ implies } p' \Downarrow \,\}$; we recover the LTS $\langle\, \mathsf{CCS}_{\mathsf{web}},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$ in the usual manner. The LTS of $\mathsf{CCS}_{\mathsf{web}}$ is essentially the LTS of contracts for web-services, because it contains the LTS of [Padovani, 2010], and the one of [Castagna et al., 2009].

Theorem 5.1.20. For every $p_1, p_2 \in \mathsf{CCS}_{\mathsf{web}}$, $p_1 \sqsubseteq^{\mathsf{web}}_{\sim\mathrm{SVR}} p_2$ if and only if $\sqsubseteq^{\mathsf{web}}_{=\mathrm{SVR}}$.

Proof. We have to prove two set inclusions,

a) $\sqsubseteq^{\mathsf{web}}_{\sim\mathrm{SVR}} \subseteq \sqsubseteq^{\mathsf{web}}_{=\mathrm{SVR}}$

b) $\sqsubseteq^{\mathsf{web}}_{=\mathrm{SVR}} \subseteq \sqsubseteq^{\mathsf{web}}_{\sim\mathrm{SVR}}$

The proof of the first inclusion is analogous to the proof of Proposition 5.1.18, and we do not discuss it.

In principle the proof of the second inclusion is similar to the one of Proposition 5.1.19, but there are more complications. We wish to show the three inclusions that follow,

i) $\sqsubseteq^{\mathsf{web}}_{=\mathrm{SVR}} \subseteq \preceq_{\mathrm{SVR}}$

ii) for every $p_1, p_2 \in \mathsf{CCS}_{\mathsf{web}}$, if $p_1 \preceq_{\mathrm{SVR}} p_2$ then $p_1 \precsim_{\mathrm{SVR}} p_2$

iii) for every $p_1, p_2 \in \mathsf{CCS}_{\mathsf{web}}$, if $p_1 \precsim_{\mathrm{SVR}} p_2$ then $p_1 \sqsubseteq^{\mathsf{web}}_{\sim\mathrm{SVR}} p_2$

While the inclusions ii) and iii) can be proven as in Proposition 5.1.19, inclusion i) cannot; it does **not** follow from the proof of completeness of $\preceq_{\mathrm{SVR}}$ (Proposition 5.1.12), because the clients used in Lemma 5.1.9, Lemma 5.1.11, and Lemma 5.1.8 are not in $\mathsf{CCS}_{\mathsf{web}}$, because those clients diverge. These lemmas, though, are true also in the LTS of $\mathsf{CCS}_{\mathsf{web}}$; their proofs are similar to the ones we have given, but rely on different clients obtained by replacing τ^∞ with $\mathsf{1}$.

The client to prove the analogous of Lemma 5.1.8 is

$$C_i \stackrel{\mathtt{def}}{=} \begin{cases} \mathsf{1} + \overline{\alpha}_i.C_{i+1} & \text{if } i < n \\ \mathsf{0} & \text{if } i = n \end{cases}$$

To prove the analogous of Lemma 5.1.9, in Eq. (5.4), we use $\mathsf{1} + \overline{\alpha}.r$ in place of $\tau^\infty + \overline{\alpha}.r$. The client to prove the analogous of Lemma 5.1.11 is as follows,

$$C_i \stackrel{\mathtt{def}}{=} \begin{cases} \mathsf{1} + \overline{\beta}_{i+1}.C_{i+1} & \text{if } i < n, \\ \sum_{i\in I} \overline{\alpha}_i.\,\mathsf{1} & \text{if } i = n \end{cases}$$

By using the clients above, we one can show that $\sqsubseteq^{\mathsf{web}}_{=\mathrm{SVR}} \subseteq \preceq_{\mathrm{SVR}}$. The inclusions b) and c) are proven as in Proposition 5.1.19.

□

Example 5.1.2, Example 5.1.3, and Theorem 5.1.20 show the conditions necessary and sufficient for the server pre-orders to coincide; they are used in the definition of $\mathsf{CCS}_{\mathsf{web}}$. But there is a more natural definition of an LTS, where the server pre-orders coincide. Let $\mathsf{CCS}^{\mathsf{fs}}_{\mathsf{web}} = \{\, p \in \mathsf{CCS}_{\mathrm{w}\tau} \mid p \text{ finite state, for every } s \in Act^\star, p \stackrel{s}{\Longrightarrow} p' \text{ implies } p' \Downarrow \,\}$.

Proposition 5.1.21. For every $p_1, p_2 \in \mathsf{CCS}^{\mathsf{fs}}_{\mathsf{web}}$, $p_1 \sqsubseteq^{\mathsf{fs}}_{\sim\mathrm{SVR}} p_2$ if and only if $\sqsubseteq^{\mathsf{fs}}_{=\mathrm{SVR}}$.

Proof. Note that since every p in $\mathsf{CCS}^{\mathsf{fs}}_{\mathsf{web}}$ is finite state, Königs'lemmaimplies that p enjoys $\mathsf{PrefInf}$: for every $u \in Act^\infty$, if for every $n \in \mathbb{N}$, $p \stackrel{u_n}{\Longrightarrow}$ then $p \stackrel{u}{\Longrightarrow}$; so $\mathsf{CCS}^{\mathsf{fs}}_{\mathsf{web}}$ is contained in $\mathsf{CCS}_{\mathsf{web}}$.

The inclusion $\sqsubseteq^{\mathsf{fs}}_{\sim\mathrm{SVR}} \subseteq \sqsubseteq^{\mathsf{fs}}_{=\mathrm{SVR}}$ follows from Theorem 5.1.20. The proof of the converse inclusion is essentially the same of of b) in Theorem 5.1.20, as the clients used there are in the LTS of $\mathsf{CCS}^{\mathsf{fs}}_{\mathsf{web}}$. □

A similar equality has been proven directly in [Bernardi and Hennessy, Bernardi and Hennessy, 2011], where the LTS is finite state, finite branching, and every term converges, but $\bowtie$ is an unspecified parameter.

In this section we have introduced the compliance server pre-order, we have explained why it differs from the MUST pre-orders, and we have devised an alternative characterisation for it (Theorem 5.1.15). We have also exhibited the two conditions on the LTS that are *necessary* for the resulting server pre-orders to coincide (Example 5.1.2, Example 5.1.3, Theorem 5.1.20).

The next natural problem is to study the client pre-order given by the compliance relation, and compare it with $\sqsubseteq_{\text{CLT}}$.

5.2 Client pre-order

Now we study when a client r_2 is better than a client r_1, if we use the compliance relation to express satisfaction. It turns out that the pre-order we obtain is different from the MUST client pre-order (and the other pre-orders we studied thus far), so we develop yet anther alternative characterisation (Theorem 5.2.25).

Definition 5.2.1. [Client pre-order]
We write $r_1 \sqsubseteq_{\text{CLT}} r_2$ whenever for every process p, $r_1 \dashv p$ implies $r_2 \dashv p$. We refer to the relation $\sqsubseteq_{\text{CLT}}$ as the *compliance client pre-order*. □

Notation In the usual way we reason on $\sqsubseteq_{\text{CLT}}$ up-to associativity and commutativity of $\oplus$ and $+$.

One of the difference between $\sqsubseteq_{\text{CLT}}$ and $\sqsubseteq_{\text{CLT}}$ is the manner in which clients that perform infinite computations are treated. Similarly to what we saw in Section 5.2, also $\sqsubseteq_{\text{CLT}}$ disregards infinite traces. Example 5.1.3 can be adapted to prove this.

Example 5.2.2. Let the process r be defined as p in Figure 4.4, where the p_k's a replaced by r_k's, and r_k denotes a process which performs a sequence of k α actions *followed by 1*. Let q be a the process with only the self loop α.

In this example we prove that $r \sqsubseteq_{\text{CLT}} q$. To this aim we define a suitable co-inductive compliance,

$$\mathcal{R} = \{\, (q', p') \mid r \dashv p, q \stackrel{\alpha^n}{\Longrightarrow} q', p \stackrel{\overline{\alpha}^n}{\Longrightarrow} p', \text{ for some } n \in N \,\}$$

To prove that $\mathcal{R}$ is a co-inductive compliance we have to show that the pairs in it enjoy three properties,

a) $q' \Downarrow$ implies $p' \Downarrow$

b) $q' \,||\, p' \stackrel{\tau}{\not\longrightarrow}$ implies $q' \stackrel{\checkmark}{\longrightarrow}$

c) $q' \,||\, p' \stackrel{\tau}{\longrightarrow} q'' \,||\, p''$

Let us fix a pair $q' \,\mathcal{R}\, p'$. We prove the requires properties. By construction we know that $r \dashv p$, and for some $n \in N$, $q \stackrel{\alpha^n}{\Longrightarrow} q'$ and $p \stackrel{\overline{\alpha}^n}{\Longrightarrow} p'$.

Suppose that $q' \Downarrow$; we have to show that $p' \Downarrow$. Since r performs every finite sequence of α's, the following computation exists,

$$r \,||\, p \Longrightarrow r' \,||\, p'$$

for some r'; by construction r' converges. The fact that $r \dashv p$ implies that $r' \dashv p'$; $r' \Downarrow$ now ensures that $p' \Downarrow$.

We prove the second property of q' and p': if $q' \,||\, p' \stackrel{\tau}{\not\longrightarrow}$ then $q' \stackrel{\checkmark}{\longrightarrow}$. To prove that the implication is true, we show that its premises are false. First note that if $p' \stackrel{\tau}{\longrightarrow}$ then $q' \,||\, p' \stackrel{\tau}{\longrightarrow}$, so let us suppose that $p' \stackrel{\tau}{\not\longrightarrow}$.

We know that $q \stackrel{\alpha^n}{\Longrightarrow} q'$ for some $n \in \mathbb{N}$. The construction of r ensures that $r \stackrel{\alpha^n}{\Longrightarrow} r' \stackrel{\alpha}{\longrightarrow}$ for some r' such that $r' \stackrel{\checkmark}{\not\longrightarrow}$. The computation $r \,||\, p \Longrightarrow r' \,||\, p'$ and $r \dashv p$ imply that $r' \dashv p'$. Since $r' \stackrel{\checkmark}{\not\longrightarrow}$,

the composition $r' \parallel p'$ is not stable; since $p' \stackrel{\tau}{\not\longrightarrow}$, it follows that $p' \stackrel{\overline{\alpha}}{\longrightarrow}$. Now we can prove that $q' \parallel p' \stackrel{\tau}{\longrightarrow}$ because $q \stackrel{\alpha}{\longrightarrow}$.

To prove the third property that q' and p' have to enjoy, suppose that $q' \parallel p' \stackrel{\tau}{\longrightarrow} q'' \parallel p''$. We have to show that $q'' \mathcal{R} p''$; this follows immediately from the construction of $\mathcal{R}$, and the fact that the only visible action performed by q is α. □

In the example above we have shown two processes r and q related by $\sqsubseteq_{\text{CLT}}$, $r \sqsubseteq_{\text{CLT}} q$, and such that $q \stackrel{\alpha^\infty}{\Longrightarrow}$, while $r \stackrel{\alpha^\infty}{\not\Longrightarrow}$.

The other differences between $\sqsubseteq_{\text{CLT}}$ and $\sqsubseteq\!\!\!\!\sim_{\text{CLT}}$ pertain only the client side of our frameworks.

Example 5.2.3. [All traces vs. unsuccesful traces]
In this example we prove that $\sqsubseteq_{\text{CLT}} \not\subseteq \sqsubseteq\!\!\!\!\sim_{\text{CLT}}$. Let $r_1 = \mathsf{1} + \alpha.\mathsf{0}$ and $r_2 = \tau^\infty + \alpha.\mathsf{0}$. We prove that $r_1 \not\sqsubseteq\!\!\!\!\sim_{\text{CLT}} r_2$. Plainly $\mathsf{0}$ MUST r_1, whereas $\mathsf{0} \not\text{MUST}\ r_2$, because r_2 has no successful states.

On the contrary, $r_1 \sqsubseteq_{\text{CLT}} r_2$. Intuitively, this is true because if $r_1 \dashv p$, then p does not perform the action $\overline{\alpha}$. The only interaction that r_2 offers is α, so r_2 and p cannot communicate; moreover, r_2 is never stable, so the following relation is a co-inductive compliance, $\mathcal{R} = \{\, (r_2, p) \mid r_1 \dashv p \,\}$. This proves that $r_1 \sqsubseteq_{\text{CLT}} r_2$. □

The example above shows that the characterisation of $\sqsubseteq_{\text{CLT}}$ should account for all the executions of the traces, and **not** only the unsuccessful ones. The reason why we can define a server that distinguishes the tests r_1 and r_2 (i.e. proves that $r_1 \not\sqsubseteq\!\!\!\!\sim_{\text{CLT}} r_2$) is that $r_2 \stackrel{\alpha}{\Longrightarrow}\not\checkmark$, whereas $r_1 \stackrel{\alpha}{\not\Longrightarrow}\not\checkmark$. Intuitively, this is the case because MUST disregards what a client does after a successful state; on the contrary $\dashv$ checks what a client does also after having reached a successful state (see Example 3.2.14).

The last difference between $\sqsubseteq_{\text{CLT}}$ and $\sqsubseteq\!\!\!\!\sim_{\text{CLT}}$ is how the action $\checkmark$ is compared with other actions. The pre-order $\sqsubseteq\!\!\!\!\sim_{\text{CLT}}$ never compares $\checkmark$, because only unsuccessful traces matters, and they do not contain states that perform $\checkmark$. The pre-order $\sqsubseteq_{\text{CLT}}$, on the contrary, treats $\checkmark$ as the best action.

Example 5.2.4. [Action $\checkmark$ in the ready sets]
In this example we prove that $\sqsubseteq_{\text{CLT}}$ treats $\checkmark$ as the best action. To do so, we prove that the following implication is neither sound nor complete with respect to the pair (r_1, r_2) in $\sqsubseteq_{\text{CLT}}$,

$$\text{for every } s \in Act^\star \text{ if } B \in \text{ACC}(r_2, s) \text{ then there exists some } A \in \text{ACC}(r_1, s) \text{ such that } A \subseteq B.$$

One sees easily that $\mathsf{1} \not\sqsubseteq\!\!\!\!\sim_{\text{CLT}} \mathsf{0}$, for $\mathsf{1} \dashv \mathsf{0}$ whereas $\mathsf{0} \not\dashv \mathsf{0}$; this means that the implication above is not sound; we prove this. Observe that either $\emptyset \in \text{ACC}(\mathsf{0}, s)$ or $\text{ACC}(\mathsf{0}, s) = \emptyset$. If $\emptyset \in \text{ACC}(\mathsf{0}, s)$, then $s = \varepsilon$; since $\emptyset \in \text{ACC}(\mathsf{1}, \varepsilon)$ we have the matching $\emptyset \subseteq \emptyset$.

The implication above is not complete. We show two clients r_1 and r_2 such that $r_1 \sqsubseteq_{\text{CLT}} r_2$, and that (r_1, r_2) does not satisfy the implication above. Let $r_1 = \alpha.\mathsf{1}$ and $r_2 = \mathsf{1}$. We explain why $\alpha.\mathsf{1} \sqsubseteq_{\text{CLT}} \mathsf{1}$. Let $\alpha.\mathsf{1} \dashv p$ for some p; the assumption and Definition 3.2.1 ensure that $p \Downarrow$. It follows that the relation $\mathcal{R} = \{\, (\mathsf{1}, p) \mid p \Downarrow \,\}$ is a co-inductive compliance. This is true because $p \Downarrow$, $\mathsf{1} \stackrel{\checkmark}{\longrightarrow}$, and $\mathsf{1}$ offers no interactions.

Now observe that the pair $(\alpha.\mathsf{1}, \mathsf{1})$ does satisfy the implication above. We explain why; we have to exhibit a $s \in Act^\star$ and a $B \in \text{ACC}(\mathsf{1}, s)$ such that there is no set $A \in \text{ACC}(\alpha.\mathsf{1}, s)$ that is contained in B. Clearly, $\emptyset \in \text{ACC}(\mathsf{1}, \varepsilon)$; since the set $\text{ACC}(\alpha.\mathsf{1}, \varepsilon)$ is the singleton $\{\{\alpha\}\}$, there is indeed no $A \in \text{ACC}(\alpha.\mathsf{1}, \varepsilon)$ such that $A \subseteq \emptyset$. □

Example 5.2.4 shows that the action $\checkmark$ should be in the ready sets of processes, and that it should be treated as the best action.

In view of the previous examples, the pre-orders $\sqsubseteq_{\text{CLT}}$ and $\sqsubseteq\!\!\!\!\sim_{\text{CLT}}$ are not comparable,

$$\sqsubseteq\!\!\!\!\sim_{\text{CLT}} \not\subseteq \sqsubseteq_{\text{CLT}}, \qquad \sqsubseteq_{\text{CLT}} \not\subseteq \sqsubseteq\!\!\!\!\sim_{\text{CLT}} \tag{5.5}$$

We summarise the differences between the client pre-orders,

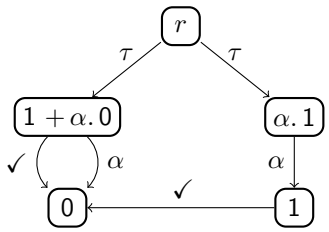


Figure 5.3: LTS of $r = (1 + \alpha.\,0) \oplus \alpha.\,1$

i) the MUST pre-order, $\sqsubseteq_{\text{CLT}}$, compares (unsuccessful) infinite traces of clients. The compliance pre-order disregards infinite traces. This is similar to what the server pre-orders scenario (see point (iii) on 100).

ii) the pre-order $\sqsubseteq_{\text{CLT}}$ compares the behaviour of clients only along unsuccessful executions of a trace. The compliance pre-order checks the behaviour along all the executions of a trace.

iii) the pre-order $\sqsubseteq_{\text{CLT}}$ disregards completely the action ✓. The relation $\sqsubseteq_{\text{CLT}}$ compares ✓ with the other actions and considers it as the best action.

Now our touchstone is Definition 4.2.30, that is the characterisation of $\sqsubseteq_{\text{CLT}}$. We want to adapt that definition, so as to mirror the differences between $\sqsubseteq_{\text{CLT}}$ and $\sqsubseteq_{\text{CLT}}$, and characterise $\sqsubseteq_{\text{CLT}}$. To carry out this task, we have to introduce more notation.

In Lemma 3.2.4 we have established that $\mathsf{0}$ complies with no process at all, and τ^∞ complies with every process. These facts leads to the ideas of usable client and of non-perfect client.

Definition 5.2.5. [Usable clients]
Let

$$\mathcal{U}^{\dashv}_{\text{CLT}} = \{\, r \mid r \dashv p, \text{ for some server } p \,\}$$

If $r \in \mathcal{U}^{\dashv}_{\text{CLT}}$ then we say that r is a *usable* client. □

The set $\mathcal{U}^{\dashv}_{\text{CLT}}$ differs from $\mathcal{U}^{\text{MUST}}_{\text{CLT}}$, and the two notions of usability are not to be confused.

Example 5.2.6. [Usability of clients are not comparable]
In this example we prove that $\mathcal{U}^{\text{MUST}}_{\text{CLT}} \not\subseteq \mathcal{U}^{\dashv}_{\text{CLT}}$ and that $\mathcal{U}^{\dashv}_{\text{CLT}} \not\subseteq \mathcal{U}^{\text{MUST}}_{\text{CLT}}$.

We prove the first inequality. Let $r = (1 + \alpha.\,0) \oplus \alpha.\,1$. We depict the client r in Figure 5.3.

To prove that $r \in \mathcal{U}^{\text{MUST}}_{\text{CLT}}$ we have to show a server p such that p MUST r. It easy routine work to check that all the maximal computations of $r \parallel \overline{\alpha}.\,\mathsf{0}$ are client-successful; so $\overline{\alpha}.\,\mathsf{0}$ MUST r. We have proven that $r \in \mathcal{U}^{\text{MUST}}_{\text{CLT}}$.

Now we want to prove that $r \notin \mathcal{U}^{\dashv}_{\text{CLT}}$, that is $r \not\dashv p$ for every $p \in \mathsf{CCS}_{\text{w}\tau}$. Intuitively, this is the case because $r \xrightarrow{\tau} \alpha.\,\mathsf{1}$, so for a server p to satisfy r (with respect to $\dashv$) it is necessary that p offers (modulo internal moves) the action $\overline{\alpha}$; so $p \stackrel{\overline{\alpha}}{\Longrightarrow} p'$. But this implies the existence of the computation

$$r \parallel p \xrightarrow{\tau} 1 + \alpha.\,\mathsf{0} \parallel p \Longrightarrow \mathsf{0} \parallel p'$$

and plainly $\mathsf{0} \not\dashv p'$, so $r \not\dashv p$.

We have shown that $r \in \mathcal{U}^{\text{MUST}}_{\text{CLT}}$, and $r \notin \mathcal{U}^{\dashv}_{\text{CLT}}$, so $\mathcal{U}^{\text{MUST}}_{\text{CLT}} \not\subseteq \mathcal{U}^{\dashv}_{\text{CLT}}$.

We prove the second inequality, $\mathcal{U}^{\dashv}_{\text{CLT}} \not\subseteq \mathcal{U}^{\text{MUST}}_{\text{CLT}}$. Intuitively, this is the case because $\dashv$ admits livelocks with no client-successful states, whereas MUST requires all the computations to be client-successful. We have already seen in Lemma 3.2.4 that $\tau^\infty \dashv p$ for every process p, so $\tau^\infty \mathcal{U}^{\dashv}_{\text{CLT}}$. On the contrary, $p \not$MUST τ^∞ for every p, so $\tau^\infty \notin \mathcal{U}^{\text{MUST}}_{\text{CLT}}$. □

We expect that if $r_1 \sqsubseteq_{\text{CLT}} r_2$ then the observable behaviours of r_1 and r_2 be related. This is not true if r_2 is "perfect", that it is satisfied by every server (τ^∞ is such a client). We need some notation to focus on the clients that are *not* satisfied by every server.

Definition 5.2.7. [Non-perfect clients]
Let

$$npf = \{\, r \mid r \not\asymp p_{\mathsf{bad}} \text{ for some server } p_{\mathsf{bad}} \,\}$$

If $r \in npf$ then we say that the process r is *non-perfect*. □

We give immediately a result that we will need in Lemma 5.2.24, and, in general, to prove the soundness of the characterisation of $\sqsubseteq_{\text{CLT}}$. Afterwards by means of examples we briefly comment on perfect clients and their characteristic property.

Lemma 5.2.8. [Characteristic property of non-perfect clients]
For every $r \in \mathsf{CCS}_{\mathrm{w}\tau}$, $r \in npf$ if and only if $r \stackrel{s}{\Longrightarrow} r'$ and $r' \Downarrow$, for some $s \in Act^\star$.

Proof. We have to show two implications,

ii) if $r \in npf$ then there exists a $s \in Act^\star$ such that $r \stackrel{s}{\Longrightarrow} r'$ and $r' \Downarrow$;

iiii) if there exists a $s \in Act^\star$ such that $r \stackrel{s}{\Longrightarrow} r'$ and $r' \Downarrow$, then $r \in npf$.

We prove the first implication. Let $r \in npf$. By definition there exists a server p_{bad} such that $r \not\asymp p_{\mathsf{bad}}$. Consider the following relation,

$$\mathcal{R} = \{\, (r,\, p) \mid r, p \in \mathsf{CCS}_{\mathrm{w}\tau},\, s \in Act^\star,\, r \stackrel{s}{\Longrightarrow} r' \text{ implies } r' \not\Downarrow \,\}$$

One can prove that $\mathcal{R}$ is a co-inductive compliance. Intuitively, this is true because if $r\ \mathcal{R}\ p$, then $r \not\Downarrow$ and $r \xrightarrow{\tau}$, so condition (a) and condition (b) of Definition 3.2.1 are trivially true. Condition (c) is also true, by construction of $\mathcal{R}$. Since $r \not\asymp p_{\mathsf{bad}}$, it follows that $r\ \not\mathcal{R}\ p_{\mathsf{bad}}$.

In turn this implies that for some r' and p', $r \parallel p_{\mathsf{bad}} \Longrightarrow r' \parallel p'$ and one of the following is true,

- $r' \Downarrow$ and $p' \not\Downarrow$
- $r' \parallel p' \stackrel{\tau}{\not\longrightarrow}$ and $r' \stackrel{\checkmark}{\not\longrightarrow}$

In both cases, $r' \Downarrow$. Since $r \stackrel{s}{\Longrightarrow} r'$ for some $s \in Act^\star$, we have proven that if $r \in npf$ then for some $s \in Act^\star$ and r', $r \stackrel{s}{\Longrightarrow} r'$ and $r' \Downarrow$.

We prove the second implication. Fix a process r such that for some $s \in Act^\star$ and r', $r \stackrel{s}{\Longrightarrow} r'$ and $r' \Downarrow$. To prove that $r \in npf$ we have to exhibit a server that does not satisfy r with respect to the compliance relation.

Let $s = \alpha_1\alpha_2 \ldots \alpha_n$, and let $p_{\mathsf{bad}} = \overline{\alpha}_1.\overline{\alpha}_2.\ldots.\overline{\alpha_n}.\tau^\infty$. The computation $r \parallel p_{\mathsf{bad}} \Longrightarrow r' \parallel \tau^\infty$, and the fact that $r' \Downarrow$ while $\tau^\infty \not\Downarrow$ let us prove that $r' \not\asymp \tau^\infty$, and so $r \not\asymp p_{\mathsf{bad}}$. □

Lemma 5.2.8 tells us that perfect clients are tightly related to divergence. Consider the next example.

Example 5.2.9. [Perfect clients]
Let $r = (\tau^\infty + \beta.\tau^\infty) \oplus \tau^\infty$; we prove that $\alpha.\,1 \sqsubseteq_{\text{CLT}} r$. This is trivially true, because r is a perfect client. Plainly, there is no trace of r that leads to a state that converges, so Lemma 5.2.8 ensures that r is perfect.

Alternatively, we can prove also that $r \dashv p$ for every p. Consider the following relation,

$$\mathcal{R} = \{\, (r,\, p) \mid p \in \mathsf{CCS}_{\mathrm{w}\tau} \,\} \cup \{\, (\tau^\infty,\, p) \mid p \in \mathsf{CCS}_{\mathrm{w}\tau} \,\}$$

$$\frac{}{r \text{ usbl } \varepsilon}\ r \in \mathcal{U}^{\dashv}_{\text{CLT}};\ [\text{USB-AX}]$$

$$\frac{}{r \text{ usbl } \alpha s}\ r \in \mathcal{U}^{\dashv}_{\text{CLT}}, r \stackrel{\alpha}{\not\Longrightarrow};\ [\text{USB-NOT}]$$

$$\frac{\bigoplus(r \text{ AFTER } \alpha) \text{ usbl } s}{r \text{ usbl } \alpha s}\ r \in \mathcal{U}^{\dashv}_{\text{CLT}}, r \stackrel{\alpha}{\Longrightarrow};\ [\text{USB-ALPHA}]$$

Figure 5.4: Inference rules for the functional $\mathcal{F}_{\mathsf{usbl}}$

The argument to prove that $\mathcal{R}$ is a co-inductive compliance is the same we used in Lemma 5.2.8 for the $\mathcal{R}$ defined there. □

Note that to prove Lemma 5.2.8 we need divergence in the LTS at hand, for we have used τ^∞ to prove the second implication.

The characteristic property of perfect clients under the assumption of convergence is not the one stated in Lemma 5.2.8. We prove this by showing a client r that is perfect in $\langle\, \mathsf{CCS}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$ but not in $\langle\, \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$.

Example 5.2.10. [Divergence and perfect clients]
Let $r_2 = 1 + \beta.1$. In order to distinguish the clients $\alpha.1$ and r_2 divergence is necessary. For instance, let $p = \overline{\alpha}.0 + \overline{\beta}.\tau^\infty$; $\alpha.1 \dashv p$; whereas $r_2 \not\dashv p$, because $r_1 \mid\mid p \Longrightarrow 1 \mid\mid \tau^\infty$, and $1 \Downarrow$, while $\tau^\infty \not\Downarrow$, so $1 \not\dashv \tau^\infty$. This shows that r_2 is not perfect.

In this example we prove that under the assumption of convergence , that is in the LTS

$$\langle\, \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$$

the client r_2 is perfect, and so $\alpha.1 \sqsubseteq_{\text{CLT}} r_2$.

We have to show that $r_2 \dashv p$ for every $p \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau}$. We prove that the following relation is a co-inductive compliance,

$$\mathcal{R} = \{\, (r_2,\, p) \mid p \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau} \,\} \cup \{\, (1,\, p) \mid p \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau} \,\}$$

We give the intuition behind the argument. Let $r\ \mathcal{R}\ p$; since $pp \in \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau}$ point (a) of Definition 3.2.1 is satisfied. Moreover, by construction $r \stackrel{\checkmark}{\longrightarrow}$, also point (b) of Definition 3.2.1 is true. Now let r be a derivative of r_2 and p' a derivative of p. if $r \mid\mid p \stackrel{\tau}{\longrightarrow} r' \mid\mid p'$, then either $r' = 1$, or $r' = r_2$. In both cases $r'\ \mathcal{R}\ p'$ by construction of $\mathcal{R}$. □

We introduce the notation to reason on the usability of clients along all the executions of finite traces.

Definition 5.2.11. [Usability after trace]
Let $\mathcal{F}_{\mathsf{usbl}} : \mathcal{P}(\mathsf{CCS}_{\mathrm{w}\tau} \times Act^\star) \longrightarrow \mathcal{P}(\mathsf{CCS}_{\mathrm{w}\tau} \times Act^\star)$ be the rule functional given by the inference rules in Figure 5.4. Lemma C.0.25 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_{\mathsf{usbl}}(X)$; we call this solution the *usable after*, and we denote it usbl : That is $\mathsf{usbl} = \mu X.\mathcal{F}_{\mathsf{usbl}}(X)$. □

The predicate usbl tells us nothing about the convergence of the clients, for instance τ^∞ usbl s for every string $s \in Act^\star$. This explains why we do not use the symbol $\Downarrow$.

The predicate usbl allows us to prove a property of traces analogous to what we proved in Lemma 4.2.15.

Lemma 5.2.12. For every $s \in Act^\star$ and $r \in \mathsf{CCS}_{w\tau}$, if r usbl s and $r \stackrel{s'}{\Longrightarrow}$ for some s' prefix of s, then $\bigoplus(r \text{ AFTER } s') \in \mathcal{U}_{\text{CLT}}^{\dashv}$.

Proof. The argument is similar to the proof of Lemma 4.2.15. □

By using usbl, we amend the definition of usable actions; we also take into account what we discussed in Example 5.2.4.

Definition 5.2.13. [Usable actions and ✓ after trace]
Let

$$\mathsf{ua}^{\checkmark}(r,s) = \{\, \alpha \in Act \mid r \stackrel{s\alpha}{\Longrightarrow} \text{ and } r \ \mathsf{usbl}\ s\alpha \,\} \cup \{\, \checkmark \mid r \stackrel{s}{\Longrightarrow} \stackrel{\checkmark}{\longrightarrow} \,\}$$

We refer to the set $\mathsf{ua}^{\checkmark}(r,s)$ as the *usable actions of r after s.* □

Again, the difference with Definition 4.2.16 is that the new predicate accounts for all the executions of s, and not only the unsuccessful ones. Moreover, now we account for the action ✓; the role of this action in the set $\mathsf{ua}^{\checkmark}$ will become evident later on.

Definition 5.2.14. [Acceptance sets with ✓]
For every process r and $s \in Act^\star$, let

$$\text{ACC}^{\checkmark}(r,s) = \{\, S^{\checkmark}(r') \mid r \stackrel{s}{\Longrightarrow} r' \stackrel{\tau}{\not\longrightarrow} \,\}$$

where $S^{\checkmark}(r) = \{\, \alpha \in Act \mid r \stackrel{\alpha}{\longrightarrow} \,\} \cup \{\, \checkmark \mid r \stackrel{\checkmark}{\longrightarrow} \,\}$. We call $\text{ACC}^{\checkmark}(r,s)$ the *acceptance set (with ✓) of r after s.* □

To treat ✓ as the best action, we change the way whereby we compare ready sets.
Let $A \sqsubseteq^{\checkmark} B$ if and only

- if $\alpha \in A$ then $\alpha \in B$ or $\checkmark \in B$.
- if $\checkmark \in A$ then $\checkmark \in B$

The relation $\sqsubseteq^{\checkmark}$ provides a sound comparison between ready sets.

Example 5.2.15. All the following inequalities are true by definition of $\sqsubseteq^{\checkmark}$.

$$\begin{array}{lll} \{\alpha,\beta\} & \sqsubseteq^{\checkmark} & \{\checkmark\} \\ \{\checkmark\} & \not\sqsubseteq^{\checkmark} & \{\alpha\} \\ \{\alpha,\delta\} & \sqsubseteq^{\checkmark} & \{\checkmark,\delta\} \\ \{\checkmark\} & \not\sqsubseteq^{\checkmark} & \emptyset \end{array}$$

The first three inequalities show that ✓ is better than any observable action. □

We have all the notation we need to spell out the behavioural characterisation of $\sqsubseteq_{\text{CLT}}$.

Definition 5.2.16. [Semantic compliance client pre-order]
Let $r_1 \preceq_{\text{CLT}} r_2$ whenever for every $s \in Act^\star$, if r_1 usbl s then for every $B \in \text{ACC}^{\checkmark}(r_2,s)$, there exist a set $A \in \text{ACC}^{\checkmark}(r_1,s)$, such that $A \cap \mathsf{ua}^{\checkmark}(r_1,s) \sqsubseteq^{\checkmark} B$. □

We prove first the completeness of $\preceq_{\text{CLT}}$ (Proposition 5.2.22), and then the soundness (Theorem 5.2.25).

Lemma 5.2.17. For every $r_1, r_2 \in \mathsf{CCS}_{w\tau}$, if $r_1 \sqsubseteq_{\text{CLT}} r_2$, $r_1 \mathcal{U}_{\text{CLT}}^{\dashv}$ and $r_2 \stackrel{\alpha}{\Longrightarrow} r_2'$, $r_2' \in npf$, then $r_1 \stackrel{\alpha}{\Longrightarrow} r_1'$, $r_1' \in npf$.

Proof. The hypothesis that $r'_2 \in npf$ ensures that there exists a p_{bad} such that $r'_2 \not\dashv p_{\mathsf{bad}}$. The hypothesis that $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$ ensures that there exists a p such that $r_1 \dashv p$. Let $\hat{p} = p + \overline{\alpha}.p_{\mathsf{bad}}$.

We prove that $r_2 \not\dashv \hat{p}$. This follows from the existence of the computation $r_2 \,||\, \hat{p} \Longrightarrow r'_2 \,||\, p_{\mathsf{bad}}$, the assumption $r'_2 \not\dashv p$ and Definition 3.2.1.

The hypothesis imply that $r_1 \not\dashv \hat{p}$. Consider the following relation,

$$\mathcal{R} = \{\, (r,\, p + \overline{\alpha}.q) \mid r, p, q \in \mathsf{CCS}_{\mathrm{w}\tau},\, r \dashv p,\, r \overset{\alpha}{\not\Longrightarrow} \,\} \cup \dashv$$

The construction of $\mathcal{R}$ ensures that it is a co-inductive compliance; intuitively, if $r\ \mathcal{R}\ p + \overline{\alpha}.q$, then either $r \dashv p + \overline{\alpha}.q$, or $r \dashv p$ and no interaction on α can happen. In both cases we can prove that all the requirements of Definition 3.2.1 are satisfied.

Since $r \not\dashv \hat{p}$, it follows that $r\ \not\mathcal{R}\ \hat{p}$; the assumption $r \dashv p$ and the definition of $\mathcal{R}$ ensure that $r \overset{\alpha}{\Longrightarrow}$ must be true. □

The previous lemma is not true for traces which are not dangerous.

Example 5.2.18. [Non perfect clients are necessary]
Let us drop, in Lemma 5.2.17, the hypothesis that $r'_2 \in npf$. The statement that we obtain is false.

It is relatively easy to prove that $\beta.1 \sqsubseteq_{\text{CLT}} r$, where $r = \beta.1 + \alpha.\tau^\infty$. To see why the new version of the lemma is false, note that $\beta.1\ \mathsf{usbl}\ \alpha$, $r \overset{\alpha}{\Longrightarrow}$, and $\beta.1 \overset{\alpha}{\not\Longrightarrow}$. □

Lemma 5.2.19. For every $r_1, r_2 \in \mathsf{CCS}_{\mathrm{w}\tau}$, if $r_1 \sqsubseteq_{\text{CLT}} r_2$, $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$, $r_1 \overset{\alpha}{\Longrightarrow}$ and $r_2 \overset{\alpha}{\Longrightarrow}$, then $\bigoplus(r_1 \text{ AFTER } \alpha) \sqsubseteq_{\text{CLT}} \bigoplus(r_1 \text{ AFTER } \alpha)$.

Proof. Let $\hat{r}_2 = \bigoplus(r_2 \text{ AFTER } \alpha)$ and $\hat{r}_1 = \bigoplus(r_1 \text{ AFTER } \alpha)$. We have to prove that $\hat{r}_1 \sqsubseteq_{\text{CLT}} \hat{r}_2$. Fix a process p' such that $\hat{r}_1 \dashv p'$, we are required to prove that $\hat{r}_2 \dashv p'$.

By hypothesis $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$, so there exists a process p such that $r_1 \dashv p$. Let $\hat{p} = p + \overline{\alpha}.p'$. Thanks to the assumptions on p' and p, one can show that $r_1 \dashv \hat{p}$; the witness of this is the co-inductive compliance

$$\mathcal{R} = \{\, (r',\, p + \overline{\alpha}.q) \mid r_1 \Longrightarrow r', r_1 \dashv p, \hat{r}_1 \dashv q \,\} \cup \dashv$$

The hypothesis $r_1 \sqsubseteq_{\text{CLT}} r_2$ ensures that $r_2 \dashv \hat{p}$. For every $r' \in (r_2 \text{ AFTER } \alpha)$. the computation $r_2 \,||\, \hat{p} \Longrightarrow r' \,||\, p'$ and Corollary 3.2.7 imply that $r' \dashv p'$. In turn this implies that $\hat{r}_2 \dashv p'$.

We have proven that if $\hat{r}_1 \dashv p'$ then $\hat{r}_2 \dashv p'$, so $\hat{r}_1 \sqsubseteq_{\text{CLT}} \hat{r}_2$. □

The compliance client pre-order relates the existence of stable states after a trace.

Lemma 5.2.20. For every $s \in Act^\star$, and every $r_1, r_2 \in \mathsf{CCS}_{\mathrm{w}\tau}$, if $r_1 \sqsubseteq_{\text{CLT}} r_2$, $r_1\ \mathsf{usbl}\ s$ and $r_2 \overset{s}{\Longrightarrow} r'_2 \overset{\tau}{\not\longrightarrow}$ then $r_1 \overset{s}{\Longrightarrow} r'_1 \overset{\tau}{\not\longrightarrow}$.

Proof. Fix a string s and two processes $r_1 \sqsubseteq_{\text{CLT}} r_2$ such that $r_2 \overset{s}{\Longrightarrow} r'_2 \overset{\tau}{\not\longrightarrow}$; we have to to exhibit a r'_1 such that $r_1 \overset{s}{\Longrightarrow} r'_1 \overset{\tau}{\not\longrightarrow}$.

The proof is by induction on s.

Base case ($s = \varepsilon$) In this case $r_2 \overset{\varepsilon}{\Longrightarrow} r'_2 \overset{\tau}{\not\longrightarrow}$, and we have to show that there exists r'_1 such that $r_1 \overset{\varepsilon}{\Longrightarrow} r'_1 \overset{\tau}{\not\longrightarrow}$.

As r'_2 is stable, it follows that $r'_2 \not\dashv \tau^\infty$, because $r'_2 \Downarrow$ and $\tau^\infty \not\Downarrow$. In turn, this ensures that $r_2 \,||\, \tau^\infty \Longrightarrow r'_2 \,||\, \tau^\infty$, imply that $r_2 \not\dashv \tau^\infty$. The hypothesis $r_1 \sqsubseteq_{\text{CLT}} r_2$ ensures that $r_1 \not\dashv \tau^\infty$. The next relation is not a co-inductive compliance

$$\mathcal{R} = \{\, (r',\, \tau^\infty) \mid r_1 \Longrightarrow r' \,\}$$

Since the pairs in $\mathcal{R}$ satisfy point (b) and point (c) of Definition 3.2.1, it follows that $\mathcal{R}$ contains a pair (r', τ^∞) such that $r' \Downarrow$. The definition of $\mathcal{R}$ implies that $r_1 \stackrel{\varepsilon}{\Longrightarrow} r'$. The definition of $\Downarrow$ ensures that $r' \Longrightarrow r'' \stackrel{\tau}{\not\longrightarrow}$ for some r'', and so $r_1 \stackrel{\varepsilon}{\Longrightarrow} r'' \stackrel{\tau}{\not\longrightarrow}$.

Inductive case ($s = \alpha s'$) In this case we have to prove that $r_2 \stackrel{\alpha s'}{\Longrightarrow} r_2' \Downarrow$. The hypothesis ensure that r_1 usbl $\alpha s'$.

Since s' is shorter than s, the inductive hypothesis states that

$$\text{for every } \hat{r}_1, \hat{r}_2 \in \mathsf{CCS}_{\mathrm{w}\tau}, \text{ if } \hat{r}_1 \sqsubseteq_{\mathrm{CLT}} \hat{r}_2,\ \hat{r}_1' \ \mathsf{usbl}\ s \text{ and } \hat{r}_2 \stackrel{s'}{\Longrightarrow} \hat{r}_2' \stackrel{\tau}{\not\longrightarrow} \text{ then } \hat{r}_1 \stackrel{s'}{\Longrightarrow} \hat{r}_1' \stackrel{\tau}{\not\longrightarrow}.$$

Let $\hat{r}_2 = \bigoplus(r_2 \text{ AFTER } \alpha)$. Definition 4.1.5 implies that $\hat{r}_2 \stackrel{s'}{\Longrightarrow} r_2'$. Since $r_2' \Downarrow$, $r_2' \in \mathit{npf}$, and so we can prove that $r_2 \stackrel{\alpha}{\Longrightarrow}$. As r_1 usbl $\alpha s'$ implies $r_1 \in \mathcal{U}_{\mathrm{CLT}}^{\dashv}$, we use Lemma 5.2.17 to prove that $r_1 \stackrel{\alpha}{\Longrightarrow}$; so the set $(r_1 \text{ AFTER } \alpha)$ is non-empty. Let $\hat{r}_1 = \bigoplus(r_1 \text{ AFTER } \alpha)$. Since $r_1 \stackrel{\alpha}{\Longrightarrow}$, the hypothesis r_1 usbl $\alpha s'$ must have been derived by using rule [USB-ALPHA],

$$\frac{\begin{array}{c}\vdots\\ \bigoplus(r_1 \text{ AFTER } \alpha)\ \mathsf{usbl}\ s\end{array}}{r_1\ \mathsf{usbl}\ \alpha s}\ r_1 \in \mathcal{U}_{\mathrm{CLT}}^{\dashv}, r_1 \stackrel{\alpha}{\Longrightarrow};\ [\text{USB-ALPHA}]$$

The premises of rule [USB-ALPHA] above ensure that $\hat{r}_1$ usbl s'; now Lemma 5.2.19 implies that $\hat{r}_1 \sqsubseteq_{\mathrm{CLT}} \hat{r}_2$, and that $\hat{r}_1$ usbl s'.

Since $\hat{r}_2 \stackrel{s'}{\Longrightarrow} r_2' \Downarrow$, we know enough to apply the inductive hypothesis to $\hat{r}_1$ and $\hat{r}_2$; it follows that $\hat{r}_1 \stackrel{s'}{\Longrightarrow} r_1' \Downarrow$ for some r_1'. The definition of $\hat{r}_1$ lets us prove that $r_1 \stackrel{\alpha s'}{\Longrightarrow} r_1' \Downarrow$. □

The next results are analogous to Lemma 4.2.27 and Lemma 4.2.28.

Lemma 5.2.21. For every $s \in \mathit{Act}^\star$, and $r_1, r_2 \in \mathsf{CCS}_{\mathrm{w}\tau}$, if $r_1 \sqsubseteq_{\mathrm{CLT}} r_2$ and r_1 usbl s, then for every $B \in \text{ACC}^{\checkmark}(r_2, s)$, there exist a $A \in \text{ACC}^{\checkmark}(r_1, s)$, such that $A \cap \mathsf{ua}^{\checkmark}(r_1, s) \sqsubseteq^{\checkmark} B$.

Proof. The argument is by induction on the string s.

Base case ($s = \varepsilon$) Fix a ready set $B \in \text{ACC}^{\checkmark}(r_2, \varepsilon)$. Thanks to the hypothesis and Lemma 5.2.20, the set $\text{ACC}^{\checkmark}(r_1, \varepsilon)$ is non-empty, $\text{ACC}^{\checkmark}(r_1, \varepsilon) = \{ A_i \mid i \in I \}$ for some non-empty set I. Now we proceed by contradiction; suppose that

$$\text{for every } i \in I \text{ there exists an action } \hat{\alpha}_i \text{ such that } \hat{\alpha}_i \in A_i \cap \mathsf{ua}^{\checkmark}(r_1, \varepsilon),\ \hat{\alpha}_i \notin B.$$

This essentially means two things,

- $\checkmark \notin B$,
- for every $i \in I$ such that $\hat{\alpha}_i \in \mathit{Act}$, $\hat{\alpha}_i \notin B$

By using this assumption, we show a server p such that (a) $r_2 \not\dashv p$, and (b) $r_1 \dashv p$.

Let for $J \subseteq I$, be the indexes of the actions $\hat{\alpha}_j$ in Act; that is the $\hat{\alpha}_i$ which are **not** $\checkmark$.

The definition of $\mathsf{ua}^{\checkmark}$ ensures that for every $j \in J$ there exists a $\hat{p}_j$ such that

$$\bigoplus(r_1 \text{ AFTER } \hat{\alpha}_j) \dashv \hat{p}_j$$

Let $p = \sum_{j \in J} \overline{\hat{\alpha}}_j.\hat{p}_j$.

We prove point (a): $r_2 \not\dashv p$. Let r_2' be the state such that $S^{\checkmark}(r_2') = B$ and $r_2' \stackrel{\tau}{\not\longrightarrow}$. By construction $p \stackrel{\tau}{\not\longrightarrow}$, so $C_n \parallel r_2''$ stable. This is true because none of the actions offered by p is matched by any actions in B. Since $\checkmark \notin B$, it follows that $r_2' \not\dashv p$. Corollary 3.2.7 and $r_2 \parallel p \Longrightarrow r_2' \parallel p$ imply that $r_2 \not\dashv p$.

We have proven (a), that is $r_2 \not\dashv p$; now we prove (b). We have to show that $r_1 \dashv p$. We are required to exhibit a co-inductive compliance $\mathcal{R}$ such that $r_1 \mathcal{R} \hat{p}$. Let

$$\begin{array}{lcl} \mathcal{R}' & = & \{\, (r', \sum_{j\in J} \overline{\hat{\alpha}_j}.\hat{p}_j) \mid r_1 \Longrightarrow r', \bigoplus(r_1 \text{ AFTER } \hat{\alpha}_j) \dashv \hat{p}_j \,\} \\ \mathcal{R} & = & \mathcal{R}' \cup \dashv \end{array}$$

Plainly, $r_1 \mathcal{R} p$; we prove that $\mathcal{R}$ is a co-inductive compliance. That is, we show that $\mathcal{R} \subseteq \mathcal{F}_{\dashv}(\mathcal{R})$. Fix a pair $r \mathcal{R} p$; Definition 3.2.1 requires us to prove three properties, namely

i) if $r \Downarrow$ then $p \Downarrow$

ii) if $r \parallel p \stackrel{\tau}{\not\longrightarrow}$ then $r \xrightarrow{\checkmark}$

iii) if $r \parallel p \xrightarrow{\tau} r' \parallel p'$ then $r' \mathcal{R} p'$

Since $r \mathcal{R} p$, either $r \dashv p$ or $r \mathcal{R}' p$. In the first case Definition 3.2.1 ensures that the pair (r, p) enjoys the three properties. Suppose that $r \mathcal{R}' p$. It follows that $r_1 \stackrel{s_k}{\Longrightarrow} r$ and $p = \sum_{j\in J} \overline{\hat{\alpha}_j}.\hat{p}_j$.

- We prove that if $r \Downarrow$ then $p \Downarrow$. This is true because by construction $p \Downarrow$.
- We prove that if $r \parallel p \stackrel{\tau}{\not\longrightarrow}$ then $r \xrightarrow{\checkmark}$. As $r \stackrel{\tau}{\not\longrightarrow}$ and $r_1 \Longrightarrow r$, $S^{\checkmark}(r) \in \text{ACC}^{\checkmark}(r_1, \varepsilon)$. The assumption $r \parallel p \stackrel{\tau}{\not\longrightarrow}$ ensures that r does not engage in any visible action $\hat{\alpha}_j$; the assumption on the ready sets A_i's, and the actions $\hat{\alpha}_j$'s ensure that $\checkmark \in S^{\checkmark}(r)$. It follows that $r \xrightarrow{\checkmark}$
- We have to prove that if $r \parallel p \xrightarrow{\tau} r' \parallel p'$, then $r' \mathcal{R} p'$. The argument depends on the rule used to infer the reduction. Note that p is stable, so we have only two cases to discuss.
  - If rule [P-RIGHT] was used, then $r \xrightarrow{\tau} r'$, and $p' = p$. Since $r_1 \Longrightarrow r \xrightarrow{\tau} r'$, $r_1 \Longrightarrow r'$. It follows that $r' \mathcal{R}' p'$, and so $r' \mathcal{R} p'$.
  - If rule [P-SYNCH] was applied, then there exists the derivation

$$\frac{r \xrightarrow{\delta} r' \quad \dfrac{\vdots}{p \xrightarrow{\overline{\delta}} p'}}{r \parallel p \xrightarrow{\tau} r' \parallel p'} \text{ [P-SYNCH]}$$

    The construction of p ensures that $\delta = \hat{\alpha}_j$ for some $j \in J$, and $p' = \hat{p}_j$. Moreover, $r' \in (r_1 \text{ AFTER } \hat{\alpha}_j)$. Since by construction $\bigoplus(r_1 \text{ AFTER } \hat{\alpha}_j)\hat{p}_j$, one can prove that $r' \dashv \hat{p}_j$, thus $r' \mathcal{R} p'$.

We have proven that the relation $\mathcal{R}$ is a co-inductive compliance, so (b) is proven: $r_1 \dashv C_0$.

Inductive case ($s = \alpha s'$) In this case we want to prove that if $B \in \text{ACC}(r_2, \alpha s')$, then $A \in \text{ACC}(r_1, \alpha s')$ such that $A \cap \mathsf{ua}^{\checkmark}(r_1, \alpha s') \sqsubseteq^{\checkmark} B$. As s' is shorter than s, the inductive hypothesis ensures the following implication,

for every $\hat{r}_1, \hat{r}_2 \in \mathsf{CCS}_{\mathrm{w}\tau}$, if $\hat{r}_1 \sqsubseteq_{\text{CLT}} \hat{r}_2$, $\hat{r}_1$ usbl s' and then for every $B' \in \text{ACC}^{\checkmark}(r_2, s')$, there exist a $A' \in \text{ACC}^{\checkmark}(r_1, s')$, such that $A' \cap \mathsf{ua}^{\checkmark}(r_1, s') \sqsubseteq^{\checkmark} B'$.

We prove enough facts as to let us use the inductive hypothesis. Since $B \in \text{ACC}(r_2, \alpha s')$ we know that $r_2 \stackrel{\alpha}{\Longrightarrow}$; let $\hat{r}_2 = \bigoplus(r_2 \text{ AFTER } \alpha)$. Note that $B \in \text{ACC}(r_2, \alpha s')$ ensures that there exists a r_2' such that $r_2 \stackrel{\alpha}{\Longrightarrow} r_2' wts' r' \Downarrow$, where r' is the term with ready set B. Lemma 5.2.8 and $r_2' \stackrel{s'}{\Longrightarrow} r' \Downarrow$, imply that $r_2' \in npf$. The hypothesis r_1 usbl $\alpha s'$ implies that $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$. Now $r_2 \stackrel{\alpha}{\Longrightarrow} r_2'$ and $r_2' \in npf$ and Lemma 5.2.17 implies that $r_1 \stackrel{\alpha}{\Longrightarrow}$. Let $\hat{r}_1 = \bigoplus(r_1 \text{ AFTER } \alpha)$. Lemma 5.2.19 implies that $\hat{r}_1 \sqsubseteq_{\text{CLT}} \hat{r}_2$.

Since $r_1 \stackrel{\alpha}{\Longrightarrow}$, the hypothesis r_1 usbl $\alpha s'$ ensures that $\hat{r}_1$ usbl s'. We have proven all we need to use the inductive hypothesis.

Since $B \in \text{ACC}(r_2, \alpha s')$, $B \in \text{ACC}(\hat{r}_2, s')$. The inductive hypothesis ensures that there exists a $A \in \text{ACC}(\hat{r}_1, s')$ such that $A' \cap \mathsf{ua}^{\checkmark}(\hat{r}_1, s') \sqsubseteq^{\checkmark} B$. Since $\text{ACC}(r_1, \alpha s') = \text{ACC}(\hat{r}_1, s')$ and $\mathsf{ua}^{\checkmark}(r_1, \alpha s') = \mathsf{ua}^{\checkmark}(r_1, s')$, it follows that there exists a set $A \in \text{ACC}(r_1, \alpha s')$ such that $A \cap \mathsf{ua}^{\checkmark}(r_1, \alpha s') \sqsubseteq^{\checkmark} B$. □

Proposition 5.2.22. [Completeness]
If $r_1 \sqsubseteq_{\text{CLT}} r_2$ then $r_1 \preceq_{\text{CLT}} r_2$.

Proof. Follows from Lemma 5.2.21. □

We need two lemmas to prove the converse of Proposition 5.2.22.

Lemma 5.2.23. If $r_1 \preceq_{\text{CLT}} r_2$ and $r_1 \xrightarrow{\tau} r_2'$ then $r_1 \preceq_{\text{CLT}} r_2'$.

Proof. Fix two processes r_1 and r_2 such that $r_1 \preceq_{\text{CLT}} r_2$ and $r_2 \xrightarrow{\tau} r_2'$. We have to show that $r_1 \preceq_{\text{CLT}} r_2'$. Definition 5.2.16 requires us to prove the following properties, for every $s \in Act^\star$ such that r_1 usbl s, for every $B \in \text{ACC}^{\checkmark}(r_2', s)$, there exist a $A \in \text{ACC}^{\checkmark}(r_1, s)$, such that $A \cap \mathsf{ua}^{\checkmark}(r_1, s) \sqsubseteq^{\checkmark} B$.

Fix a string $s \in Act^\star$ such that r_1 usbl s. Suppose that $B \in \text{ACC}^{\checkmark}(r_2', s)$; Definition 5.2.14 and the hypothesis $r_2 \xrightarrow{\tau} r_2'$ imply that $B \in \text{ACC}^{\checkmark}(r_2, s)$. The hypothesis $r_1 \preceq_{\text{CLT}} r_2$ and Definition 5.2.16 imply that there exist a $A \in \text{ACC}^{\checkmark}(r_1, s)$, such that $A \cap \mathsf{ua}^{\checkmark}(r_1, s) \sqsubseteq^{\checkmark} B$. □

Lemma 5.2.24. Let $r_1 \preceq_{\text{CLT}} r_2$. If $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$, $r_2 \xrightarrow{\alpha} r_2'$, and $r_2' \in npf$, then

i) the set $(r_1 \text{ AFTER } \alpha)$ is non-empty

ii) if $(r_1 \text{ AFTER } \alpha) \neq \emptyset$, then $\bigoplus(r_1 \text{ AFTER } \alpha) \preceq_{\text{CLT}} r_2'$

Proof. We prove point (i). Since $r_2' \in npf$, Lemma 5.2.8 ensures that $r_2' \stackrel{s}{\Longrightarrow} r_2'' \Downarrow$ for some r_2'', and so $r_2 \stackrel{\alpha s}{\Longrightarrow} r_2''$. Either $r_1 \not$usbl αs or r_1 usbl αs.

- if $r_1 \not$usbl αs then rule [USB-NOT] (see Figure 5.4) cannot be used, for otherwise r_1 usbl α. Since $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$, it follows that the second side condition of [USB-NOT] must be false: $r_1 \stackrel{\alpha}{\Longrightarrow}$.
- if r_1 usbl αs then $r_2 \stackrel{\alpha s}{\Longrightarrow} r_2''$ and the hypothesis $r_1 \preceq_{\text{CLT}} r_2$ implies that $r_1 \stackrel{\alpha s}{\Longrightarrow} r_1'$; so $r_1 \stackrel{\alpha}{\Longrightarrow}$

We prove point (ii). Suppose that the set $(r_1 \text{ AFTER } \alpha)$ be non-empty, and let $\hat{r} = \bigoplus(r_1 \text{ AFTER } \alpha)$. We have to explain why $\hat{r} \preceq_{\text{CLT}} r_2'$. Definition 5.2.16 requires us to prove the following condition, for every $B \in \text{ACC}^{\checkmark}(r_2', s)$, there exist a $A \in \text{ACC}^{\checkmark}(\hat{r}, s)$, such that $A \cap \mathsf{ua}^{\checkmark}(r_1, s) \sqsubseteq^{\checkmark} B$

First note that the hypothesis $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$ ensures the implication

$$\text{for every } s \in Act^\star, \text{ if } \hat{r} \text{ usbl } s \text{ then } r_1 \text{ usbl } \alpha s \tag{5.6}$$

This is true because if $\hat{r}$ usbl s then there is the following derivation

$$\frac{\hat{r} \text{ usbl } s}{r_1 \text{ usbl } \alpha s}\ r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}, r_1 \stackrel{\alpha}{\Longrightarrow};\ [\text{USB-ALPHA}]$$

Fix a string $s \in Act^\star$ such that $\hat{r}$ usbl s and $B \in \text{ACC}^{\checkmark}(r_2', s)$ for some set B.

Since r_1 usbl αs, Definition 5.2.16, and the hypothesis $r_1 \preceq_{\text{CLT}} r_2$ imply that there exist a $A \in \text{ACC}^{\checkmark}(r_1, \alpha s)$, such that $A \cap \mathsf{ua}^{\checkmark}(r_1, \alpha s) \sqsubseteq^{\checkmark} B$. Thanks to the equality $\mathsf{ua}^{\checkmark}(r_1, \alpha s) = \mathsf{ua}^{\checkmark}(\hat{r}, s)$ the previous inclusion becomes $A \cap \mathsf{ua}^{\checkmark}(\hat{r}, s) \sqsubseteq^{\checkmark} B$. The equality $\text{ACC}^{\checkmark}(r_1, \alpha s) = \text{ACC}^{\checkmark}(\hat{r}, s)$ implies that $A \in \text{ACC}^{\checkmark}(r_1, s)$.

□

Theorem 5.2.25. [Alternative characterisation $\sqsubseteq_{\text{CLT}}$]
For every $r_1, r_2 \in \mathsf{CCS}_{\text{w}\tau}$, $r_1 \sqsubseteq_{\text{CLT}} r_2$ if and only if $r_1 \preceq_{\text{CLT}} r_2$.

Proof. We have to show that if $r_1 \preceq_{\text{CLT}} r_2$, then $r_1 \sqsubseteq_{\text{CLT}} r_2$. It suffices to prove that the relation $\mathcal{R}$ defined below is a co-inductive compliance.

$$\mathcal{R} = \{\, (r_2,\, p) \mid r_1 \preceq_{\text{CLT}} r_2,\, r_1 \dashv p \,\}$$

Fix a pair $r\ \mathcal{R}\ p$; we are required to prove that the pair affords three properties, namely

a) if $r \Downarrow$ then $p \Downarrow$

b) if $r \parallel p \overset{\tau}{\not\longrightarrow}$, then $r \xrightarrow{\checkmark}$

c) if $r \parallel p \xrightarrow{\tau} r' \parallel p'$, then $r'\ \mathcal{R}\ p'$

Either $r \notin \mathit{npf}$ or $r \in \mathit{npf}$. In the first case $r \dashv p$, so (a) and point (b) are guaranteed by Definition 3.2.1. We prove (c). Suppose that $r \parallel p \xrightarrow{\tau} r' \parallel p'$. Since r is perfect, so must be r'; it follows that $r' \dashv p'$. Since $r' \preceq_{\text{CLT}} r'$, $r'\ \mathcal{R}\ p'$.

Suppose that $r \in \mathit{npf}$. Then there exists an r_1 and a p such that $r_1 \dashv p$ and $r_1 \preceq_{\text{CLT}} r$. Since $r_1 \dashv p$, r_1 usbl ε; that is

$$r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}} \tag{5.7}$$

We prove (a). Suppose $r \Downarrow$; then $r \overset{\varepsilon}{\Longrightarrow} r' \overset{\tau}{\not\longrightarrow}$. Since $r_1 \in \mathcal{U}^{\dashv}_{\text{CLT}}$ (Eq. (5.7) above), and Definition 5.2.16 ensures that the set $r_1 \overset{\varepsilon}{\Longrightarrow} r'_1 \overset{\tau}{\not\longrightarrow}$. The assumption that $r_1 \dashv p$ and $r_1 \parallel p \Longrightarrow r'_1 \parallel p$, imply that $r'_1 \dashv p$. Condition (a) of Definition 3.2.1 and $r \overset{\tau}{\not\longrightarrow}$ imply that $p \Downarrow$.

We prove point (b). Assume that $r \parallel p \overset{\tau}{\not\longrightarrow}$; our aim is to show that $r \xrightarrow{\checkmark}$.

The composition $r \parallel p$ is stable, so $r \overset{\tau}{\not\longrightarrow}$. It follows that $S^{\checkmark}(r) \in \text{ACC}^{\checkmark}(r, \varepsilon)$. As r_1 usbl ε, so Lemma 5.2.21 implies that there exists a set $A \in \text{ACC}^{\checkmark}(r_1, \varepsilon)$ such that

$$A \cap \mathsf{ua}^{\checkmark}(r_1, \varepsilon) \sqsubseteq^{\checkmark} S^{\checkmark}(r)$$

Definition 5.2.14 ensures that there exists a state r'_1 such that $r_1 \overset{\varepsilon}{\Longrightarrow} r'_1 \overset{\tau}{\not\longrightarrow}$, and $S^{\checkmark}(r'_1) = A$. It follows that

$$S^{\checkmark}(r'_1) \cap \mathsf{ua}^{\checkmark}(r_1, \varepsilon) \sqsubseteq^{\checkmark} S^{\checkmark}(r)$$

We prove that $r'_1 \parallel p \overset{\tau}{\not\longrightarrow}$.

It suffices to show that if $p \xrightarrow{\alpha} p'$, then $\overline{\alpha} \notin S^{\checkmark}(r'_1)$.

If $\overline{\alpha} \in S^{\checkmark}(r'_1)$, then let $r' \in (r_1 \text{ AFTER } \alpha)$; plainly $r_1 \parallel p \xrightarrow{\tau} r' \parallel p'$. The assumption $r_1 \dashv p$ implies that $r' \dashv p'$. This can be used to prove that $\bigoplus(r_1 \text{ AFTER } \alpha) \dashv p'$. It follows that if $\overline{\alpha} \in S^{\checkmark}(r'_1)$, then $\overline{\alpha} \in \mathsf{ua}^{\checkmark}(r_1, \varepsilon)$. This implies that $\overline{\alpha} \in S^{\checkmark}(r)$, and so $r \parallel p \xrightarrow{\tau}$. As this contradicts the assumption $r \parallel p \overset{\tau}{\not\longrightarrow}$, it follows that $r'_1 \notin S^{\checkmark}(r'_1)$.

Since p is stable, r'_1 is stable, and $p \xrightarrow{\alpha}$ implies that $r'_1 \overset{\overline{\alpha}}{\not\longrightarrow}$, it follows that $r'_1 \parallel p \overset{\tau}{\not\longrightarrow}$. The assumption $r_1 \dashv p$, and $r_1 \Longrightarrow r'_1$ imply that $r'_1 \dashv p$. Since $r'_1 \parallel p \overset{\tau}{\not\longrightarrow}$, Definition 3.2.1 ensures that $r'_1 \xrightarrow{\checkmark}$, so $\checkmark \in S^{\checkmark}(r'_1)$; in turn this implies that $\checkmark \in S^{\checkmark}(r)$. It follows that $r \xrightarrow{\checkmark}$.

Now we prove (c). Suppose that $r \parallel p \xrightarrow{\tau} r' \parallel p'$; we are required to prove that $r'\ \mathcal{R}\ p'$. The definition of $\mathcal{R}$ requires us to exhibit a $\hat{r}$ such that

$$\hat{r} \preceq_{\text{CLT}} r', \quad \hat{r} \dashv p'$$

The argument is by case analysis on why $r \parallel p \xrightarrow{\tau} r' \parallel p'$.

If r' is perfect, then $r' \preceq_{\text{CLT}} r'$ and $r' \dashv p'$, so the $\hat{r}$ we are after is r itself.

Let us assume $r' \in \mathit{npf}$. If the reduction is due to rule [P-LEFT], then $r \xrightarrow{\tau} r'$, and $p' = p$. The assumption $r_1 \preceq_{\text{CLT}} r$ and Lemma 5.2.23 let us prove that $r_1 \preceq_{\text{CLT}} r'$. As $r_1 \dashv p$, we know enough to state that $r'\ \mathcal{R}\ p'$.

If the reduction is due to rule [P-RIGHT], then $p \xrightarrow{\tau} p'$, and $r = r'$; the candidate $\hat{r}$ is r_1. On the one hand know that $r_1 \preceq_{\text{CLT}} r$, thus $r_1 \preceq_{\text{CLT}} r'$; on the other hand $r_1 \dashv p$, thus point (c) of Definition 3.2.1 implies that $r_1 \dashv p'$. It follows that $r' \mathcal{R} p'$.

If the reduction is due to rule [P-SYNCH], the $r \xrightarrow{\alpha} r'$, $p \xrightarrow{\overline{\alpha}} p'$. Since $r' \in npf$, Eq. (5.7) above, $r_1 \preceq_{\text{CLT}} r$ and Lemma 5.2.24 imply that $\bigoplus(r_1 \text{ AFTER } \alpha) \preceq_{\text{CLT}} r'$.

Our candidate $\hat{r}$ is $\bigoplus(r_1 \text{ AFTER } \alpha)$. We have to prove that $\hat{r} \dashv p'$. Let

$$\begin{array}{rcl} \mathcal{R}' & = & \{\, (r', p') \mid \hat{r} \Longrightarrow r', p \stackrel{\alpha}{\Longrightarrow} p' \,\} \\ \mathcal{R} & = & \mathcal{R}' \cup \dashv \end{array}$$

The argument to prove that the relation $\mathcal{R}$ is a co-inductive compliance is similar to proof of Lemma 3.2.8.

Since $\hat{r} \dashv p'$ and $\hat{r} \preceq_{\text{CLT}} r'$, the definition of $\mathcal{R}$ ensures that $r' \mathcal{R} p'$. □

5.2.1 Comparison with other pre-orders

In this section we show that $\sqsubseteq_{\text{CLT}}$ differs from all the pre-orders that we have studied so far. This is not surprising, as one would not expect clients to be compared as servers or peers; yet it proves that it is necessary to introduce Definition 5.2.16.

In Eq. (5.5) we have shown that the client pre-orders are not comparable. Also the following inequalities are true

$$\begin{array}{rcl} 1 & \sqsubseteq_{\text{SVR}} & 0 \\ \alpha.1 + \beta.0 & \not\sqsubseteq_{\text{SVR}} & \alpha.0 \\ \alpha.1 & \sqsubseteq_{\text{P2P}} & 1 + \alpha.0 \end{array} \qquad \begin{array}{rcl} 1 & \not\sqsubseteq_{\text{CLT}} & 0 \\ \alpha.1 + \beta.0 & \sqsubseteq_{\text{CLT}} & \alpha.0 \\ \alpha.1 & \not\sqsubseteq_{\text{CLT}} & 1 + \alpha.0 \end{array}$$

The inequalities in the first two rows let us prove also that $\sqsubseteq_{\text{CLT}}$ is not comparable with $\sqsubseteq_{\text{SVR}}$. The fact that $\sqsubseteq_{\text{CLT}} \not\subseteq \sqsubseteq_{\text{CLT}}$ and the set inclusion $\sqsubseteq_{\text{P2P}} \subset \sqsubseteq_{\text{CLT}}$ imply that $\sqsubseteq_{\text{CLT}} \not\subseteq \sqsubseteq_{\text{P2P}}$.

In Section 5.1, we have used the relations $\precsim_{\text{SVR}}$ and $\preceq_{\text{SVR}}$ to prove that in sub-LTSs of

$$\langle\, \mathsf{CCS}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \,\rangle$$

the MUST server pre-order and the compliance server pre-order are comparable, or even coincide. This is possible because the characterisations remain sound and complete even if we focus on certain sub-LTSs of the original one. This is not the case for the client characterisations $\precsim_{\text{CLT}}$ and $\preceq_{\text{CLT}}$.

Let us restrict our attention to the LTS of convergent terms.

Proposition 5.2.26. The relation $\precsim_{\text{CLT}}$ is not a complete description of $\sqsubseteq^{\Downarrow}_{\text{CLT}}$: $\sqsubseteq^{\Downarrow}_{\text{CLT}} \not\subseteq \precsim_{\text{CLT}}$.

Proof. In Example 4.2.26 we have shown that $1 \sqsubseteq_{\text{CLT}} 1 \oplus 1$ under the assumption of convergence. Observe now that $1 \not\precsim_{\text{CLT}} 1 \oplus 1$, because $1 \oplus 1 \stackrel{\varepsilon}{\Longrightarrow}_{\not\checkmark}$, whereas $1 \stackrel{\varepsilon}{\not\Longrightarrow}_{\not\checkmark}$. □

A similar result is true for the compliance based pre-order.

Proposition 5.2.27. The relation $\preceq_{\text{CLT}}$ is not a complete description of $\sqsubseteq^{\Downarrow}_{\text{CLT}}$: $\sqsubseteq^{\Downarrow}_{\text{CLT}} \not\subseteq \preceq_{\text{CLT}}$.

Proof. In Example 5.2.10 we have proven that $\alpha.1 \sqsubseteq^{\Downarrow}_{\text{CLT}} 1 + \beta.1$. Note now that $\alpha.1 \not\preceq_{\text{CLT}} 1 + \beta.1$, because $\{\, \checkmark \,\} \in \text{ACC}^{\checkmark}(1 + \beta.1, \beta)$, whereas $\text{ACC}^{\checkmark}(\alpha.1, \beta) = \emptyset$. □

We leave as an open problem the characterisation of the client pre-orders on sub-LTSs of the one we used. We partly address this issue in Section 6.2 and Section 6.5, where we characterise the client pre-orders on the LTS denoted by session contracts. In that context the restricted MUST client pre-order is coarser than than the restricted compliance client pre-order (see Corollary 6.5.15).

5.3 Peer pre-order

Our knowledge of the compliance server pre-order and of the compliance client pre-order lets us study easily the third pre-order given by the relation $\dashv$, the compliance peer pre-order.

Definition 5.3.1. [Compliance peer pre-order]
We write $p \sqsubseteq_{\text{P2P}} q$ if and only if $p \dashv_{\text{P2P}} r$ implies that $q \dashv_{\text{P2P}} r$ for every process r. We refer to the relation $\sqsubseteq_{\text{P2P}}$ as the *compliance peer pre-order*. □

First, we prove that $\sqsubseteq_{\text{P2P}}$ and $\underset{\sim}{\sqsubseteq}_{\text{P2P}}$ are not comparable, so we have to find a behavioural characterisation for $\sqsubseteq_{\text{P2P}}$.

Example 5.3.2. In this example we prove the following inequalities,

$$\sqsubseteq_{\text{P2P}} \not\subseteq \underset{\sim}{\sqsubseteq}_{\text{P2P}}, \qquad \underset{\sim}{\sqsubseteq}_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{P2P}}$$

We prove the left inequality. We exhibit two processes p and q such that $p \sqsubseteq_{\text{P2P}} q$ and $p \not\underset{\sim}{\sqsubseteq}_{\text{P2P}} q$. In Example 5.2.6 we proved that that the term $p = (1 + \alpha.0) \oplus \alpha.1$ is not usable with respect to $\dashv$.[1] Let $q = \alpha.0$. The inequality $p \sqsubseteq_{\text{P2P}} q$ is true because the process p is not a usable peer, that is there exist no r such that $r \dashv_{\text{P2P}} p$, then $r \dashv_{\text{P2P}} q$.

In the MUST setting, the process p is a usable peer, for instance $\overline{\alpha}.\overline{\beta}.1$ MUST^{P2P} p. The process q on the other hand is not a usable peer, because it perform no $\checkmark$ at all. It follows that $r \not\text{MUST}^{\text{P2P}} q$, and so $p \not\underset{\sim}{\sqsubseteq}_{\text{P2P}} q$. We have shown that $\sqsubseteq_{\text{P2P}} \not\subseteq \underset{\sim}{\sqsubseteq}_{\text{P2P}}$.

We prove the right inequality. We see easily that $\alpha.1 \underset{\sim}{\sqsubseteq}_{\text{P2P}} 1 + \alpha.0$. On the contrary, $\alpha.1 \not\sqsubseteq_{\text{P2P}} 1 + \alpha.0$; a peer that shows this is $r = \overline{\alpha}.1$. The proof that $\alpha.1 \dashv_{\text{P2P}} r$ amounts to showing that the following relation is a peer compliance,

$$\{ (\alpha.1,\, \overline{\alpha}.1),\, (1,\, 1) \}$$

On the contrary, $1 + \alpha.0 \,||\, r \xrightarrow{\tau} 0 \,||\, 1$; since $0 \overset{\checkmark}{\not\longrightarrow}$, $1 + \alpha.0 \not\dashv_{\text{P2P}} r$. We have proven that $\underset{\sim}{\sqsubseteq}_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{P2P}}$. □

The scenario that we haven seen in Section 4.3 appears in the setting of compliance as well: it is easy to show that $\sqsubseteq_{\text{CLT}} \cap \sqsubseteq_{\text{SVR}} \subseteq \sqsubseteq_{\text{P2P}}$ (see Proposition 5.3.28), while the converse is false, $\sqsubseteq_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{CLT}} \cap \sqsubseteq_{\text{SVR}}$. This is because $\sqsubseteq_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{SVR}}$, and the core of the difference between $\sqsubseteq_{\text{P2P}}$ and $\sqsubseteq_{\text{SVR}}$ is the non-trivial usability of peers.

Example 5.3.3. The mutual compliance pre-order is not in the server pre-order; that is $\sqsubseteq_{\text{P2P}} \not\subseteq \sqsubseteq_{\text{SVR}}$. We have $\alpha.0 \sqsubseteq_{\text{P2P}} 1$; this is true because the peer $\alpha.0$ is not usable.

On the contrary, $\alpha.0 \not\sqsubseteq_{\text{SVR}} 1$, the distinguishing test being $r = \overline{\alpha}.1$. □

We will see that the pre-order $\sqsubseteq_{\text{P2P}}$ is related to $\sqsubseteq_{\text{SVR}}$ and $\sqsubseteq_{\text{CLT}}$ in the same way that $\underset{\sim}{\sqsubseteq}_{\text{P2P}}$ is related to $\underset{\sim}{\sqsubseteq}_{\text{SVR}}$ and $\underset{\sim}{\sqsubseteq}_{\text{CLT}}$. The proofs of the client-properties of $\sqsubseteq_{\text{P2P}}$, though, are not as easy as the the proof of Proposition 4.3.5.

In Section 4.3 we have used Lemma 4.1.23 to prove that $\underset{\sim}{\sqsubseteq}_{\text{P2P}} \subseteq \underset{\sim}{\sqsubseteq}_{\text{CLT}}$ (Proposition 4.3.5). The proof of Proposition 4.3.5 does not work for $\sqsubseteq_{\text{P2P}}$. The crucial difference between the MUST setting and the compliance setting is that if $r \xrightarrow{\checkmark}$ then p MUST r is trivially true, whereas $r \dashv p$ may be false; for instance $\alpha.0$ MUST $\overline{\alpha}.0 + 1$, whereas $\overline{\alpha}.0 + 1 \not\dashv \alpha.0$.

Another issue, and indeed the more demanding one, is that the usability of peers differs from the usability of clients.

[1] Note that the term p is called r_1 in Example 5.2.6.

Example 5.3.4. [Usability of peers]
In this example we show that $r \in \mathcal{U}^{\dashv}_{\text{CLT}}$ does not imply that there exists a peer p such that $r \dashv_{\text{P2P}} p$.

Let $r = \alpha.\tau^{\infty} + \alpha.\mathsf{1}$. It is easy to prove that $r \in \mathcal{U}^{\dashv}_{\text{CLT}}$, for instance $r \dashv \overline{\alpha}.\mathsf{0}$.

If we treat r as a peer, then to show that is usable we have to exhibit a p such that $r \dashv_{\text{P2P}} p$. Observe point (i) of Definition 3.2.11, and consider a process $\overline{\alpha}.p$. If $p \Downarrow$ then $r \not\dashv_{\text{P2P}} \overline{\alpha}.p$ because $\tau^{\infty} \not\Downarrow$. If $p \not\Downarrow$ then $r \not\dashv_{\text{P2P}} \overline{\alpha}.p$ because $\mathsf{1} \Downarrow$. It follows that for every $p \in \mathsf{CCS}_{\text{w}\tau}$, $r \not\dashv p$. □

Example 5.3.4 motivates the introduction of yet another set of usable processes. In the current setting, it is convenient to discuss few general properties of peers and $\sqsubseteq_{\text{P2P}}$, then prove the client-properties of $\sqsubseteq_{\text{P2P}}$, and afterwards its server-properties. The client-properties and the server-properties of $\sqsubseteq_{\text{P2P}}$ are proven in two different subsections.

Definition 5.3.5. [Usable peers] Let $\mathcal{U}^{\dashv_{\text{P2P}}} = \{\, p \mid p \dashv_{\text{P2P}} q \text{ for some peer } q \,\}$ □

In Example 5.3.4 we have seen a client that is not a usable peer. Indeed, to be usable as a peer is more demanding than to be usable as a client. The additional requirement that is necessary for a peer p to be usable is that after any trace, all the derivatives of p be either convergent or divergent; formally, if $p \stackrel{s}{\Longrightarrow}$ then one of the following is true,

1. for every $p' \in (p \text{ AFTER } s)$, $p' \Downarrow$
2. for every $p' \in (p \text{ AFTER } s)$, $p' \not\Downarrow$

For instance, in Example 5.3.4 r is not usable because $(r \text{ AFTER } \alpha) = \{\, \tau^{\infty}, \mathsf{1} \,\}$, and $\tau^{\infty} \not\Downarrow$, while $\mathsf{1} \Downarrow$.

From now on we use the symbol usbl to denote a predicate defined as in Definition 5.2.11, but using $\mathcal{U}^{\dashv_{\text{P2P}}}$ in place of $\mathcal{U}^{\dashv}_{\text{CLT}}$. This is true also for the set of usable actions ua.

Lemma 5.3.6. For every $s \in Act^{\star}$ and $p \in \mathsf{CCS}_{\text{w}\tau}$, if $p \;\mathsf{usbl}\; s$ and $p \stackrel{s'}{\Longrightarrow}$ for some s' prefix of s, then $\bigoplus(p \text{ AFTER } s') \in \mathcal{U}^{\dashv_{\text{P2P}}}$.

Proof. The argument is similar to the proof of Lemma 4.2.15. □

We need a result on the convergence of peers. Recall the symbol $\downdownarrows$ given in Definition 5.1.4.

Lemma 5.3.7. For every $p, r \in \mathsf{CCS}_{\text{w}\tau}$ such that $p \dashv_{\text{P2P}} r$, if $r \downdownarrows \overline{s}$ and $r \stackrel{\overline{s}}{\Longrightarrow}$, then $p \downdownarrows s$.

Proof. If $p \stackrel{s}{\not\Longrightarrow}$ the argument is straightforward. Suppose that $p \stackrel{s}{\Longrightarrow}$. Pick a $r' \in (r \text{ AFTER } s)$; for every $p' \in (p \text{ AFTER } s)$ we infer the computation

$$p \,||\, r \Longrightarrow p' \,||\, r'$$

from which it follows that $p' \dashv_{\text{P2P}} r'$. The hypothesis that $r \downdownarrows s$ implies that $r' \Downarrow$, and so point (a) of Definition 3.2.11 ensures that $p' \Downarrow$. The only assumption on p' is that $p' \in (p \text{ AFTER } s)$, so the argument implies that $\bigoplus(p \text{ AFTER } s) \Downarrow$. In turn this lets us prove that $p \downdownarrows s$. □

Lemma 5.3.8. For every string $s \in Act^{\star}$, and $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \sqsubseteq_{\text{P2P}} q$, $p \;\mathsf{usbl}\; s$, $q \stackrel{s}{\Longrightarrow}$, then $p \stackrel{s}{\Longrightarrow}$.

Proof. Fix a string $s \in Act^{\star}$ and two processes p and q that satisfy the hypothesis. To prove that $p \stackrel{s}{\Longrightarrow}$ we reason by contradiction. Suppose that $p \stackrel{s}{\not\Longrightarrow}$, and let s' be the longest prefix of s performed by p. Let $s' = \alpha_1\alpha_2 \ldots \alpha_n$, where $n < \mathsf{len}(s)$. We use this assumption to define a peer A that distinguishes, in the sense that

a) $A \dashv_{\text{P2P}} p$

b) $A \not\dashv_{\text{P2P}} q$

The hypothesis that p usbl s and Lemma 5.3.6 imply that for every $0 \leq i \leq n$ there exists a peer r_n such that

$$\bigoplus(p \text{ AFTER } s_n) \dashv_{\text{P2P}} r_n$$

Let

$$A_i \stackrel{\text{def}}{=} \begin{cases} r_i + \overline{\alpha}_{i+1}.A_{i+1} & \text{if } i \leq n \\ \overline{\alpha}_i.\mathsf{0} & \text{if } i = n+1 \end{cases}$$

The peer that distinguishes p and q is A_0.

To see why $A_0 \not\dashv_{\text{P2P}} q$, consider the computation

$$A_0 \mid\mid q \Longrightarrow \mathsf{0} \mid\mid q'$$

where q' is reached by q after the trace $\alpha_1 \ldots \alpha_{n+1}$. If $q' \not\Downarrow$, then $\mathsf{0} \not\dashv_{\text{P2P}} q'$; if $q' \Downarrow$ then the computation can be extended to a stable state $\mathsf{0} \mid\mid q''$; since $\mathsf{0}$ it follows that $\mathsf{0} \not\dashv_{\text{P2P}} q''$. In both cases $A_0 \not\dashv_{\text{P2P}} q$.

The hypothesis $p \sqsubseteq_{\text{P2P}} q$ implies that $A_0 \not\dashv p$. In view of this fact, to prove that $p \stackrel{s}{\Longrightarrow}$ it is enough to show a co-inductive mutual compliance $\mathcal{R}$ such that if $p \stackrel{s}{\not\Longrightarrow}$, then $A_0 \mathcal{R} p$.

Let

$$\mathcal{R} = \{\, (r + \overline{\beta}.q, p) \mid r \dashv_{\text{P2P}} p, \text{ if } p \stackrel{\alpha}{\Longrightarrow} \text{ then } \bigoplus(p \text{ AFTER } \beta) \dashv_{\text{P2P}} q \,\} \cup \dashv_{\text{P2P}}$$

We explain why the elements of $\mathcal{R}$ enjoy the properties required by Definition 3.2.11. Let $A \mathcal{R} p'$, we show one by one the following facts,

i) $A \Downarrow$ if and only $p' \Downarrow$

ii) if $A \mid\mid p' \stackrel{\tau}{\not\longrightarrow}$ then $A \stackrel{\checkmark}{\longrightarrow}$, $p \stackrel{\checkmark}{\longrightarrow}$

iii) if $A \mid\mid p' \stackrel{\tau}{\longrightarrow} A' \mid\mid p''$ then $A' \mathcal{R} p''$

If $A \dashv p'$ then the properties above are true, so suppose that $A \mathcal{R} p'$ because of the auxiliary relation in the definition of $\mathcal{R}$. It follows that $A = r + \overline{\beta}.q$, that $r \dashv p$, and that $p \stackrel{\beta}{\Longrightarrow}$.

Point (ii) is true because $r \dashv_{\text{P2P}} p$ and $A \Downarrow$ if and only if $r \Downarrow$.

To prove point (ii) suppose that $A \mid\mid p' \stackrel{\tau}{\not\longrightarrow}$. It follows that $A \stackrel{\tau}{\not\longrightarrow}$, and that $r \mid\mid p' \stackrel{\tau}{\not\longrightarrow}$, so $r \dashv_{\text{P2P}} p$ implies $r \stackrel{\checkmark}{\longrightarrow}$; this ensures that $A \stackrel{\checkmark}{\longrightarrow}$.

The proof of point (iii) is by case analysis on the rule used to infer the reduction $A \mid\mid p' \stackrel{\tau}{\longrightarrow} A' \mid\mid p''$. If the rule used is [P-Left], then $A \stackrel{\tau}{\longrightarrow} A'$ and $p' = p''$. The internal move of A must have been due to the r in it, so $A' = r' + \overline{\beta}.q$ and $r \stackrel{\tau}{\longrightarrow} r'$. Definition 3.2.11 ensures that $r' \dashv p$, so $A' \mathcal{R} p''$.

If the rules used to infer the reduction was [P-Right], then $A = A'$ and $p' \stackrel{\tau}{\longrightarrow} p''$. Definition 3.2.11 ensures that $r' \dashv p$. If $p'' \stackrel{\beta}{\not\Longrightarrow}$ then $A' \mathcal{R} p''$. If $p'' \stackrel{\beta}{\Longrightarrow}$, then note that $\bigoplus(p' \text{ AFTER } \beta) \dashv_{\text{P2P}} q$ implies that $\bigoplus(p'' \text{ AFTER } \beta) \dashv_{\text{P2P}} q$, and so $A' \mathcal{R} p''$.

If the reduction is due to [P-Synch], then the argument depends on the which one of the summands of A performed the action. If r caused the interaction of A, then $A' = r'$ for some r' such that $r \stackrel{\overline{\delta}}{\longrightarrow} r'$; it follows that $p' \stackrel{\delta}{\longrightarrow} p''$. Since $r \dashv_{\text{P2P}} p$, Definition 3.2.11 implies that $r' \dashv_{\text{P2P}} p''$; in turn this implies that $A' \mathcal{R} p''$.

If $\overline{\beta}.q$ caused the interaction of A, then $A' = q$ and $p' \in (p' \text{ AFTER } \beta)$. As $\bigoplus(p' \text{ AFTER } \beta) \dashv_{\text{P2P}} q$, it follows that $p'' \dashv_{\text{P2P}} q$, and so $A'' \dashv_{\text{P2P}} p''$.

If $p \stackrel{s}{\not\Longrightarrow}$ then one can prove that $A_0 \mathcal{R} p$, but this cannot be true, so $p \stackrel{s}{\Longrightarrow}$. □

Lemma 5.3.9. For every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \sqsubseteq_{\text{P2P}} q$, $p \in \mathcal{U}^{\dashv_{\text{P2P}}}$, and $q \stackrel{\alpha}{\Longrightarrow}$, then $p \stackrel{\alpha}{\Longrightarrow}$.

Proof. The argument is similar to the proof of Lemma 5.2.17, but in this case we use $p_{\mathsf{bad}} = \mathsf{0}$. □

Lemma 5.3.10. For every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \sqsubseteq_{\text{P2P}} q$, $p \in \mathcal{U}^{\dashv_{\text{P2P}}}$, $p \stackrel{\alpha}{\Longrightarrow}$, and $q \stackrel{\alpha}{\Longrightarrow}$, then $\bigoplus(p \text{ AFTER } \alpha) \sqsubseteq_{\text{P2P}} \bigoplus(p \text{ AFTER } \alpha)$.

Proof. The proof of this lemma is similar to the proof of Lemma 5.2.19. □

Client-properties of $\sqsubseteq_{\mathbf{p2p}}$

Now we turn our attention to the client-properties of $\sqsubseteq_{\text{P2P}}$.

Lemma 5.3.11. For every $s \in Act^\star$, and every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \sqsubseteq_{\text{P2P}} q$, p usbl s and $\text{ACC}^{\checkmark}(q, s) \neq \emptyset$ then $\text{ACC}^{\checkmark}(p, s) \neq \emptyset$.

Proof. The proof of this lemma is similar to the proof of Lemma 5.2.20 □

To adapt Lemma 5.2.21 to the pre-order $\sqsubseteq_{\text{P2P}}$ we have to change the way in which the action ✓ is compared with the other actions. According to the compliance client pre-order, the action ✓ is better than any visible action, and this fact is mirrored by the use of $\sqsubseteq^{\checkmark}$ in Lemma 5.2.21. When we use the peer pre-order, ✓ is no longer the best action.

Example 5.3.12. [✓ not best action]
We prove that $\alpha.1 \not\sqsubseteq_{\text{P2P}} 1$. Intuitively, this is true because 1 offers to the peers fewer interactions than $\alpha.1$. In fact, $\alpha.1 \dashv_{\text{P2P}} \overline{\alpha}.1$, as the relation $\{ (\alpha.1, \overline{\alpha.1}), (1\ 1) \}$ is a co-inductive peer compliance. On the contrary, $1 \not\dashv_{\text{P2P}} \overline{\alpha}.1$, because $1 \parallel \overline{\alpha}.1 \stackrel{\tau}{\not\longrightarrow}$ and $\overline{\alpha}.1 \stackrel{\checkmark}{\not\longrightarrow}$. □

Lemma 5.3.13. For every $s \in Act^\star$, and $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \sqsubseteq_{\text{P2P}} q$ and p usbl s, then for every $B \in \text{ACC}^{\checkmark}(q, s)$, there exist a $A \in \text{ACC}^{\checkmark}(p, s)$, such that $A \cap \mathsf{ua}^{\checkmark}(p, s) \subseteq B$.

Proof. The proof of this lemma is similar to the proof of Lemma 5.2.21, and uses Lemma 5.3.11 in place of Lemma 5.2.20. In the current setting, the action ✓ may be in the ready set B, and we replace the server p of Lemma 5.2.21 with a peer r. The proof that $r \not\dashv_{\text{P2P}} q$ depends on the fact that r does not perform ✓. □

Corollary 5.3.14. *For every $s \in Act^\star$, and $p, q \in \mathsf{CCS}_{w\tau}$, if $p \sqsubseteq_{\text{P2P}} q$ and p usbl s, then for every $B \in \text{ACC}^{\checkmark}(q, s)$, there exist a $A \in \text{ACC}^{\checkmark}(p, s)$, such that $A \cap \mathsf{ua}^{\checkmark}(p, s) \sqsubseteq^{\checkmark} B$.*

Proof. It follows from the fact that $A \subseteq B$ implies that $A \sqsubseteq^{\checkmark} B$. □

Lemma 5.3.15. The pre-order $\sqsubseteq_{\text{P2P}}$ is contained in $\preceq_{\text{CLT}}$.

Proof. It follows from Corollary 5.3.14 and Definition 5.2.16. □

Example 5.3.12 proves that converse of Lemma 5.3.15 is not true.

Server-properties of $\sqsubseteq_{\mathbf{p2p}}$

To prove the server-properties of $\sqsubseteq_{\text{P2P}}$ we amend the convergence predicate $\downdownarrows$ (Definition 5.1.4) and $\preceq_{\text{SVR}}$ (Definition 5.1.7) so as to account for the usability of peers.

Definition 5.3.16. [Peer compliance convergence after trace]
For every $s \in Act^\star$ and process p, we write $p \downdownarrows_{\text{P2P}} s$ if and only if p usbl s and $p \downdownarrows s$. □

Definition 5.3.17. Let $p \preccurlyeq_{u\text{SVR}} q$ whenever for every $s \in Act^\star$, if $p \downdownarrows_{\text{P2P}} s$, then

(1) $q \downdownarrows s$

(2) for every $B \in \text{ACC}(q, s)$ there exists some $A \in \text{ACC}(p, s)$ such that $A \cap \mathsf{ua}^{\checkmark}(p, s) \subseteq B$

(3) if $q \stackrel{s}{\Longrightarrow}$, then $p \stackrel{s}{\Longrightarrow}$ □

We adapt Lemma 5.1.9 and Lemma 5.1.11 to the new setting.

Lemma 5.3.18. For every string $s \in Act^\star$, and every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \sqsubseteq_{\text{P2P}} q$ and $p \downdownarrows_{\text{P2P}} s$, then $q \downdownarrows s$.

Proof. The argument is by induction on s.

Base case ($s = \varepsilon$) In this case $p \downdownarrows_{\text{P2P}} \varepsilon$, so $p \in \mathcal{U}^{\dashv_{\text{P2P}}}$ and $p \Downarrow$. We have to prove that $q \downdownarrows \varepsilon$, that is $q \Downarrow$. Since $p \in \mathcal{U}^{\dashv_{\text{P2P}}}$ there exists a peer r such that $p \dashv_{\text{P2P}} r$; since $p \Downarrow$, $r \Downarrow$. The hypothesis $p \sqsubseteq_{\text{P2P}} q$ implies that $q \dashv_{\text{P2P}} r$. Definition 3.2.11 and $r \Downarrow$ ensure that $q \Downarrow$.

Inductive case ($s = \alpha s'$) In this case we have to prove that $q \downdownarrows \alpha s'$. If $q \overset{\alpha}{\not\Longrightarrow}$ then we derive

$$\frac{}{q \downdownarrows \alpha s'}\ q \overset{\alpha}{\not\Longrightarrow};\ [\text{WCONV-AX-NOT}]$$

If $q \overset{\alpha}{\Longrightarrow}$, then let $\hat{q} = \bigoplus(q \text{ AFTER } \alpha)$. Since s' is shorter than s, the inductive hypothesis states the following implication,

$$\text{for every } p', q' \in \mathsf{CCS}_{\text{w}\tau}, \text{ if } p' \sqsubseteq_{\text{P2P}} q', \text{ and } p' \downdownarrows_{\text{P2P}} s', \text{ then } q' \downdownarrows s'.$$

Since $p \in \mathcal{U}^{\dashv_{\text{P2P}}}$, Lemma 5.3.9 and $q \overset{\alpha}{\Longrightarrow}$ imply that $p \overset{\alpha}{\Longrightarrow}$; let $\hat{p} = \bigoplus(p \text{ AFTER } \alpha)$. Lemma 5.3.10 implies that $\hat{p} \sqsubseteq_{\text{P2P}} \hat{q}$. The hypothesis $p \downdownarrows_{\text{P2P}} \alpha s'$ ensures that $\hat{p} \downdownarrows_{\text{P2P}} s'$. We have proven enough facts to use the inductive hypothesis, which implies that $\hat{q} \downdownarrows s'$. Since $q \overset{\alpha}{\Longrightarrow}$, we derive the following fact,

$$\frac{\hat{p} \downdownarrows s'}{q \downdownarrows \alpha s'}\ q \overset{\alpha}{\Longrightarrow};\ [\text{WCONV-ALPHA}]$$

□

Corollary 5.3.19. *For every $s \in Act^\star$, and $p, q \in \mathsf{CCS}_{w\tau}$, if $p \sqsubseteq_{\text{P2P}} q$ and $p \downdownarrows_{\text{P2P}} s$, then for every $B \in \text{ACC}(q, s)$, there exist a $A \in \text{ACC}(p, s)$, such that $A \cap \mathsf{ua}^{\checkmark}(p, s) \subseteq B$.*

Proof. The hypothesis $p \downdownarrows_{\text{P2P}} s$ implies that p usbl s, so the result follows from Lemma 5.3.13 and the facts that for every r and $s' \in Act^\star$ (a) $B \in \text{ACC}^{\checkmark}(r, s')$ if and only if $B \setminus \{\checkmark\} \in \text{ACC}^{\checkmark}(r, s')$; and (b) $A \cap \mathsf{ua}^{\checkmark}(p, s) \subseteq B$ implies that $A \setminus \{\checkmark\} \cap \mathsf{ua}^{\checkmark}(p, s) \subseteq B \setminus \{\checkmark\}$. □

Thanks to these results we can define a server pre-order that takes the usability of peers into the account.

Lemma 5.3.20. For every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, $p \sqsubseteq_{\text{P2P}} q$ if and only if $p \preccurlyeq_{u\text{SVR}} q$.

Proof. Immediate from Lemma 5.3.18, Corollary 5.3.19, and Lemma 5.3.8. □

The converse of the previous lemma is not true.

Example 5.3.21. In this example we prove that $\preccurlyeq_{u\text{SVR}} \not\subseteq \sqsubseteq_{\text{P2P}}$. Let $p = \tau^\infty$ and $q = \beta.\,\mathsf{0}$. Since p diverges and perform no trace, for every $s \in Act^\star$, $p \not\downdownarrows_{\text{P2P}} s$; it follows that $p \preccurlyeq_{u\text{SVR}} q$ is trivially true. However, $p \dashv_{\text{P2P}} \tau^\infty$, whereas $q \not\dashv_{\text{P2P}} \tau^\infty$, so $p \not\sqsubseteq_{\text{P2P}} q$. □

We will need the next two lemmas.

Lemma 5.3.22. For every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \preccurlyeq_{u\text{SVR}} q$ and $q \xrightarrow{\tau} q'$ then $p \preccurlyeq_{u\text{SVR}} q'$.

Proof. The argument is analogous to the proof of Lemma 5.1.13. □

Lemma 5.3.23. For every $p, q \in \mathsf{CCS}_{\text{w}\tau}$, if $p \preccurlyeq_{u\text{SVR}} q$, $p \in \mathcal{U}^{\dashv_{\text{P2P}}}$, and $q \xrightarrow{\alpha} q'$ then

i) the set $(p \text{ AFTER } \alpha)$ is non-empty

ii) $\bigoplus(p \text{ AFTER } \alpha) \preccurlyeq_{u\text{SVR}} \alpha$

Proof. The argument is analogous to the proof of Lemma 5.1.14, with the additional argument that if $\bigoplus(p \text{ AFTER } \alpha)$ usbl s then p usbl αs, which is proven by the derivation

$$\dfrac{\dfrac{\vdots}{\bigoplus(p \text{ AFTER } \alpha) \text{ } \mathsf{usbl} \text{ } s}}{p \text{ } \mathsf{usbl} \text{ } \alpha s} \; p \in \mathcal{U}^{\dashv_{\text{P2P}}}, p \stackrel{\alpha}{\Longrightarrow}; \; [\text{USB-ALPHA}]$$

□

We have seen that up-to the usability of peers, $\sqsubseteq_{\text{P2P}}$ enjoys the properties required by $\preceq_{\text{SVR}}$, and this has lead to the definition of $\preccurlyeq_{u\text{SVR}}$. It is now clear how to define a behavioural characterisation of $\sqsubseteq_{\text{P2P}}$.

Definition 5.3.24. [Semantic compliance peer pre-order]
Let $p \preceq_{\text{P2P}} q$ whenever $p \preceq_{\text{CLT}} q$ and $p \preccurlyeq_{u\text{SVR}} q$. □

The relation $\preceq_{\text{P2P}}$ is preserved by internal and external moves, in the following sense.

Corollary 5.3.25. *For every $p, q \in \mathsf{CCS}_{w\tau}$, if $p \preceq_{\text{P2P}} q$ then*

i) if $q \xrightarrow{\tau} q'$ then $p \preceq_{\text{P2P}} q'$

ii) if $q \xrightarrow{\alpha} q'$ then $(p \text{ AFTER } \alpha) \neq \emptyset$ and $\bigoplus(p \text{ AFTER } \alpha) \preceq_{\text{P2P}} q'$

Proof. The first implication follows from Lemma 5.2.23 and Lemma 5.3.22. The second implication follows form Lemma 5.3.23 and Lemma 5.3.23. □

In Example 5.3.12 and Example 5.3.21 we have seen that the relations $\preceq_{\text{CLT}}$ and $\preccurlyeq_{u\text{SVR}}$ are not sound descriptions of $\sqsubseteq_{\text{P2P}}$. The intersection used in Definition 5.3.24 let us prove the soundness of $\preceq_{\text{P2P}}$.

Theorem 5.3.26. [Alternative characterisation $\sqsubseteq_{\text{P2P}}$]
For every $p, q \in \mathsf{CCS}_{w\tau}$, $p \sqsubseteq_{\text{P2P}} q$ if and only if $p \preceq_{\text{P2P}} q$.

Proof. We are required to prove two implications, that after unwinding the definition of $\preceq_{\text{P2P}}$ become

- if $p \sqsubseteq_{\text{P2P}} q$ then $p \preceq_{\text{CLT}} q$ and $p \preccurlyeq_{u\text{SVR}} p$
- if $p \preceq_{\text{CLT}} q$ and $p \preccurlyeq_{u\text{SVR}} p$ then $p \sqsubseteq_{\text{P2P}} q$

The first implication is proven by Lemma 5.3.15 and Lemma 5.3.20, so we prove only the second one.

We have to show that for every pair $p \preceq_{\text{P2P}} q$, if $p \dashv_{\text{P2P}} r$, then $q \dashv_{\text{P2P}} r$. As we have to exhibit that two processes are in the mutual compliance relation, it is enough to show a co-inductive mutual compliance that contains them.

Let

$$\mathcal{R} = \{ (q, r) \mid p \preceq_{\text{CLT}} q, p \preccurlyeq_{u\text{SVR}} q, \text{ and } r \dashv_{\text{P2P}} p \}$$

We prove that the relation $\mathcal{R}$ is a co-inductive mutual compliance: $\mathcal{R} \subseteq \mathcal{F}^{\dashv_{\text{P2P}}}(\mathcal{R})$.

Fix a pair $q \mathcal{R} r$. There exists a process p such that $p \preceq_{\text{CLT}} q$, $p \preccurlyeq_{u\text{SVR}} q$, and $p \dashv_{\text{P2P}} r$. It follows that r usbl ε.

Definition 3.2.11 requires us to prove the following three facts,

a) $q \Downarrow$ if and only if $r \Downarrow$

b) if $r \parallel q \stackrel{\tau}{\not\longrightarrow}$ then $r \xrightarrow{\checkmark}$ and $q \xrightarrow{\checkmark}$

c) if $r \parallel q \xrightarrow{\tau} r' \parallel q'$ then $r' \mathcal{R} q'$

We prove point (a). We are requires to show two implications,

- if $q \Downarrow$ then $r \Downarrow$,
- if $r \Downarrow$ then $q \Downarrow$

We prove the first implication. Suppose that $q \Downarrow$; we have to explain why $r \Downarrow$. The assumption $q \Downarrow$ implies that $S^{\checkmark}(q) \in \text{ACC}^{\checkmark}(q, \varepsilon)$. Since r usbl ε, Definition 5.2.16 implies that there exists a ready set in $\text{ACC}^{\checkmark}(p, \varepsilon)$. Definition 5.2.14 implies that there exists a p' such that $p \Longrightarrow p' \Downarrow$. Since $p \dashv_{\text{P2P}} r$, Corollary 3.2.7 and the computation $r \parallel p \Longrightarrow r \parallel p'$ ensure that $r' \dashv_{\text{P2P}} p'$. Since $p' \Downarrow$ Definition 3.2.11 implies that $r \Downarrow$.

We have proven the first one of the two implications above; now we prove the second one. Suppose that $r \Downarrow$; we have to prove that $q \Downarrow$. The proof of this is analogous to the proof of point (a) in Theorem 5.1.15.

We have proven point (a). Now we prove point (b): if $r \parallel q \stackrel{\tau}{\not\longrightarrow}$, then $r \xrightarrow{\checkmark}$ and $q \xrightarrow{\checkmark}$. The proof that $q \xrightarrow{\checkmark}$ is analogous to the proof of point (b) in Theorem 5.2.25. The proof that $r \xrightarrow{\checkmark}$ relies on Definition 5.3.17 and is similar to the proof of point (b) in Theorem 5.1.15.

The proof of point (c) is analogous to the proof of point (c) in Theorem 5.1.15, but in this case the argument relies on Corollary 5.3.25 rather than Lemma 5.1.13 and Lemma 5.1.14. □

5.3.1 Relations between pre-orders

The compliance pre-orders and the MUST pre-orders are related alike (see Figure 4.13):

$$\sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}} \subset \sqsubseteq_{\text{P2P}} \subset \sqsubseteq_{\text{CLT}} \tag{5.8}$$

In Figure 5.5 and Figure 5.6 we summarise our knowledge on the pre-orders for processes that we have studied.

To prove that the above set inclusions between the pre-orders above are strict, we use Example 5.3.3 and Example 5.3.12, in which we have shown the ensuing inequalities

$$\begin{array}{rcl@{\qquad\qquad}rcl} \alpha.0 & \sqsubseteq_{\text{P2P}} & 1 & \alpha.0 & \not\sqsubseteq_{\text{SVR}} & 1 \\ \alpha.1 + \beta.1 & \sqsubseteq_{\text{CLT}} & 1 + \beta.1 & \alpha.1 + \beta.1 & \not\sqsubseteq_{\text{P2P}} & 1 + \beta.1 \end{array}$$

The proofs of the set inclusions in (5.8) are straightforward.

Proposition 5.3.27. $\sqsubseteq_{\text{P2P}} \subseteq \sqsubseteq_{\text{CLT}}$

Proof. Thanks to Theorem 5.3.26 the set $\sqsubseteq_{\text{P2P}}$ equals $\preceq_{\text{P2P}}$, and Definition 5.3.24 implies that $\preceq_{\text{P2P}} \subseteq \preceq_{\text{CLT}}$. Theorem 5.2.25 implies that $\sqsubseteq_{\text{P2P}} \subseteq \sqsubseteq_{\text{CLT}}$. □

Proposition 5.3.28. $\sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}} \subseteq \sqsubseteq_{\text{P2P}}$.

Proof. Let $(p_1, p_2) \in \sqsubseteq_{\text{SVR}} \cap \sqsubseteq_{\text{CLT}}$; we are required to prove that if $r \dashv_{\text{P2P}} p_1$, then $r \dashv_{\text{P2P}} p_2$. Fix a process r which mutually complies with p_1; we reason as follows.

$r \dashv_{\text{P2P}} p_1$	By assumption
$r \dashv p_1$ and $p_1 \dashv r$	Thanks to Lemma 3.2.12
$r \dashv p_2$ and $p_1 \dashv r$	Because $p_1 \sqsubseteq_{\text{SVR}} p_2$
$r \dashv p_2$ and $p_2 \dashv r$	Because $p_1 \sqsubseteq_{\text{CLT}} p_2$
$r \dashv_{\text{P2P}} p_2$	Thanks to Lemma 3.2.12

□

$$
\begin{array}{ccccccc}
\sqsubseteq_{\mathrm{SVR}} & & \sqsubseteq_{\mathrm{SVR}} \cap \sqsubseteq_{\mathrm{CLT}} & \subset & \sqsubseteq_{\mathrm{P2P}} & \subset & \sqsubseteq_{\mathrm{CLT}} \\
\neq & & & & \neq & & \neq \\
\sqsubseteq\!\!\!\sim_{\mathrm{SVR}} & & \sqsubseteq\!\!\!\sim_{\mathrm{SVR}} \cap \sqsubseteq\!\!\!\sim_{\mathrm{CLT}} & \subset & \sqsubseteq\!\!\!\sim_{\mathrm{P2P}} & \subset & \sqsubseteq\!\!\!\sim_{\mathrm{CLT}}
\end{array}
$$

Figure 5.5: The relations between MUST and compliance pre-orders on processes.

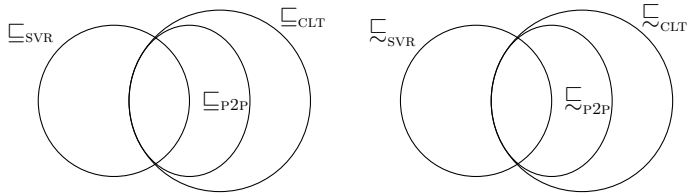


Figure 5.6: Euler diagram of the set theoretic relations between the MUST pre-orders and the compliance pre-orders on processes.

We have fully exhibited the properties of the pre-orders due to MUST and the ones due to the relation $\dashv$. In Figure 5.7 we recall the symbols that we devised so far. Each row of Figure 5.7 contains the notation that we had to introduce in order to characterise the pre-order on the left.

5.4 Related Work

While the server pre-order $\sqsubseteq_{\mathrm{SVR}}$ is inspired by the strong subcontract relations of [Padovani, 2010] and [Castagna et al., 2009], we believe that the client and the peer pre-orders, $\sqsubseteq_{\mathrm{CLT}}$ and $\sqsubseteq_{\mathrm{P2P}}$, are original.

To the best of our knowledge, this thesis presents the first comparison between the refinements for servers generated respectively by the MUST testing and the compliance relation, in an infinite branching LTS with divergence.

We compare our results with the state of the art.

Recall from Section 3.4 the strong compliance and the behavioural compliance, denoted respectively $\dashv^{\mathsf{str}}$ and $\dashv^{\mathsf{bhv}}$.

We have already seen that in the LTS used in [Padovani, 2010], $\langle\, \mathsf{CCS}^{\mathsf{rec},\mathsf{fb},\Downarrow}_{\mathrm{w}\tau}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$, our relation $\dashv$ and the relation $\dashv^{\mathsf{str}}$ coincide. It follows that our $\sqsubseteq^{\mathsf{fb},\Downarrow}_{\mathrm{SVR}}$ is the strong subcontract (Definition 2.2 of the work by Padovani). The alternative characterisation of the pre-order $\sqsubseteq^{\mathsf{fb},\Downarrow}_{\mathrm{SVR}}$ given afterwards in Definition 2.6 (co-inductive strong subcontract) seems to be not complete, as it does not allow

$\sqsubseteq\!\!\!\sim_{\mathrm{SVR}}$:	–	$\Longrightarrow$	AFTER	$\Downarrow$	ACC	–	–
$\sqsubseteq_{\mathrm{SVR}}$:	–	$\Longrightarrow$	AFTER	$\downdownarrows$	ACC	–	–
$\sqsubseteq\!\!\!\sim_{\mathrm{CLT}}$:	$\mathcal{U}^{\mathrm{MUST}}_{\mathrm{CLT}}$	$\Longrightarrow\!\!\!\!\!\!\diagup$	AFTER$\!\!\!\!\diagup$	$\Downarrow^{\checkmark}$	ACC$\!\!\!\!\diagup$	ua$\!\!\!\!\diagup$	usbl$\!\!\!\!\diagup$
$\sqsubseteq_{\mathrm{CLT}}$:	$\mathcal{U}^{\dashv}_{\mathrm{CLT}}$	$\Longrightarrow$	AFTER	$\Downarrow$	ACC$^{\checkmark}$	ua$^{\checkmark}$	usbl
$\sqsubseteq_{\mathrm{P2P}}$:	$\mathcal{U}^{\dashv_{\mathrm{P2P}}}$	–	–	–	–	–	–

Figure 5.7: Predicates to characterise the server, the client, and the peer pre-orders.

$1 \preceq 0$. This glitch is due to the presence of $\checkmark$ in the ready sets.

Since $\sqsubseteq_{\text{SVR}}^{\mathsf{fb},\Downarrow}$ and the strong subcontract of [Padovani, 2010] coincide, our Theorem 5.1.20 implies that server pre-order given by $\dashv^{\mathsf{str}}$ (see Definition 2.2 in that paper) coincides with the MUST testing pre-order. Padovani states that this was proven in [Laneve and Padovani, 2007], but this seems not to be the case, as the definitions in [Laneve and Padovani, 2007] are not the ones used in [Padovani, 2010]. To remark the differences between the two settings, we give some details of [Laneve and Padovani, 2007].

Let $\preceq^{lp07}$ denote the subcontract relation used in [Laneve and Padovani, 2007, Definition 2], and recall from Section 3.4 the notion of constrained contracts. Laneve and Padovani prove a theorem which resembles our Theorem 5.1.20, but is weaker:

Theorem 2 Let $\text{I} = \mathit{names}(\tau)$. $\text{I}[\sigma] \preceq^{lp07} \text{I}[\tau]$ if and only if $\sigma \sqsubseteq_{\text{MUST}} \tau$.

First of all, as constrained contracts are pairs (ie. a set of actions and a behaviour), the subcontract $\preceq^{lp07}$ is not comparable with the MUST pre-order; formally

$$\preceq^{lp07} \not\subseteq \sqsubseteq_{\text{SVR}}, \quad \sqsubseteq_{\text{SVR}} \not\subseteq \preceq^{lp07} \tag{5.9}$$

One may argue that this has no significance, as it is still easy to prove that

$$\text{if } \sigma_1 \sqsubseteq_{\text{SVR}} \sigma_2 \text{ then } \mathit{names}(\sigma_2)[\sigma_1] \preceq^{lp07} \mathit{names}(\sigma_2)[\sigma_2]$$

The converse implication, though, relies heavily on the interfaces of the constrained contracts at hand, and in general is not true:

$$\text{I}[\sigma] \preceq^{07} \text{I'}[\sigma'] \text{ does not imply that } \sigma \sqsubseteq_{\text{SVR}} \sigma' \tag{5.10}$$

For instance, we can prove the following

$$\begin{array}{lll} \emptyset[0] & \preceq^{lp07} & \{\,\alpha\,\}[\alpha.\,0] \\ 0 & \not\sqsubseteq_{\text{MUST}} & \alpha.\,0 \end{array}$$

It follows that *in the sense shown above* the pre-order $\preceq^{lp07}$ is coarser than the MUST pre-order. As a matter of fact, the introduction of [Laneve and Padovani, 2007] states that $\preceq^{lp07}$ *resembles* the MUST pre-order; not that $\preceq^{lp07}$ equals the MUST pre-order.

In Section 3.4 we have discussed the framework used by [Castagna et al., 2009], and pointed out that it is not clear whether our compliance relation equals their strong compliance. As a consequence, the claim (see page 13 of that paper) that on their LTS the MUST pre-order and the strong subcontract (i.e. the server pre-order) coincide does not follow trivially from our results.

The peer pre-order $\sqsubseteq_{\text{P2P}}$ is reminiscent of the symmetric pre-order $\sqsubseteq^{\mathsf{ds}}$ of [Bugliesi et al., 2010]. There the LTS as a number of restrictions, which we do not have in this thesis, so we think that $\sqsubseteq_{\text{P2P}}$ is a generalisation of $\sqsubseteq^{\mathsf{ds}}$, as long as we focus on compositions of *two* peers.

Fair theories Recall the peer refinements $\preceq_O^{-1}$ used by [Bravetti and Zavattaro, 2009]; we already discussed them in Section 4.4. In Example 5.1.16 we have described the intuitions that are also behind the next results. For every action α, the following inequalities are true

$$\begin{array}{lll} \mu x.\,(\,\alpha.x \oplus 1\,) & \sqsubseteq_{\text{P2P}} & \mu x.\,\alpha.x \\ \mu x.\,(\,\alpha.x \oplus 1\,) & \not\preceq_{\{\overline{\alpha}\}}^{-1} & \mu x.\,\alpha.x \end{array}$$

where the peer used to prove the second fact is $\overline{\alpha}.\,1$.

Also the ensuing facts are true,

$$\begin{array}{lll} \mu x.\,\alpha.x & \preceq_{\mathcal{N}}^{-1} & \mu x.\,\beta.x \\ \mu x.\,\alpha.x & \not\sqsubseteq_{\text{P2P}} & \mu x.\,\beta.x \end{array}$$

The first fact holds because no system containing $\mu x.\,\alpha.x$ can be correct (in the sense of [Bravetti and Zavattaro, 2009]). The peer that we use to prove the second fact is $\mu x.\,\overline{\alpha}.x$.

The pre-order $\sqsubseteq_{\text{P2P}}$ and the pre-orders $\preceq_O^{-1}$ are not comparable, but when O is trivial:

Proposition 5.4.1. For every set of output actions O, the following statements are true,

- if O is non-empty, then $\sqsubseteq_{\text{P2P}} \not\subseteq \preceq_O^{-1}$
- $\preceq_O^{-1} \not\subseteq \sqsubseteq_{\text{P2P}}$

Chapter 6

Modelling first-order session types

Thus far we have studied two theories that let us replace processes preserving the correctness of the overall system, and we have explained why the compliance-theory should be preferred over the MUST-theory.

In this chapter we shift our attention from the theories for the general LTS of processes, to the theories for session contracts and their LTS $\langle \mathsf{SC}_{\text{fo}}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$. By doing this, we can try to explain the *behavioural* meaning of the relation $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ in terms of one of the pre-orders we have introduced. This attempt does not work. In a series of examples that go from Example 6.0.2 to Example 6.0.5 we explain why none of the pre-orders that we defined is a fully abstract model of the relation $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$.

By and large, the issue is that the MUST and the compliance pre-orders are defined on the LTS of processes, and not of session contracts. The consequence is that those behavioural pre-orders are more demanding than a model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ should be, for they compare the terms with respect to too many contexts.

Example 6.0.2. [$\sqsubseteq_{\text{SVR}}$ not complete]
Recall the bijection $\mathcal{M}$ defined in Section 2.3. We prove that according to the interpretation $\mathcal{M}$, it is not true that if $T \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S$, then $\mathcal{M}(T) \sqsubseteq_{\text{SVR}} \mathcal{M}(S)$. Let $T = \&\langle \texttt{latte}\colon \text{END} \rangle$ and $S = \&\langle \texttt{latte}\colon \text{END}, \texttt{moka}\colon \text{END} \rangle$. It is relatively easy to prove that the ensuing relation is a co-inductive sub-typing,

$$\{(\&\langle \texttt{latte}\colon \text{END} \rangle, \&\langle \texttt{latte}\colon \text{END}, \texttt{moka}\colon \text{END} \rangle), (\text{END}, \text{END})\}$$

It follows that $T \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S$.

Consider now that images of the types T and S through $\mathcal{M}$;

$$\begin{array}{rcl} \mathcal{M}(T) & = & ?\texttt{latte}.1 \\ \mathcal{M}(S) & = & ?\texttt{latte}.1 + ?\texttt{moka}.1 \end{array}$$

A client that distinguishes these servers is $r = !\texttt{latte}.1 + !\texttt{moka}.0$; for $\mathcal{M}(T)$ MUST r, whereas r and $\mathcal{M}(S)$ in parallel perform a maximal computation that is not client-successful,

$$r \parallel \mathcal{M}(S) \xrightarrow{\tau} 0 \parallel 1 \overset{\tau}{\not\longrightarrow}$$

The computation above is due to the synchronisation on `moka`. □

The previous examples can be used to prove that the compliance server pre-order is not a complete model of the First-order sub-typing.

Example 6.0.3. [$\sqsubseteq_{\text{CLT}}$ not sound]
We prove that $\rho_1 \sqsubseteq_{\text{CLT}} \rho_2$ does not imply that $\mathcal{M}^{-1}(\rho_1) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \mathcal{M}^{-1}(\rho_2)$. The impact of usability of

clients on $\sqsubseteq_{\mathrm{CLT}}$ lets us prove this fact easily. On the one hand $\mu x.\,!\mathtt{moka}.x \sqsubseteq_{\mathrm{CLT}} 1$, while there is no co-inductive sub-typing relation that contains the pair $(\mu X.\,\oplus\langle\,\mathtt{moka}\colon X\,\rangle,\ \mathrm{END})$. □

In Example 6.0.3 the processes we used are session contracts, so the example proves that the relation $\sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}}$ is not a sound model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ either.

Example 6.0.4. [$\sqsubseteq_{\mathrm{CLT}}$ not complete]
We show two session types, T and S that are related by the First-order sub-typing, but whose images through $\mathcal{M}$ are not related by $\sqsubseteq_{\mathrm{CLT}}$. Consider the types

$$\begin{aligned} T &= \&\langle\,\mathtt{latte}\colon \mathrm{END}\,\rangle \\ S &= \&\langle\,\mathtt{latte}\colon \mathrm{END},\ \mathtt{moka}\colon \mathtt{tea}\colon \mathrm{END}\,\rangle \end{aligned}$$

A co-inductive sub-typing that contains the pair $(T,\ S)$ is the following relation

$$\{(T,\ S),\ (\mathrm{END},\ \mathrm{END})\}$$

so $T \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S$. However, the ensuing inequality is true,

$$?\mathtt{latte}.\,1 \not\sqsubseteq_{\mathrm{CLT}} ?\mathtt{latte}.\,1 + ?\mathtt{moka}.?\mathtt{tea}.\,1$$

A server that distinguishes the two process clients above is $p = !\mathtt{moka}.\,0 + !\mathtt{latte}.\,0$. The maximal computations of $\mathcal{M}(T) \parallel p$ are client-successful, whereas there exists the following computation of $\mathcal{M}(S) \parallel p$:

$$\mathcal{M}(S) \parallel p \xrightarrow{\tau} ?\mathtt{tea}.\,1 \parallel 0 \overset{\tau}{\not\longrightarrow}$$

□

The last example proves that neither $\sqsubseteq_{\mathrm{CLT}}$ provides a complete model of First-order sub-typing.

Example 6.0.5. [$\sqsubseteq_{\mathrm{CLT}}$ not sound]
We can easily prove $!\mathtt{Bool}.\,1 \sqsubseteq_{\mathrm{CLT}} 1$; at the same time $![\,\mathtt{Bool}\,];\mathrm{END} \not\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \mathrm{END}$; this proves that $\sqsubseteq_{\mathrm{CLT}}$ does not provide a sound model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$. □

Example 6.0.5 can be adapted to prove that neither $\sqsubseteq_{\mathrm{SVR}}$ is a sound model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$.

The peer pre-orders are contained in the client pre-orders, so the examples above prove that neither the relations $\sqsubseteq_{\mathrm{P2P}}$ and $\sqsubseteq_{\mathrm{P2P}}$ are fully abstract models of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$.

The examples we have shown highlights some features of the model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ that we are looking for:

- the domain we reason on should contain only terms that are images of session types;
- the image of END has to be related only with itself;
- in $\mathsf{CCS}_{\mathrm{w}\tau}$-like languages the image of branch types $\&\langle\,\ldots\,\rangle$ has to allow refinements of the form $\alpha \preceq \alpha + \beta$ to happen.

The definitions of pre-orders that we used in Chapter 4 and Chapter 5 apply equally well to session contracts; but they turn out to be inappropriate, as they compare session contracts from the point of view of satisfying contexts who may be the general processes from Section 2.2. We have shown this in Example 6.0.2 and Example 6.0.4.

In this chapter we tailor the definitions of the server and the client pre-orders so as to consider only the LTS of session contracts. We introduce four pre-orders,

$$\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}},\qquad \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}},\qquad \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}},\qquad \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \tag{6.1}$$

We refer to the pre-orders in (6.1) as the *restricted pre-orders*. Two of the restricted pre-orders are defined using the MUST testing, and the other two are based on the compliance relation. All the new pre-orders differ from the ones of Chapter 4 and Chapter 5. This calls for the definitions of their alternative characterisation; in particular behavioural characterisations.

The alternative pre-orders are the following three,

$$\precsim^{syn}_{\text{SVR}}, \qquad \precsim^{syn}_{\text{CLT}}, \qquad \preceq^{syn}_{\text{CLT}} \tag{6.2}$$

The relations that characterise the four pre-orders in (6.1) are just three. Indeed, we will prove that the relations $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ and $\sqsubseteq^{\mathsf{fo}}_{\preceq_{\text{SVR}}}$ coincide (Corollary 6.4.8), while the relations $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{CLT}}}$ and $\sqsubseteq^{\mathsf{fo}}_{\preceq_{\text{CLT}}}$ differ, and the latter is contained in the former (Corollary 6.5.15).

It may come as a surprise that in (6.1) we do not define the pre-orders for peers. In this setting this is not necessary, because we will use the sub-typing relation $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ as refinement for peers.

The relations in (6.2) are syntax oriented, in that $\precsim^{syn}_{\text{SVR}}$ and $\preceq^{syn}_{\text{CLT}}$ are defined completely looking at the syntax of terms; whereas $\precsim^{syn}_{\text{CLT}}$ is defined using the syntax of terms, but also their usability as clients. Since the pre-orders in (6.2) are also co-inductively defined, it it easy to prove the main results of this chapter,

- the intersections of the restricted server and the client pre-orders is a behavioural description of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$; that is, $\sqsubseteq^{\mathsf{fo}}_{\preceq_{\text{SVR}}} \cap \sqsubseteq^{\mathsf{fo}}_{\preceq_{\text{CLT}}}$ is a fully abstract model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ with respect to the interpretation $\mathcal{M}$ of Section 6.3;
- the pre-order $\sqsubseteq^{\mathsf{fo}}_{\preceq_{\text{SVR}}} \cap \sqsubseteq^{\mathsf{fo}}_{\preceq_{\text{CLT}}}$ coincides with the pre-order $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \cap \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{CLT}}}$ (Proposition 6.5.19).

The second result means that as long as we are concerned with the behaviour that session contracts can express and models of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$, it does not matter whether we pick the MUST testing or the compliance relation as our satisfaction relation, for both choices give rise to the same behavioural explanation of the First-order sub-typing.

Structure of the chapter. In Section 6.1 we define and characterise the pre-order $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$, while in Section 6.2 we define and characterise the pre-order $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{CLT}}}$. Both pre-orders are unsound models of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$, so we use a set intersection to remove the pairs that hinder the soundness of the pre-orders, and in Section 6.3 we prove that the relation $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \cap \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{CLT}}}$ is a fully abstract model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ (see Theorem 6.3.4).

In Section 6.4 and Section 6.5 we study the the restricted pre-orders given by the compliance relation, respectively the one for servers and the one for clients.

6.1 Restricted server pre-order

In this section we introduce a server pre-order for session contracts, $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$. This pre-order is a variation of the MUST server pre-order which is less demanding than the original one (Definition 4.1.1), in that it considers as possible clients only session contracts. The chief result of this section is an alternative characterisation of $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ (Proposition 6.1.10), which is a) co-inductive, and b) syntax oriented. These two properties of the characterisation will help us in showing a fully abstract model of the First-order sub-typing (Theorem 6.3.4).

Let us introduce the new pre-order.

Definition 6.1.1. [Restricted MUST server pre-order]
We write $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_2$ if and only if $\{\, \rho \in \mathsf{SC}_{\text{HO}} \mid \rho \dashv \sigma_1 \,\} \subseteq \{\, \rho \in \mathcal{C}_{\text{HO}} \mid \rho \dashv \sigma_2 \,\}$. The symbol $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ denotes a binary relation that we name *restricted server pre-order*. □

The restricted MUST server pre-order is more generous than $\sqsubseteq_{\precsim_{\text{SVR}}}$, in that it allows refinements of the kind $\alpha \sqsubseteq \alpha + \beta$.

Example 6.1.2. We prove that $?\texttt{latte}.1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} ?\texttt{moka}.1 + ?\texttt{latte}.1$. If a $?\texttt{latte}.1$ MUST ρ then, modulo unfolding, ρ has to be defined by an internal sum. Moreover this sum can only contain the summand $!\texttt{latte}.\rho'$, and therefore $?\texttt{moka}.1 + ?\texttt{latte}.1$ MUST ρ.

Consider now the process $p = !\texttt{latte}.1 + !\texttt{moka}.0$. One can check that $?\texttt{latte}.1$ MUST p, whereas $?\texttt{moka}.1 + ?\texttt{latte}.1 \not\text{MUST}\ p$. It therefore follows that $?\texttt{latte}.1 \not\sqsubseteq_{\simeq_{\mathrm{SVR}}} ?\texttt{moka}.1 + ?\texttt{latte}.1$. □

Example 6.1.2 proves that $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$ does not enjoy point (1b) of Definition 4.1.17; so in the LTS

$$\langle\, \mathsf{SC}_{\mathrm{fo}},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \,\rangle$$

the pre-order $\precsim_{\mathrm{SVR}}$ is not a complete characterisation of the must server pre-order.

Example 6.1.3. [e-vote, ballot refinement]
We give a more concrete instance of the previous example. Recall Example 2.3.5 and consider the session contract

$$\begin{aligned}\mathrm{BallotB} = \mu x.\, ?\texttt{Login}.(\, &!\texttt{Wrong}.1 \oplus \\ &!\texttt{Ok}.(\, ?\texttt{VoteA}.1 + ?\texttt{VoteB}.1 + ?\texttt{VoteC}.1 + ?\texttt{VoteD}.1\,)\,)\end{aligned}$$

BallotB offers to a voter more options than Ballot, and intuitively it should be possible to use a server that guarantees BallotB in place of a server that guarantees Ballot. This is not the case if the contracts are compared with $\sqsubseteq_{\simeq_{\mathrm{SVR}}}$, because Ballot $\not\sqsubseteq_{\simeq_{\mathrm{SVR}}}$ BallotB. On the other hand, if we restrict our attention to session contracts, and thus to the pre-order $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$, we have Ballot $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$ BallotB. □

Lemma 6.1.4. For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\mathrm{fo}}$, $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} \sigma_2$ if and only if $\mathrm{UNF}(\sigma_1) \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} \mathrm{UNF}(\sigma_2)$.

Proof. This lemma follows from Lemma 3.1.9. □

Lemma 6.1.5. [Bottom element]
The pre-order $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$ enjoys the following properties,

(i) it has a bottom element

(ii) if $\sigma_\perp$ is a bottom element of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$ then $\mathrm{UNF}(\sigma_\perp) = 1$

Proof. To prove point (i) we show that 1 is a bottom element of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$, that is $1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} \sigma$ for every session contract σ. Let ρ be a session contract such that 1 MUST ρ. The session contract 1 offers no interaction. Because of the restricted syntax of session contracts, ρ must also be, modulo unfolding, the session contract 1. Now fix a session contract σ. Clearly σ MUST 1, therefore from an application of Proposition 3.2.10 it follows that $\rho \dashv \sigma$.

To prove part point (ii) let $\sigma_\perp$ be an arbitrary bottom element of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$. We are required to show that $\mathrm{UNF}(\sigma_\perp) = 1$. From the definition of bottom element follows $\sigma_\perp \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} 1$. An application of Lemma 6.1.4 gives $\mathrm{UNF}(\sigma_\perp) \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} 1$. Now an analysis of the possible syntactic structure of $\mathrm{UNF}(\sigma_\perp)$ yields that it must be 1 itself. □

Point (ii) of Lemma 6.1.5 is relevant because 1 is not the only bottom element; for example it is also true that $\mu X.\, 1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} \sigma$ for every σ.

The bottom elements of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$ make this pre-order a non sound model of the sub-typing $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$.

Example 6.1.6. Recall that $\mathcal{M}(\mathrm{END}) = 1$. In the restricted MUST server pre-order the session contract 1 is a least element, being smaller or equal to every other session contract. On the other

$$\frac{}{1 \precsim^{syn}_{\text{SVR}} \sigma_2} \text{ [AX-SRV]}$$

$$\frac{\sigma_1' \precsim^{syn}_{\text{SVR}} \sigma_2'}{!\mathtt{t}_1.\sigma_1' \precsim^{syn}_{\text{SVR}} !\mathtt{t}_2.\sigma_2'} \ \mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1; \text{ [R-OUT-F]}$$

$$\frac{\sigma_1' \precsim^{syn}_{\text{SVR}} \sigma_2'}{?\mathtt{t}_1.\sigma_1' \precsim^{syn}_{\text{SVR}} ?\mathtt{t}_2.\sigma_2'} \ \mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2; \text{ [R-IN-F]}$$

$$\frac{\sigma_1^1 \precsim^{syn}_{\text{SVR}} \sigma_1^2 \ \ldots \ \sigma_{|I|}^1 \precsim^{syn}_{\text{SVR}} \sigma_{|I|}^2}{\sum_{i\in I} ?\mathtt{l_i}.\sigma_i^1 \precsim^{syn}_{\text{SVR}} \sum_{j\in J} ?\mathtt{l_j}.\sigma_j^2,} \ I \subseteq J \text{ [R-BRANCH]}$$

$$\frac{\sigma_1^1 \precsim^{syn}_{\text{SVR}} \sigma_1^2 \ \ldots \ \sigma_{|J|}^1 \precsim^{syn}_{\text{SVR}} \sigma_{|J|}^2}{\bigoplus_{i\in I} ?\mathtt{l_i}.\sigma_i^1 \precsim^{syn}_{\text{SVR}} \bigoplus_{j\in J} ?\mathtt{l_j}.\sigma_j^2} \ J \subseteq I; \text{ [R-CHOICE]}$$

$$\frac{\text{UNF}(\sigma_1) \precsim^{syn}_{\text{SVR}} \text{UNF}(\sigma_2)}{\sigma_1 \precsim^{syn}_{\text{SVR}} \sigma_2} \ \mathit{depth}(\sigma_1) + \mathit{depth}(\sigma_2) > 0; \text{ [R-UNFOLD]}$$

Figure 6.1: Inference rules for the rule functional $\mathcal{F}^{\precsim^{syn}_{\text{SVR}}}$

hand, for session types END $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ T if and only if UNF(T) = END. Consequently the relation $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ is an unsound model for sub-typing between session types. For example:

$$1 \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} !\mathtt{Bool}.\,1, \quad \text{END} \not\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} !\lceil \mathtt{Bool} \rceil; \text{END}$$

□

Although the restricted MUST server pre-order is not a fully abstract model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$, we provide a characterisation of this restricted pre-order. The reason is that in Section 6.3 we will use $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ to define a fully abstract model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$.

Lemma 6.1.7. Let σ_1 and σ_2 be session contracts such that $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_2$. The following properties are true,

(a) if UNF$(\sigma_1) = !\mathtt{t}_1.\sigma_1'$ then UNF$(\sigma_2) = !\mathtt{t}_2.\sigma_2'$, $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $\sigma_1' \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_2'$

(b) if UNF$(\sigma_1) = ?\mathtt{t}_1.\sigma_1'$ then UNF$(\sigma_2) = ?\mathtt{t}_2.\sigma_2'$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and $\sigma_1' \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_2'$

(c) if UNF$(\sigma_1) = \sum_{i\in I} ?\mathtt{l_i}.\sigma_i^1$ then UNF$(\sigma_2) = \sum_{j\in J} ?\mathtt{l_j}.\sigma_j^2$, with $I \subseteq J$ and $\sigma_i^1 \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_i^2$

(d) if UNF$(\sigma_1) = \bigoplus_{i\in I} !\mathtt{l_i}.\sigma_i^1$ then UNF$(\sigma_2) = \bigoplus_{j\in J} !\mathtt{l_j}.\sigma_j^2$, with $J \subseteq I$ and $\sigma_j^1 \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_j^2$

Proof. We discuss the point (a), as the arguments for the other points of the lemma are analogous.

Let UNF$(\sigma_1) = !\mathtt{t}_1.\sigma_1'$; we have to prove that UNF$(\sigma_2) = !\mathtt{t}_2.\sigma_2'$, for some base type $\mathtt{t}_2$ such that $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$, and some session contract σ_2' such that $\sigma_1' \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_2'$.

We begin by defining a client that is passed by σ_1'. Pick a ρ' such that σ_1' MUST ρ', and let $\rho = ?\mathtt{t}_1.\rho'$. It is relatively easy to prove that σ_1 MUST ρ.

The hypothesis $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \sigma_2$ imply that σ_2 MUST ρ. Lemma 6.1.4 implies that UNF$(\sigma_1) \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ UNF(σ_2). Given the form of ρ, it follows that UNF$(\sigma_2) = !\mathtt{t}_2.\sigma_2'$, for some $\mathtt{t}_1$ such that $?\mathtt{t}_1 \bowtie_c !\mathtt{t}_2$. Since UNF$(\sigma_2) \overset{\checkmark}{\not\longrightarrow}$, and $\rho \,||\, \text{UNF}(\sigma_2) \xrightarrow{\tau} \rho' \,||\, \sigma_2'$, it follows that σ_2' MUST ρ'. Since the only hypothesis on ρ' is that σ_1' MUST ρ', it follows that $\sigma_1' \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \rho'$. □

The properties of $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ exhibited in Lemma 6.1.7, let us define the alternative characterisation of the restricted MUST server pre-order.

Definition 6.1.8. [Syntactic server pre-order]
Let $\mathcal{F}^{\precsim^{syn}_{\text{SVR}}} : \mathcal{P}(\mathsf{SC}^2_{\text{fo}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{fo}})$ be the rule functional given by the inference rules in Figure 6.1. If $X \subseteq \mathcal{F}^{\precsim^{syn}_{\text{SVR}}}(\mathcal{R})$, then we say that X is a syntactic server pre-order. Lemma C.0.26 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\precsim^{syn}_{\text{SVR}}}(\mathcal{R})$; we call this solution the *syntactic server pre-order*, and we denote it $\precsim^{syn}_{\text{SVR}}$. That is $\precsim^{syn}_{\text{SVR}} = \nu X.\mathcal{F}^{\precsim^{syn}_{\text{SVR}}}(\mathcal{R})$. □

Corollary 6.1.9. *The restricted* MUST *server pre-order is a co-inductive syntactic server pre-order. Formally,* $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \subseteq \mathcal{F}^{\precsim^{syn}_{\text{SVR}}}(\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}})$

Proof. It follows from Lemma 6.1.7. □

Proposition 6.1.10. [Co-inductive characterisation $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$]
The restricted MUST server pre-order and the syntactic server pre-order coincide. Formally, $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} = \precsim^{syn}_{\text{SVR}}$.

Proof. We have to prove two set inclusions, namely

a) $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}} \subseteq \precsim^{syn}_{\text{SVR}}$

b) $\precsim^{syn}_{\text{SVR}} \subseteq \sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$

The first set inclusion follows form the fact that $\sqsubseteq^{\mathsf{fo}}_{\precsim_{\text{SVR}}}$ is a co-inductive server pre-order (Corollary 6.1.9), and the Knaster-Tarski theorem, which ensures that $\precsim^{syn}_{\text{SVR}} = \bigcup\{\, \mathcal{R} \,|\, \mathcal{R} \subseteq \mathcal{F}_{\precsim^{syn}_{\text{SVR}}}(\mathcal{R}) \,\}$.

To prove the second set inclusion, we show a more general result. Let $\mathcal{S}$ be a co-inductive server pre-order. We prove the following implication

$$\text{if } \sigma_1 \;\mathcal{S}\; \sigma_2 \text{ and } \sigma_1 \text{ MUST } \rho \text{ then } \sigma_2 \text{ MUST } \rho.$$

Let

$$\mathcal{R} = \{\, (\sigma_2,\, \rho) \mid \sigma_1 \text{ MUST } \rho, \text{ and } \sigma_1 \;\mathcal{S}\; \sigma_2 \text{ for some } \sigma_1 \in \mathsf{SC}_{\text{fo}} \,\}$$

the proof of the implication above amounts in showing the set inclusion $\mathcal{R} \subseteq$ MUST.

In view of Lemma 3.3.14, we know the equalities

$$\begin{array}{rcl} \mathcal{R} & = & \{\, (\sigma_2,\, \rho) \mid \rho \text{ MUST}^{\text{s}} \sigma_1, \text{ and } \sigma_1 \;\mathcal{S}\; \sigma_2 \text{ for some } \sigma_1 \in \mathsf{SC}_{\text{fo}} \,\} \\ & = & (\text{MUST}^{\text{s}} \circ \mathcal{S})^{-1} \end{array}$$

and the set inclusion that we want to prove becomes $(\text{MUST}^{\text{s}} \circ \mathcal{S})^{-1} \subseteq \text{MUST}^{\text{s}\ -1}$. It suffices to prove that $\text{MUST}^{\text{s}} \circ \mathcal{S} \subseteq \text{MUST}^{\text{s}}$.

Fix a pair $\rho\ (\text{MUST}^{\text{s}} \circ \mathcal{S})\ \sigma_2$; by construction there exists a σ_1 such that ρ MUST$^{\text{s}}$ σ_1 and $\sigma_1\ \mathcal{S}\ \sigma_2$. The argument to prove that ρ MUST$^{\text{s}}$ σ_2 is by rule induction on the derivation of

$$\frac{\vdots}{\rho \text{ MUST}^{\text{s}} \sigma_1} \tag{6.3}$$

In the base case the last rule used in the derivation is the axiom [A-UNIT], thus $\rho = \mathsf{1}$, and we can derive

$$\frac{}{\rho \text{ MUST}^{\text{s}} \sigma_2}\ [\text{A-UNIT}]$$

In the inductive case, we have five subcases to discuss. Since all the arguments are similar, we give the details of just two cases.

- Suppose the the last rule applied in the derivation (6.3) is [R-Alpha]. The derivation has the shape

$$\dfrac{\dfrac{\vdots}{\rho \text{ MUST}^{\text{s}} \sigma_1'}}{!\mathtt{t}.\rho' \text{ MUST}^{\text{s}} ?\mathtt{t}_1.\sigma_1'} \; ?\mathtt{t}_1 \bowtie_c !\mathtt{t};\ [\text{R-Alpha}]$$

  so $\sigma_1 = ?\mathtt{t}_1.\sigma_1'$.

  Knowing the form of σ_1, we show the form of σ_2. By hypothesis $\mathcal{S} \subseteq \mathcal{F}_{\precsim_{\text{SVR}}^{syn}}(\mathcal{S})$, so we know that $(\sigma_1, \sigma_2) \in \mathcal{F}_{\precsim_{\text{SVR}}^{syn}}(\mathcal{S})$; and, given the form of σ_1 and Definition 6.1.8 it must be the case that $\text{UNF}(\sigma_2) = ?\mathtt{t}_2.\sigma_2'$ for some $\mathtt{t}_2$ such that $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$, and $\sigma_2' \in \mathsf{SC}_{\text{fo}}$ such that $\sigma_1' \; \mathcal{S} \; \sigma_2'$.

  We are now ready to show the derivation of $\rho \text{ MUST}^{\text{s}} \sigma_2$. As that the derivation of $\rho' \text{ MUST}^{\text{s}} \sigma_1'$ is shorter than the derivation in (6.3), and $\rho' \text{ MUST}^{\text{s}} \sigma_1' \; \mathcal{S} \; \sigma_2'$, we can use the inductive hypothesis to state that there exists the following derivation

$$\dfrac{\vdots}{\rho \text{ MUST}^{\text{s}} \sigma_2'}$$

  We extend the derivation above with an application of the rule [R-Alpha]. First, we check that the side conditions are true. We have proven that $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$. The last fact and $\mathtt{t} \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ implies that $\mathtt{t} \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$, so $?\mathtt{t}_2 \bowtie_c !\mathtt{t}$. Now we apply the rule and obtain the derivation

$$\dfrac{\dfrac{\vdots}{\rho' \text{ MUST}^{\text{s}} \sigma_2'}}{!\mathtt{t}.\rho' \text{ MUST}^{\text{s}} ?\mathtt{t}_2.\sigma_2'} \; ?\mathtt{t}_2 \bowtie_c !\mathtt{t};\ [\text{R-Alpha}]$$

  We have derived $\rho_1 \text{ MUST}^{\text{s}} \text{ UNF}(\sigma_2)$. If $\mathit{depth}(\sigma_2) = 0$, then we have derived $\rho_1 \text{ MUST}^{\text{s}} \sigma_2$. If $\mathit{depth}(\sigma_2) > 0$, then we extend the derivation above as follows,

$$\dfrac{\dfrac{\dfrac{\vdots}{\rho' \text{ MUST}^{\text{s}} \sigma_2'}}{!\mathtt{t}.\rho' \text{ MUST}^{\text{s}} ?\mathtt{t}_2.\sigma_2'} \; ?\mathtt{t}_2 \bowtie_c !\mathtt{t};\ [\text{R-Alpha}]}{\rho_1 \text{ MUST}^{\text{s}} \sigma_2} \; \mathit{depth}(\rho_1) + \mathit{depth}(\sigma_2) > 0;\ [\text{R-Unfold}]$$

- Suppose that the last rule used in the derivation (6.3) is [R-Unfold]. The premises of the rule ensures that there exists a derivation of $\text{UNF}(\rho) \text{ MUST}^{\text{s}} \text{ UNF}(\sigma_1)$. By construction $\sigma_1 \; \mathcal{S} \; \sigma_2$; the hypothesis that $\mathcal{S}$ is a syntactic server pre-order ensures that $\text{UNF}(\sigma_1) \; \mathcal{S} \; \text{UNF}(\sigma_2)$.

  It follows that $\text{UNF}(\rho) \text{ MUST}^{\text{s}} \text{ UNF}(\sigma_1) \; \mathcal{S} \; \text{UNF}(\sigma_2)$. Now, since the derivation of $\text{UNF}(\rho) \text{ MUST}^{\text{s}} \text{ UNF}(\sigma_1)$ is shorter than the derivation in (6.3), we can apply the inductive hypothesis, which ensures that there exists the derivation

$$\dfrac{\vdots}{\text{UNF}(\rho) \text{ MUST}^{\text{s}} \text{ UNF}(\sigma_2)}$$

  If $\mathit{depth}(\rho) + \mathit{depth}(\sigma_2) = 0$, then $\text{UNF}(\rho) = \rho$ and $\text{UNF}(\sigma_2) = \sigma_2$, so $\text{UNF}(\rho) \text{ MUST}^{\text{s}} \text{ UNF}(\sigma_2)$ implies that $\rho \text{ MUST}^{\text{s}} \rho_2$.

  If $\mathit{depth}(\rho) + \mathit{depth}(\sigma_2) > 0$, we can extend the derivation above,

$$\dfrac{\dfrac{\vdots}{\text{UNF}(\rho) \text{ MUST}^{\text{s}} \text{ UNF}(\sigma_2)}}{\rho \text{ MUST}^{\text{s}} \sigma_2} \; \mathit{depth}(\rho) + \mathit{depth}(\sigma_2) > 0;\ [\text{R-Unfold}]$$

Since the relations $\precsim^{syn}_{\text{SVR}}$ is a co-inductive syntactic server pre-order the implication that we have proven ensures that $\precsim^{syn}_{\text{SVR}} \subseteq \sqsubseteq^{\mathsf{fo}}_{\text{SVR}}$. □

To prove the previous proposition we have used Lemma 3.3.14. This is not necessary, and there exists a direct proof of the Proposition 6.1.10 that use the definition of MUST. We have presented a proof based on Lemma 3.3.14 because the direct proof is almost identical to the proof of Proposition 6.4.7.

To prove Proposition 6.4.7, we will need a weaker version of Lemma 4.1.15 that explains how the acceptance sets of session contracts in $\precsim^{syn}_{\text{SVR}}$ are related. In this context, the acceptance sets are related as long as no visible is performed.

Lemma 6.1.11. Let $\mathcal{R}$ be a co-inductive syntactic server pre-order. For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\text{fo}}$, if $\sigma_1 \mathcal{R} \sigma_2$ and $B \in \text{ACC}(\sigma_2, \varepsilon)$, then there exists a set $A \in \text{ACC}(\sigma_1, \varepsilon)$ such that $A \subseteq B$.

Proof. Fix a set $B \in \text{ACC}(\sigma_2, \varepsilon)$; since $\text{ACC}(\sigma_2, \varepsilon) = \text{ACC}(\text{UNF}(\sigma_2), \varepsilon)$, $B \in \text{ACC}(\text{UNF}(\sigma_2), \varepsilon)$.

According to the cases in Definition 6.1.8 and a case analysis on the form of $\text{UNF}(\sigma_2)$, one can show an $A \in \text{ACC}(\text{UNF}(\sigma_1), \varepsilon)$ which satisfies the required properties. We discuss two cases. If $\text{UNF}(\rho_1) = \mathsf{1}$, then $\text{ACC}(\text{UNF}(\sigma_1), \varepsilon) = \{\emptyset\}$, and $\emptyset \subseteq B$. If $\text{UNF}(\sigma_1) = !\mathtt{t}_1.\sigma_1'$, then $B = \{\mathtt{t}_2\}$ for some $\mathtt{t}_2$ such that $\mathtt{t}_2 \preccurlyeq_{\mathsf{b}} \mathtt{t}_1$. Since $\{!\mathtt{t}_1\} \in \text{ACC}(\sigma_1, \varepsilon)$, the singleton $\{!\mathtt{t}_1\}$ is the A we are after. We leave the details of the other case analysis to the reader.

Since $\sigma_1 \Longrightarrow \text{UNF}(\sigma_1)$, $\text{ACC}(\text{UNF}(\sigma_1), \varepsilon) \subseteq \text{ACC}(\sigma_1, \varepsilon)$, so $A \in \text{ACC}(\sigma_1, \varepsilon)$. □

In this section we have introduced and characterised the server pre-order that results from restricting our attention on the LTS of session contracts $\langle \mathsf{SC}_{\text{fo}}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$. In the next section we perform a similar task, but focusing on the client side.

6.2 Restricted must client pre-order

We define the restricted MUST client pre-order in the obvious way, and then we characterise it.

Definition 6.2.1. [Restricted MUST client pre-order]
If ρ_1 and ρ_2 are session contracts, then we write $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2$ whenever for every $\sigma \in \mathsf{SC}_{\text{fo}}$, σ MUST ρ_1 implies that σ MUST ρ_2. We call the relation denoted by the symbol $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ the restricted MUST client pre-order. □

The new pre-order differs from the original $\sqsubseteq_{\text{CLT}}$ of Definition 4.2.1.

Example 6.2.2. We have shown in Example 6.1.2 that $?\mathtt{latte}.\mathsf{1} \sqsubseteq^{\mathsf{fo}}_{\text{SVR}} ?\mathtt{moka}.\mathsf{1} + ?\mathtt{latte}.\mathsf{1}$. A similar argument, this time applied to client-side session contracts, can be used to show that

$$?\mathtt{latte}.\mathsf{1} \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} ?\mathtt{moka}.?\mathtt{moka}.\mathsf{1} + ?\mathtt{latte}.\mathsf{1}$$

Similarly to what happens for server session contracts, if we turn our attention to processes then the session contracts above are no longer related. Let us see why. The client $?\mathtt{latte}.\mathsf{1}$ is satisfied by the server $!\mathtt{latte}.\mathsf{1} + !\mathtt{moka}.\mathsf{1}$, because the action $!\mathtt{moka}$ will never be performed by the server. On the other hand $?\mathtt{moka}.?\mathtt{moka}.\mathsf{1} + ?\mathtt{latte}.\mathsf{1} \,||\, !\mathtt{latte}.\mathsf{1} + !\mathtt{moka}.\mathsf{1} \xrightarrow{\tau} ?\mathtt{moka}.\mathsf{1} \,||\, \mathsf{1} \not\xrightarrow{\tau}$. and $?\mathtt{moka}.\mathsf{1} \not\xrightarrow{\checkmark}$; this proves that $!\mathtt{latte}.\mathsf{1} + !\mathtt{moka}.\mathsf{1} \not\text{MUST} ?\mathtt{moka}.?\mathtt{moka}.\mathsf{1} + ?\mathtt{latte}.\mathsf{1}$

This argument above would have proves that $!\mathtt{latte}.\mathsf{1} \not\sqsubseteq_{\text{CLT}} ?\mathtt{moka}.?\mathtt{moka}.\mathsf{1} + ?\mathtt{latte}.\mathsf{1}$. We have therefore shown that $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \not\subseteq \sqsubseteq_{\text{CLT}}$. □

Lemma 6.2.3. For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\text{fo}}$, $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \sigma_2$ if and only if $\text{UNF}(\sigma_1) \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \text{UNF}(\sigma_2)$.

Proof. This lemma follows from Lemma 3.1.9. □

In Lemma 6.1.5 we saw that the pre-order $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}}$ has bottom elements; the pre-order $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ enjoys both the same property, and the dual one: $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ has top and bottom elements.

Lemma 6.2.4. [Top element]
The pre-order $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ enjoys the following two properties,

(i) it has a top element

(ii) if $\sigma_\top$ is a top element of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ then $\mathrm{UNF}(\sigma_\top) = \mathsf{1}$

Proof. Since σ MUST $\mathsf{1}$ for every session contract σ, the session contract $\mathsf{1}$ it is a top element in the restricted compliance client pre-order. Moreover, reasoning as in Lemma 6.1.5 we can show that if $\sigma_\top$ is an arbitrary top element then $\mathrm{UNF}(\sigma_\top) = \mathsf{1}$. □

Intuitively, the bottom elements in $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ are the clients that are not not satisfied by any server. These clients are the session contracts that never perform $\checkmark$, for instance $\mu x.\,!\mathtt{Int}.x$.

Lemma 6.2.5. [Bottom] For every $\rho \in \mathsf{SC}_{\mathrm{fo}}$, and every $\mathtt{t} \in \mathtt{BT}$, $\mu x.\,!\mathtt{t}.x \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}} \rho$.

Proof. Let ρ be a session contract. The argument to prove that $\mu x.\,!\mathtt{t}.x \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}} \rho$ does not depend on ρ. The inequality is true because for every session contract σ, no maximal computation of $\mu x.\,!\mathtt{t}.x \,||\, \sigma$ is client-successful. □

Definition 6.2.6. [Usable clients]
Let

$$\mathcal{U}^{\mathsf{SC}_{\mathrm{fo}}}_{\mathrm{CLT}} = \{\, \rho \in \mathsf{SC}_{\mathrm{fo}} \mid \sigma \text{ MUST } \rho \text{ for some } \sigma \in \mathsf{SC}_{\mathrm{fo}} \,\}$$

□

The existence of top elements of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ renders the restricted MUST client pre-order a non sound model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$.

Example 6.2.7. Recall that $\mathcal{M}(\mathrm{END}) = \mathsf{1}$. The restricted MUST client pre-order presents the dual issue as it relates every session contract to $\mathsf{1}$; it is one of the top element. Once again a model based on $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ would be unsound:

$$!(!\mathtt{Int}).\,\mathsf{1} \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}} \mathsf{1}, \quad ![\,!\mathtt{Int}\,];\mathrm{END} \not\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \mathrm{END}$$

□

The relation $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ is not a sound model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ also because of the non usable clients.

Example 6.2.8. [$\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$ not sound with respect to $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$] Also in this example we prove that there exists session contracts ρ_1 and ρ_2 such that $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}} \rho_2$ and $\mathcal{M}^{-1}(\rho_1) \not\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \mathcal{M}^{-1}(\rho_2)$. In this example, though, we leverage the non-trivial usability of clients rather than the top elements of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}}$.

We discuss first some preliminary facts. It is relatively easy to prove the following inequalities

$$\begin{array}{rcl} \mu x.\,!\mathtt{Bool}.x & \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}} & ?\mathtt{Int}.\,\mathsf{1} \\ ?\mathtt{moka}.\,\mathsf{1} + ?\mathtt{latte}.\,\mathsf{1} + ?\mathtt{stout}.\mu x.\,!\mathtt{Bool}.x & \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}} & ?\mathtt{moka}.\,\mathsf{1} + ?\mathtt{latte}.\,\mathsf{1} + ?\mathtt{tea}.\sigma \end{array}$$

The first inequality is true because $\mu x.\,!\mathtt{Real}.x$ and $\mu x.\,!\mathtt{Bool}.x$ are not usable. Intuitively, the third inequality is true because the action $?\mathtt{stout}$ is not usable, so it is safe to drop it altogether. Let $\rho_1 = ?\mathtt{moka}.\,\mathsf{1} + ?\mathtt{latte}.\,\mathsf{1} + ?\mathtt{stout}.\mu x.\,!\mathtt{Bool}.x$ and $\rho_2 = ?\mathtt{moka}.\,\mathsf{1} + ?\mathtt{latte}.\,\mathsf{1} + ?\mathtt{tea}.\sigma$. If σ MUST ρ_1, then the unfolding of σ must be an internal sum, possibly with just one summand. The term $\mathrm{UNF}(\sigma)$ can interact *only* with the usable actions of ρ; since these actions are $?\mathtt{moka}$ and $?\mathtt{latte}$, and they are both offered by ρ_2, one can prove that σ MUST ρ_2.

All the inequalities that we have shown prove that $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$ is not a sound model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$; that is, the terms related by $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$ above, have images via $\mathcal{M}$ not related by the sub-typing:

$$\begin{array}{lll} \mu X.\,![\,\texttt{Bool}\,];X & \not\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} & ?[\,\texttt{Int}\,];\text{END} \\ S_1 & \not\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} & S_2 \end{array}$$

where $S_1 = \&\langle\, \texttt{moka}\colon \text{END},\ \texttt{latte}\colon \text{END},\ \texttt{stout}\colon \mu X.\texttt{Bool}X \,\rangle$ and $S_2 = \&\langle\, \texttt{moka}\colon \text{END},\ \texttt{latte}\colon \text{END},\ \texttt{tea}\colon \mathcal{M}^{-1}(\sigma) \,\rangle$. □

In Lemma 6.1.7 we exhibited the properties of the restricted MUST server pre-order. A similar result holds for the restricted MUST client pre-order, up-to the impact of non usable clients.

Lemma 6.2.9. Let ρ_1 and ρ_2 be two session contracts such that $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \rho_2$. If $\rho_1 \in \mathcal{U}^{\mathsf{SC}_{\text{fo}}}_{\text{CLT}}$ and $\text{UNF}(\rho_2) \neq \mathbf{1}$, then one of the following is true

(i) if $\text{UNF}(\rho_1) = !\mathtt{t}_1.\rho_1'$, then $\text{UNF}(\rho_2) = !\mathtt{t}_2.\rho_2'$ $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $\rho_1' \precsim^{syn}_{\text{CLT}} \rho_2'$

(ii) if $\text{UNF}(\rho_1) = ?\mathtt{t}_1.\rho_1'$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ then $\text{UNF}(\rho_2) = ?\mathtt{t}_2.\rho_2'$ and $\rho_1' \precsim^{syn}_{\text{CLT}} \rho_2'$

(iii) if $\text{UNF}(\rho_1) = (\sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho_i^1) + (\sum_{k\in K} ?\mathtt{l}_\mathtt{k}.\rho_k^1)$ with $\rho_k^1 \in \mathcal{U}^{\mathsf{SC}_{\text{fo}}}_{\text{CLT}}$ for every $k \in K$, and $\rho_i^1 \notin \mathcal{U}^{\mathsf{SC}_{\text{fo}}}_{\text{CLT}}$ for every $i \in I$ then $\text{UNF}(\rho_2) = \sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho_j^2$ with $K \subseteq J$, and $\rho_n^1 \precsim^{syn}_{\text{CLT}} \rho_n^2$ for every $n \in (I \cup K) \cap J$,

(iv) if $\text{UNF}(\rho_1) = \bigoplus_{i\in I} !\mathtt{l}_\mathtt{i}.\rho_i^1$ then $\text{UNF}(\rho_2) = \bigoplus_{j\in J} !\mathtt{l}_\mathtt{j}.\rho_j^2$ with $J \subseteq I$ and $\rho_j^1 \precsim^{syn}_{\text{CLT}} \rho_j^2$ for every $i \in J$

Proof. The proof is almost the same of Lemma 6.1.7. The main difference is that to define a session contract σ that satisfies the ρ_1 at hand, we need ρ_1 to be usable. This has an impact on how external sums are related by $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$, so we present the proof of point (iii).

Fix two session contracts such that $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \rho_2$ and $\rho_1 \in \mathcal{U}^{\mathsf{SC}_{\text{fo}}}_{\text{CLT}}$, and suppose that

$$\text{UNF}(\rho_1) = (\sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho_i^1) \ + \ (\sum_{k\in K} ?\mathtt{l}_\mathtt{k}.\rho_k^1)$$

for some finite sets I and K. Let $\hat{k}$ be the cardinality of K. We have to prove that

a) $\text{UNF}(\rho_2) = \sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho_j^2$

b) $K \subseteq J$

c) $\rho_n^1 \precsim^{syn}_{\text{CLT}} \rho_n^2$ for every $n \in I \cap J$

Since $\rho_1 \in \mathcal{U}^{\mathsf{SC}_{\text{fo}}}_{\text{CLT}}$ also $\text{UNF}(\rho_1)$ is usable. This implies that the set K must be non-empty. For each $k \in K$ let σ_k a session contract such that ρ_k^1 MUST^{s} σ_k, and let $\sigma = \bigoplus_{k\in K} !\mathtt{t}_\mathtt{k}.\sigma_k$. Now we derive

$$\frac{\begin{array}{c}\vdots\\ \rho_1^1 \ \text{MUST}^{\text{s}}\ \sigma_1\end{array} \quad \ldots \quad \begin{array}{c}\vdots\\ \rho_{\hat{k}}^1 \ \text{MUST}^{\text{s}}\ \sigma_{\hat{k}}\end{array}}{\sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho_i^1 \ \text{MUST}^{\text{s}}\ \bigoplus_{k\in K} !\mathtt{l}_\mathtt{k}.\sigma_k}\ K \subseteq I;\ [\text{R-EXCH}]$$

Lemma 3.3.14 ensures that σ MUST $\text{UNF}(\rho_1)$. The hypothesis $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \rho_2$ and Lemma 3.1.9 imply that $\text{UNF}(\rho_1) \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \text{UNF}(\rho_2)$. The last fact and σ MUST $\text{UNF}(\rho_1)$ imply that σ MUST $\text{UNF}(\rho_2)$, so, in view of Lemma 3.3.14, there exists the derivation of $\text{UNF}(\rho_2)$ MUST^{s} σ. By hypothesis $\text{UNF}(\rho_2) \neq \mathbf{1}$, and so, as $\mathit{depth}(\text{UNF}(\rho_2)) + \mathit{depth}(\sigma) = 0$, the derivation of $\text{UNF}(\rho_2)$ MUST^{s} σ must be due to rule [R-EXCH],

$$\frac{\begin{array}{c}\vdots\\ \rho_1^2 \ \text{MUST}^{\text{s}}\ \sigma_1\end{array} \quad \ldots \quad \begin{array}{c}\vdots\\ \rho_{\hat{k}}^2 \ \text{MUST}^{\text{s}}\ \sigma_{\hat{k}}\end{array}}{\sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho_j^2 \ \text{MUST}^{\text{s}}\ \bigoplus_{k\in K} !\mathtt{l}_\mathtt{k}.\sigma_k}\ K \subseteq J;\ [\text{R-EXCH}]$$

The derivation above proves that $\text{UNF}(\rho_2) = \sum_{j\in J} ?\mathtt{l_j}.\rho_j^2$ for some set J such that $K \subseteq J$. We prove that $\rho_n^1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \rho_n^2$ for every $n \in (I \cup K) \cap J$. Fix an $n \in (I \cup K) \cap J$. Either $n \in K$ or $n \notin K$. For all the $n \in K$, the only hypothesis on σ_n is that ρ_n^1 MUST$^{\text{s}}$ σ_n. Since the derivation above proves that ρ_n^2 MUST$^{\text{s}}$ σ_n, Lemma 3.3.14 ensures that we have proven that if σ MUST ρ_n^1 then σ MUST ρ_n^2. If $n \notin K$, then ρ_n^1 is not usable; this implies that $\rho_n^1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \rho_n^2$. □

Example 6.2.8 shows that in the proof of the previous lemma the hypothesis of ρ_1 being usable is necessary.

Definition 6.2.10. [Syntactic MUST client pre-order]
Let $\mathcal{F}_{\precsim^{syn}_{\text{CLT}}} : \mathcal{P}(\mathsf{SC_{fo}}^2) \longrightarrow \mathcal{P}(\mathsf{SC_{fo}}^2)$ be defined so that $(\rho_1, \rho_2) \in \mathcal{F}_{\precsim^{syn}_{\text{CLT}}}(\mathcal{R})$ whenever if $\rho_1 \in \mathcal{U}^{\mathsf{SC_{fo}}}_{\text{CLT}}$ and $\text{UNF}(\rho_2) \neq \mathsf{1}$ one of the following is true:

(i) if $\text{UNF}(\rho_1) = !\mathtt{t}_1.\rho_1'$, then $\text{UNF}(\rho_2) = !\mathtt{t}_2.\rho_2'$ $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $\rho_1' \precsim^{syn}_{\text{CLT}} \rho_2'$

(ii) if $\text{UNF}(\rho_1) = ?\mathtt{t}_1.\rho_1'$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ then $\text{UNF}(\rho_2) = ?\mathtt{t}_2.\rho_2'$ and $\rho_1' \precsim^{syn}_{\text{CLT}} \rho_2'$

(iii) if $\text{UNF}(\rho_1) = (\sum_{i\in I} ?\mathtt{l_i}.\rho_i^1) + (\sum_{k\in K} ?\mathtt{l_k}.\rho_k^1)$ with $\rho_k^1 \in \mathcal{U}^{\mathsf{SC_{fo}}}_{\text{CLT}}$ for every $k \in K$, and $\rho_i^1 \notin \mathcal{U}^{\mathsf{SC_{fo}}}_{\text{CLT}}$ for every $i \in I$ then $\text{UNF}(\rho_2) = \sum_{j\in J} ?\mathtt{l_j}.\rho_j^2$ with $K \subseteq J$, and $\rho_n^1 \precsim^{syn}_{\text{CLT}} \rho_n^2$ for every $n \in (I \cup K) \cap J$

(iv) if $\text{UNF}(\rho_1) = \bigoplus_{i\in I} !\mathtt{l_i}.\rho_i^1$ then $\text{UNF}(\rho_2) = \bigoplus_{j\in J} !\mathtt{l_j}.\rho_j^2$ with $J \subseteq I$ and $\rho_j^1 \precsim^{syn}_{\text{CLT}} \rho_j^2$ for every $i \in J$

If $X \subseteq \mathcal{F}^{\precsim^{syn}_{\text{CLT}}}(X)$, then we say that X is a *syntactic client relation*. Lemma C.0.27 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\precsim^{syn}_{\text{CLT}}}(X)$; we call this solution the *client pre-order*, and we denote it $\precsim^{syn}_{\text{CLT}}$. That is $\precsim^{syn}_{\text{CLT}} = \nu X.\mathcal{F}^{\precsim^{syn}_{\text{CLT}}}(X)$. □

In Definition 6.2.10 we refer to the set $\mathcal{U}^{\mathsf{SC_{fo}}}_{\text{CLT}}$; since this set is not syntactically defined, this seems to hinder the relation $\precsim^{syn}_{\text{CLT}}$ from being syntactic-oriented.

Corollary 6.2.11. *The restricted* MUST *client pre-order is a co-inductive client pre-order. Formally,*

$$\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \subseteq \mathcal{F}_{\precsim^{syn}_{\text{CLT}}}(\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}})$$

Proof. It follows from Lemma 6.2.9. □

The converse of Corollary 6.2.11 is also true.

Proposition 6.2.12. [Alternative characterisation $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$]
The restricted MUST client pre-order and the syntactic must client pre-order coincide. Formally, $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} = \precsim^{syn}_{\text{CLT}}$.

Proof. We are required to prove two set inclusions,

a) $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \subseteq \precsim^{syn}_{\text{CLT}}$

b) $\precsim^{syn}_{\text{CLT}} \subseteq \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$

The first implication follows from Corollary 6.2.11, so we prove only the second implication. To this end, we prove a more general result: we show that every co-inductive syntactic server pre-order is contained in $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$.

Fix a relation $\mathcal{S}$ that is a co-inductive syntactic server pre-order. To prove that $\mathcal{S} \subseteq \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$, we have to show that if $\rho_1 \mathcal{S} \rho_2$ and σ MUST ρ_1, then σ MUST ρ_2. Let

$$\mathcal{R} = \{ (\rho_2, \sigma) \mid \sigma \text{ MUST } \rho_1, \rho_1 \mathcal{S} \rho_2 \}$$

it is enough to show that $\mathcal{R} \subseteq \text{MUST}^{-1}$. Thanks to Lemma 3.3.14, this is equivalent to showing that $\mathcal{R} \subseteq \text{MUST}^{\text{s}}$. We prove this fact.

Fix a pair $\rho_2 \; \mathcal{R} \; \sigma$; by construction of $\mathcal{R}$, we know that $\sigma(\text{MUST} \circ \mathcal{S})\rho_2$, so there exist a ρ_1 such that $\sigma \; \text{MUST} \; \rho_1$ and $\rho_1 \; \mathcal{S} \; \rho_2$.

Lemma 3.3.14 implies that there is a finite derivation of $\rho_1 \; \text{MUST}^{\text{s}} \; \sigma$. The argument to prove that $\rho_2 \; \text{MUST}^{\text{s}} \; \sigma$ is by rule induction on the derivation of

$$\frac{\vdots}{\rho_1 \; \text{MUST}^{\text{s}} \; \sigma} \tag{6.4}$$

In the base case the last rule used in the derivation in (6.4) is the axiom [A-UNIT]. It follows that $\rho_1 = \mathbf{1}$. As $\text{UNF}(\rho_1) = \mathbf{1}$, Definition 6.1.8 and $\rho_1 \; \mathcal{S} \; \rho_2$ ensure that $\text{UNF}(\rho_2) = \mathbf{1}$. Now we apply the axiom,

$$\frac{}{\text{UNF}(\rho_2) \; \text{MUST}^{\text{s}} \; \sigma} \; [\text{A-UNIT}]$$

If $\mathit{depth}(\rho_2) + \mathit{depth}(\rho_1) = 0$ then $\rho_2 = \text{UNF}(\rho_2)$, thus we have derived $\rho_2 \; \text{MUST}^{\text{s}} \; \sigma$.

If $\mathit{depth}(\rho_2) + \mathit{depth}(\rho_1) > 0$ then we extend the derivation by unfolding the session contracts,

$$\frac{\dfrac{}{\text{UNF}(\rho_2) \; \text{MUST}^{\text{s}} \; \sigma} \; [\text{A-UNIT}]}{\rho_2 \; \text{MUST}^{\text{s}} \; \sigma} \; \mathit{depth}(\rho_2) + \mathit{depth}(\rho_1) > 0; \; [\text{R-UNFOLD}]$$

In the inductive case we have to discuss five cases, depending on the last rule used in (6.4). We show the arguments for two cases, as the other cases can be proven by using similar arguments.

- If the last rule in (6.4) is [R-ALPHA], then the derivation is

$$\frac{\dfrac{\vdots}{\rho_1' \; \text{MUST}^{\text{s}} \; \sigma'}}{\alpha.\rho_1' \; \text{MUST}^{\text{s}} \; \overline{\alpha}.\sigma'} \; [\text{R-ALPHA}]$$

  It follows that $\rho_1 = \alpha.\rho_1'$ and that $\sigma = \overline{\alpha}.\sigma'$. Since $\text{UNF}(\rho_1) = \rho_1$, the fact that $\rho_1 \; \mathcal{S} \; \rho_2$ and Definition 6.1.8 imply that $\text{UNF}(\rho_2) = \beta.\rho_2'$ for some β such that $\overline{\alpha} \bowtie_c \beta$ and $\rho_1' \; \mathcal{S} \; \rho_2$. Since the derivation of $\rho_1' \; \text{MUST}^{\text{s}} \; \sigma'$ is shorter than (6.4), and $\sigma' \; \text{MUST} \; \rho_1' \; \mathcal{S} \; \rho_2'$, the inductive hypothesis ensures that $\rho_2' \; \text{MUST}^{\text{s}} \; \sigma'$. It follows that we can derive

$$\frac{\dfrac{\vdots}{\rho_2' \; \text{MUST}^{\text{s}} \; \sigma'}}{\beta.\rho_2' \; \text{MUST}^{\text{s}} \; \overline{\alpha}.\sigma'} \; [\text{R-ALPHA}]$$

  We have proven that $\text{UNF}(\rho_2) \; \text{MUST}^{\text{s}} \; \sigma$. If $\mathit{depth}(\rho_2) = 0$ then we have proven that $\rho_2 \; \text{MUST}^{\text{s}} \; \sigma$. If $\mathit{depth}(\rho_2) > 0$ then an application of rule [R-UNFOLD] lets us prove that $\text{UNF}(\rho_2) \; \text{MUST}^{\text{s}} \; \sigma$.

- If the last rule used in (6.4) is [R-UNFOLD], then the derivation is

$$\frac{\dfrac{\vdots}{\text{UNF}(\rho_1) \; \text{MUST}^{\text{s}} \; \text{UNF}(\sigma)}}{\rho_1 \; \text{MUST}^{\text{s}} \; \sigma} \; [\text{R-UNFOLD}]$$

  Since $\text{UNF}(\rho_1) \; \mathcal{S} \; \text{UNF}(\rho_1)$ and $\text{UNF}(\rho_1) \; \text{MUST}^{\text{s}} \; \text{UNF}(\sigma)$, it follows that $\text{UNF}(\rho_2) \; \mathcal{R} \; \text{UNF}(\sigma)$. The inductive hypothesis states that there exists a derivation of $\text{UNF}(\rho_2) \; \text{MUST}^{\text{s}} \; \text{UNF}(\sigma)$. If

$depth(\rho_2) + depth(\sigma) = 0$ then we have proven that $\rho_2 \text{ MUST}^{\text{s}} \sigma$. Otherwise we derive

$$\frac{\dfrac{\vdots}{\text{UNF}(\rho_2) \text{ MUST}^{\text{s}} \text{ UNF}(\sigma)}}{\rho_2 \text{ MUST}^{\text{s}} \sigma} \; depth(\rho_2) + depth(\sigma) > 0 \; [\text{R-UNFOLD}]$$

We have proven that for every $\rho_2 \; \mathcal{R} \; \sigma$ we can derive $\rho_2 \text{ MUST}^{\text{s}} \sigma$. In view of the definition of $\mathcal{R}$, this proves that if $\rho_1 \precsim^{syn}_{\text{CLT}} rho_2$ then $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \rho_2$. □

In this section we have unravelled the behavioural characterisation of the pre-order $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$. This characterisation turns out to to be co-inductive and syntax directed, as the characterisation of $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{SVR}}}$.

In the next section we will use the characterisation of the restricted MUST server pre-order and the restricted MUST client pre-order to define a fully abstract model of the sub-typing on first-order session types.

6.3 A behavioural model of first-order sub-typing

As we have shown, the difficulty is to find a natural pre-order on session contracts which accurately reflects the sub-typing relation on session contracts. There are two obvious candidates, the restricted MUST server pre-order and the restricted MUST client pre-order on session contracts. The difficulty lies in the interpretation of END.

Definition 6.3.1. [Session contract pre-order]
For $\sigma_1, \sigma_2 \in \mathsf{SC}_{\text{fo}}$ let $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{=_{\text{P2P}}} \sigma_2$ whenever $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{SVR}}} \sigma_2$ and $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}} \sigma_2$. □

Example 6.3.2. It is instructive to see the behaviour of 1, the image of END under $\mathcal{M}$, relative to this combined pre-order. First suppose $\sigma \sqsubseteq^{\mathsf{fo}}_{=_{\text{P2P}}} 1$ for some session contract σ. This implies $\sigma \sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{SVR}}} 1$ and therefore, as we have shown in Lemma 6.1.5, σ must be a bottom element relative to $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{SVR}}}$, so $\text{UNF}(\sigma)$ must be 1. A similar argument, using the pre-order $\sqsubseteq^{\mathsf{fo}}_{\simeq_{\text{CLT}}}$ ensures that if $1 \sqsubseteq^{\mathsf{fo}}_{=_{\text{P2P}}} \sigma$ then $\text{UNF}(\sigma)$ must be 1.

In other words modulo unfolding the only session contract related to 1 via $\sqsubseteq^{\mathsf{fo}}_{=_{\text{P2P}}}$ is 1 itself. □

Proposition 6.3.3. [Completeness]
For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\text{fo}}$, $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{=_{\text{P2P}}} \sigma_2$ implies $\mathcal{M}^{-1}(\sigma_1) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \mathcal{M}^{-1}(\sigma_2)$.

Proof. Let $\mathcal{R}$ be the relation over session types defined as follows,

$$\mathcal{R} = \{(\mathcal{M}^{-1}(\sigma_1), \mathcal{M}^{-1}(\sigma_2)) \mid \sigma_1 \precsim^{syn}_{\text{SVR}} \sigma_2, \sigma_1 \precsim^{syn}_{\text{CLT}} \sigma_2\}$$

If we prove that $\mathcal{R}$ is a type simulation (see Definition 2.1.13), then the result will follow because of Proposition 6.1.10 and Proposition 6.2.12.

To show that $\mathcal{R}$ is a type simulation we are required to prove the set inclusion $\mathcal{R} \subseteq \mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}(\mathcal{R})$.

Fix a pair $S_1 \; \mathcal{R} \; S_2$. There exist a σ_1 and a σ_2 such that $S_1 = \mathcal{M}^{-1}(\sigma_1)$, $S_2 = \mathcal{M}^{-1}(\sigma_1)$, and $\sigma_1 \precsim^{syn}_{\text{SVR}} \sigma_2$ and $\sigma_1 \precsim^{syn}_{\text{CLT}} \sigma_2$.

The proof proceeds by a case analysis on the depth of the types S_1 and S_2, and then by case analysis on S_1.

Suppose that $depth(S_1) + depth(S_2) > 0$. Since $\sigma_1 \precsim^{syn}_{\text{SVR}} \sigma_2$ and $\sigma_1 \precsim^{syn}_{\text{CLT}} \sigma_2$, Lemma 6.1.4 and Lemma 6.2.3 ensure that $\text{UNF}(\sigma_1) \precsim^{syn}_{\text{SVR}} \text{UNF}(\sigma_2)$ and $\text{UNF}(\sigma_1) \precsim^{syn}_{\text{CLT}} \text{UNF}(\sigma_2)$, and so

$$\mathcal{M}^{-1}(\text{UNF}(\sigma_1)) \; \mathcal{R} \; \mathcal{M}^{-1}(\text{UNF}(\sigma_2))$$

point (iii) of Lemma 2.3.12 implies the following fact,

$$\textsc{unf}(\mathcal{M}^{-1}(\sigma_1))\ \mathcal{R}\ \textsc{unf}(\mathcal{M}^{-1}(\sigma_2))$$

so $\textsc{unf}(S_1)\ \mathcal{R}\ \textsc{unf}(S_2)$. Now we can apply [R-Unfold],

$$\frac{\textsc{unf}(S_1) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \textsc{unf}(S_2)}{S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2}\ \mathit{depth}(S_1) + \mathit{depth}(S_2) > 0\ [\textsc{r-Unfold}]$$

Suppose now that $\mathit{depth}(S_1) + \mathit{depth}(S_2) = 0$; then the unfoldings of S_1 and S_2 are the types themselves. Now we reason by case analysis on S_1; that is $\mathcal{M}^{-1}(\sigma_1)$.

- Suppose $\mathcal{M}^{-1}(\sigma_1) = \textsc{end}$. According to Definition 2.1.13 we have to show that $\mathcal{M}^{-1}(\sigma_2) = \textsc{end}$. The definition of $\mathcal{M}$ implies that $\sigma_1 = \mathtt{1}$; moreover In Example 6.3.2 above we have already reasoned that $\textsc{unf}(\sigma_2)$ must be $\mathtt{1}$, and so $\mathcal{M}^{-1}(\sigma_2) = \textsc{end}$.

- Suppose $\mathcal{M}^{-1}(\sigma_1) = ![\,\mathtt{t}_1\,]; S_1'$. We are required to prove that

$$\mathcal{M}^{-1}(\sigma_2) = ![\,\mathtt{t}_2\,]; S_2', \tag{6.5}$$

  for some $\mathtt{t}_2$ and S_2' such that $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $(\mathcal{M}(S_1'), \mathcal{M}(S_2')) \in \precsim^{syn}_{\text{SVR}} \cap \precsim^{syn}_{\text{CLT}}$.

  As $\sigma_1 \precsim^{syn}_{\text{SVR}} \sigma_2$, Definition 6.1.8 implies that $\sigma_2 = !\mathtt{t}_2.\sigma_2'$ for some $\mathtt{t}_2$ such that $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and some σ_2' such that $\mathcal{M}(S_1) \precsim^{syn}_{\text{SVR}} \sigma_2'$. This ensures that (6.5) above is satisfied. By the definition of S_2 we also have the requirement $\mathcal{M}(S_1) \precsim^{syn}_{\text{SVR}} \sigma_2'$.

  It remains to show $\mathcal{M}(S_1) \precsim^{syn}_{\text{CLT}} \mathcal{M}(S_2)$. But this follows from $\sigma_1 \precsim^{syn}_{\text{CLT}} \sigma_2$, and Definition 6.2.10.

- A case that requires particular care is when $S_1 = \&\langle\, \mathtt{l}_1\colon S_1^1,\ \ldots\ , \mathtt{l}_{\mathtt{i}}\colon S_m^1 \,\rangle$. Definition 2.1.13 requires us to prove that three things

  1) $S_2 = \&\langle\, \mathtt{l}_1\colon S_1^2, \ldots, \mathtt{l}_{\mathtt{i}}\colon S_n^2 \,\rangle$
  2) $m \leq n$
  3) for every $1 \leq i \leq m$, $S_i^1\ \mathcal{R}\ S_i^2$

  The construction of $\mathcal{R}$ ensures that there are two session contracts σ_1 and σ_2 such that $\sigma_1 \precsim^{syn}_{\text{SVR}} \sigma_2$ and $\sigma_1 \precsim^{syn}_{\text{CLT}} \sigma_2$; and such that $S_1 = \mathcal{M}(\sigma_1)$ and $S_2 = \mathcal{M}(\sigma_2)$. We know that $\sigma_1 = \sum_{i\in I} ?\mathtt{l}_{\mathtt{i}}.\sigma_i^1$, with $\mathcal{M}^{-1}(\mathcal{S}_i^1) = \sigma_1^1$ for every $i \in [1;n]$. Rule [R-Branch] in Figure 6.1 Definition 6.1.8 ensures that $\sigma_2 = \sum_{j\in J} ?\mathtt{l}_{\mathtt{j}}.\sigma_j^1$ for some set J such that $[1;n] \subseteq J$ and that for every $i \in [1;m]$, $\sigma_i^1 \precsim^{syn}_{\text{SVR}} \sigma_i^2$.

  The set inclusion $[1;m] \subseteq J$ implies that $J = [1;n]$ for some $n \in N$ such that $m \leq n$. Since $S_2 = \mathcal{M}(\sigma_2)$, $\sigma_2 = \sum_{j\in J} ?\mathtt{l}_{\mathtt{j}}.\sigma_j^1$, and $J = [1;n]$, we can prove 1) and 2) above: $S_2 = \&\langle\, \mathtt{l}_1\colon S_1^2, \ldots \mathtt{l}_{\mathtt{i}}\colon S_n^2 \,\rangle$ for some n such that $m \leq n$.

  We still have to prove 3). The equality in Eq. (6.5) ensures that for every $j \in [1;n]$, $S_j^2 = \mathcal{M}\sigma_j^2$; that is $\mathcal{M}^{-1}(\mathcal{S}_i^2) = \sigma_j^2$. We have already proven that for every $i \in [1;m]$, $\mathcal{M}^{-1}(\mathcal{S}_i^1) = \sigma_1^1$. We have to show that $S_i^1\ \mathcal{R}\ S_i^2$ for every $i \in [1;m]$. Fix an $i \in [1;m]$. To prove that $S_i^1\ \mathcal{R}\ S_i^2$ we have to show two facts, namely that $\mathcal{M}^{-1}(S_i^1) \precsim^{syn}_{\text{SVR}} \mathcal{M}^{-1}(S_i^2)$ and that $\mathcal{M}^{-1}(S_i^1) \precsim^{syn}_{\text{CLT}} \mathcal{M}^{-1}(S_i^2)$. Thanks to the equalities that we have already proven, we have to show that $\sigma_i^1 \precsim^{syn}_{\text{SVR}} \sigma_i^2$ and that $\sigma_i^1 \precsim^{syn}_{\text{CLT}} \sigma_i^2$. We have already shown the first fact; so we have to explain why $\sigma_i^1 \precsim^{syn}_{\text{CLT}} \sigma_i^2$. As $\sigma_1 \precsim^{syn}_{\text{CLT}} \sigma_2$, Point (iii) of Definition 6.2.10 and $I \subseteq J$ imply that for every $i \in I$, $\sigma_1^1 \precsim^{syn}_{\text{CLT}} \sigma_2^1$.

The proof for the remaining cases is similar to the argument already shown, and left to the reader. □

Theorem 6.3.4. [Full abstraction]
For every $T_1, T_2 \in \mathsf{ST}_{\mathrm{fo}}$, $T_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} T_2$ if and only if $\mathcal{M}(T_1) \sqsubseteq^{\mathsf{fo}}_{\mathrm{P2P}} \mathcal{M}(T_2)$.

Proof. Thanks to Proposition 6.3.3, it is sufficient to prove that $T_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} T_2$ implies $\mathcal{M}(T_1) \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{SVR}}} \mathcal{M}(T_2)$ and $\mathcal{M}(T_1) \sqsubseteq^{\mathsf{fo}}_{\simeq_{\mathrm{CLT}}} \mathcal{M}(T_2)$. As an example we outline the proof of the former. Because of Proposition 6.1.10 it is sufficient to show that the relation $\mathcal{R}$ given by

$$\mathcal{R} = \{\, (\sigma_1,\, \sigma_2) \mid T_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} T_2,\ \mathcal{M}(T_1) = \sigma_1,\ \mathcal{M}(T_2) = \sigma_2 \,\}$$

is a syntactic server pre-order, that is $\mathcal{R} \subseteq \mathcal{F}^{\preccurlyeq^{syn}_{\mathrm{SVR}}}(\mathcal{R})$, where $\mathcal{F}^{\preccurlyeq^{syn}_{\mathrm{SVR}}}$ is given in Definition 6.1.8.

Suppose $\sigma_1\ \mathcal{R}\ \sigma_2$; we have to prove that the pair $(\sigma_1,\, \sigma_2)$ is in $\mathcal{F}^{\preccurlyeq^{syn}_{\mathrm{SVR}}}(\mathcal{R})$. By definition there exist T_1 and T_2 such that $\mathcal{M}(T_1) = \sigma_1$, that $\mathcal{M}(T_2) = \sigma_2$, and that $T_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} T_2$.

We do first a case analysis on the depth of σ_1 and σ_2, and then reason by case analysis on σ_1.

Suppose that $\mathit{depth}(\sigma_1) + \mathit{depth}(\sigma_2) > 0$; to prove that $(\sigma_1,\, \sigma_2) \in \mathcal{F}^{\preccurlyeq^{syn}_{\mathrm{SVR}}}(\mathcal{R})$ it is enough to show that $\textsc{unf}(\sigma_1)\ \mathcal{R}\ \textsc{unf}(\sigma_2)$. The fact that $T_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} T_2$ and Lemma 2.1.16 imply that $\textsc{unf}(T_1) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \textsc{unf}(T_2)$. The construction of $\mathcal{R}$ guarantees that

$$\mathcal{M}(\textsc{unf}(T_1))\ \mathcal{R}\ \mathcal{M}(\textsc{unf}(T_2))$$

Point (iii) of Lemma 2.3.12 ensures that we can commute the unfolding with the interpretation $\mathcal{M}$,

$$\textsc{unf}(\mathcal{M}(T_1))\ \mathcal{R}\ \textsc{unf}(\mathcal{M}(T_2))$$

and so $\textsc{unf}(\sigma_1)\ \mathcal{R}\ \textsc{unf}(\sigma_2)$. Now we apply [R-Unfold],

$$\frac{\textsc{unf}(\sigma_1) \preccurlyeq^{syn}_{\mathrm{SVR}} \textsc{unf}(\sigma_2)}{\sigma_1 \preccurlyeq^{syn}_{\mathrm{SVR}} \sigma_2}\ \mathit{depth}(\sigma_1) + \mathit{depth}(\sigma_2) > 0\ [\textsc{R-Unfold}]$$

Suppose now that $\mathit{depth}(\sigma_1) + \mathit{depth}(\sigma_2) = 0$; this implies that $\textsc{unf}(\sigma_1) = \sigma_1$ and that $\textsc{unf}(\sigma_2) = \sigma_2$. We proceed reasoning by case analysis on σ_1.

- If $\sigma_1 = \mathsf{1}$, then Definition 6.1.8 ensures that $(\sigma_1,\, \sigma_2)$ is in $\mathcal{F}^{\preccurlyeq^{syn}_{\mathrm{SVR}}}(\mathcal{R})$.
- If $\sigma_1 = ?\mathtt{t}_1.\sigma_1'$ we have to show that
$$\sigma_2 = ?\mathtt{t}_2.\sigma_2'$$
with $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and $\sigma_1'\ \mathcal{R}\ \sigma_2'$.

  The fact that $\mathcal{M}^{-1}(\sigma_1) \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \mathcal{M}^{-1}(\sigma_2)$ let us use Definition 2.1.13 to deduce that $\mathcal{M}^{-1}(\sigma_2) = ?[\,t_2\,];.\mathcal{M}^{-1}(\sigma_2')$ for some $\mathtt{t}_2$ such that $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and some $\mathcal{M}^{-1}(\sigma_2')$ such that $\mathcal{M}^{-1}(\sigma_1') \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \mathcal{M}^{-1}(\sigma_2')$. From the last inequality and the definition of $\mathcal{R}$ it follows that $\sigma_1'\ \mathcal{R}\ \sigma_2'$. Since we have proven the conditions on the input actions $\mathtt{t}_1$, $\mathtt{t}_2$ and on the continuations σ_1', σ_2' we have left only to show that the structure of σ_2 is the required one; this follows from another application of part (iii) of Lemma 2.3.12.

The other cases are analogous and left to the reader. □

Corollary 6.3.5. *If $\bowtie_c$ is decidable, then the relation $\sqsubseteq^{\mathsf{fo}}_{\mathrm{P2P}}$ is decidable.*

Proof. To begin with, note that $\mathcal{M}$ is defined by structural induction, and in a similar fashion we can define its inverse $\mathcal{M}^{-1}$; so $\mathcal{M}^{-1}$ is decidable. The corollary then follows from Corollary 2 of [Gay and Hole, 2005], which ensures that the relation $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ is decidable, and our Theorem 6.3.4, whereby we can prove the isomorphism $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \cong \sqsubseteq^{\mathsf{fo}}_{\mathrm{P2P}}$. □

6.3.1 Examples and applications

In this subsection we give a series of examples in order to discuss the results we obtained. The first two example are of theoretical nature, whereas the last one shows an application.

Example 6.3.6. [Type simulations and the weak simulation relation]
At this stage, a natural question arises, which concerns the relationship between type simulations and weak simulations [Milner, 1999]. Assume the standard definition of the weak simulation [Milner, 1999]; we use the symbol $\lesssim$ to denote the greatest weak simulation relation.

We begin by showing that, even though two session types are in a co-inductive types simulation, their images through $\mathcal{M}$ need not be in a weak simulation. Consider the relation

$$\mathcal{R} = \{ (\oplus\langle \mathtt{l}_1\colon \text{END},\ \mathtt{l}_2\colon \text{END}\rangle, (\oplus\langle \mathtt{l}_1\colon \text{END}\rangle), (\text{END}, \text{END}) \}$$

The standard co-inductive proof technique let one prove that the relation $\mathcal{R}$ is a type simulation. On the other hand, the definition of $\mathcal{M}$ implies that

$$\mathcal{M}(\oplus\langle \mathtt{l}_1\colon \text{END},\ \mathtt{l}_2\colon \text{END}\rangle) = !\mathtt{l}_1.1 \oplus !\mathtt{l}_2.1$$
$$\mathcal{M}(\oplus\langle \mathtt{l}_1\colon \text{END}\rangle) = !\mathtt{l}_1.1$$

Then $\mathcal{M}(\oplus\langle \mathtt{l}_1\colon \text{END},\ \mathtt{l}_2\colon \text{END}\rangle) \not\lesssim \mathcal{M}(\oplus\langle \mathtt{l}_1\colon \text{END}\rangle)$ because $!\mathtt{l}_1.1 \oplus !\mathtt{l}_2.1 \xrightarrow{\tau}\xrightarrow{\mathtt{l}_2}$, while $!\mathtt{l}_1.1 \overset{\mathtt{l}_2}{\not\longrightarrow}$. We have proven that $S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2$ does not imply $\mathcal{M}(S_1) \lesssim \mathcal{M}(S_2)$.

Looking at the foregoing argument, one might be tempted to reason that if $S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2$ then $\mathcal{M}(S_2) \lesssim \mathcal{M}(S_1)$. We prove that this is not the case. We can prove that

$$?\mathtt{l}_1.1 \sqsubseteq^{\mathsf{fo}}_{\text{P2P}} ?\mathtt{l}_2.1 + ?\mathtt{l}_1.1$$

An application of $\mathcal{M}^{-1}$ gives us:

$$\mathcal{M}^{-1}(?\mathtt{l}_1.1) = \&\langle \mathtt{l}_1\colon \text{END}\rangle$$
$$\mathcal{M}^{-1}(?\mathtt{l}_2.1 + ?\mathtt{l}_1.1) = \&\langle \mathtt{l}_1\colon \text{END},\ \mathtt{l}_2\colon \text{END}\rangle$$

A look at the definition of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$, Definition 2.1.13, lets one prove that for every type simulation $\mathcal{R}$

$$\&\langle \mathtt{l}_1\colon \text{END},\ \mathtt{l}_2\colon \text{END}\rangle \not\mathcal{R}\ \&\langle \mathtt{l}_1\colon \text{END}\rangle$$

and, therefore,

$$\&\langle \mathtt{l}_1\colon \text{END},\ \mathtt{l}_2\colon \text{END}\rangle \not\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \&\langle \mathtt{l}_1\colon \text{END}\rangle$$

□

Example 6.3.7. [e-vote, revisited]
In this example we use Theorem 6.3.4 in conjunction with Theorem 2 of [Gay and Hole, 2005], in order to show how the set based pre-order $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ can be used to guarantee that a process P_a can be safely replaced by a suitable process P_b.

Consider two contracts BallotA and BallotB such that BallotA $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ BallotB. Let $BallotA = \mathcal{M}^{-1}(\text{BallotA})$ and $BallotB = \mathcal{M}^{-1}(\text{BallotB})$. From Theorem 6.3.4 it follows that

$$BallotA \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} BallotB \tag{6.6}$$

Let $\perp_c$ denote the *coinductive duality relation* defined as in Definition 9 of [Gay and Hole, 2005]. Suppose now that BltSrvA(x^+), BltSrvB(x^+) and Voter(x^-) are pi calculus processes (as in [Gay

and Hole, 2005]) such that

$$\begin{array}{rcl} \{x^+ : BallotA\} & \vdash & \textsc{BltSrvA}(x^+), \\ \{x^+ : BallotB\} & \vdash & \textsc{BltSrvB}(x^+), \\ \{x^- : Voter\} & \vdash & \textsc{Voter}(x^-) \end{array}$$

for some session type $Voter$ such that $Voter \perp_c BallotA$. By means of the typing rules of [Gay and Hole, 2005], it is possible to derive

$$\dfrac{\dfrac{\dfrac{\vdots}{\{x^+ : BallotA\} \vdash \textsc{BltSrvA}(x^+)} \quad \dfrac{\vdots}{\{x^- : Voter\} \vdash \textsc{Voter}(x^-)}}{\{x^+ : BallotA\}, x^- : Voter \vdash \textsc{BltSrvA}(x^+) \mid \textsc{Voter}(x^-)}\ [\text{T-Par}]}{\vdash (\nu x : BallotA)\ \textsc{BltSrvA}(x^+) \mid \textsc{Voter}(x^-)}\ [\text{T-NewS}]$$

Then (6.6) above and Theorem 2 of [Gay and Hole, 2005] can be used to guarantee that if process $\textsc{BltSrvB}(x^+)$ is used in place of process $\textsc{BltSrvA}(x^+)$, then no communication error will happen along the channel x.

One can use non-recursive versions of the contracts seen in Examples 2.3.5 and 6.1.3 to obtain contracts that satisfy the assumptions above:

$$\begin{array}{rcl} \text{BallotA} & = & ?\texttt{Login}.(\,!\texttt{Wrong}.\,1 \oplus !\texttt{Ok}.(\,?\texttt{VoteA}.\,1 + ?\texttt{VoteB}.\,1\,)\,) \\ \text{BallotB} & = & ?\texttt{Login}.(\,!\texttt{Wrong}.\,1 \oplus \\ & & \qquad !\texttt{Ok}.(\,?\texttt{VoteA}.\,1 + ?\texttt{VoteB}.\,1 + ?\texttt{VoteC}.\,1 + ?\texttt{VoteD}.\,1\,)\,) \\ Voter & = & \mathcal{M}^{-1}(!\texttt{Login}.(\,?\texttt{Wrong}.\,1 + ?\texttt{Ok}.(\,!\texttt{VoteA}.\,1 \oplus !\texttt{VoteB}.\,1\,)\,)) \end{array}$$

□

Example 6.3.8. [Protocol conformance]
As already remarked, the language for contracts is a sublanguage of CCS without τ's [De Nicola and Hennessy, 1987], and consequently contracts are suitable for specifying communication protocols.

Assume a protocol Pr to be specified by a contract σ, and let Q be a process (in the sense of [Gay and Hole, 2005]), which is well-typed under the environment Γ. Assume also that $\Gamma(x) = S$ for some channel x.

We want to answer the following question:

(Q) "Does the session type S conform to the protocol specification σ?"

Clearly, as long as the notion of conformance is not well defined, it is not possible to give an answer (at least not a trutworthy one).

In light of Theorem 6.3.4, we propose the following definition of conformance. Assume the standard definition of weak bisimilarity equivalence [Milner, 1999]; we denote this relation $\approx$. We say that a session type S conforms to a protocol specification σ if and only if $\mathcal{M}(S) \approx \sigma$.

To answer the question (Q) now one has only to prove that $\mathcal{M}(S) \approx \sigma$ or to show a counter example to this statement.

For example, if we had given a specification of the protocol POP3 [Rose, 1988] with a contract σ, then we would have been able to check whether the session type POP3 of [Gay et al., 2003] conforms to σ.

In order for the notion of conformance we have given to be of some practical consequence, one last thing has to be ascertained. We have to study whether weak bisimilarity equivalence, when restricted to session contracts, is *decidable*. We leave this as an open problem worth further investigation. □

6.4 Revisiting the restricted server pre-order

In this section we investigate the restricted pre-order for server that arise by using the compliance relation. The result of our study is that the restricted pre-orders for servers coincide (see Corollary 6.4.8).

Definition 6.4.1. [Restricted compliance server pre-order]
For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\mathrm{fo}}$, we write $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$ if and only if whenever for every $\rho \in \mathsf{SC}_{\mathrm{fo}}$, $\rho \dashv \sigma_1$ implies that $\rho \dashv \sigma_2$. We call the relation denoted by the symbol $\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}$ the restricted compliance server pre-order. □

When comparing session contracts relative to this pre-order it will be convenient to work modulo unfolding, which is possible because of the following result:

Lemma 6.4.2. For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\mathrm{fo}}$, $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$ if and only if $\mathrm{UNF}(\sigma_2) \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \mathrm{UNF}(\sigma_2)$.

Proof. Follows from Corollary 3.2.9 and 3.2.10. □

The set based relation $\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}$ is contained in $\precsim^{syn}_{\mathrm{SVR}}$; this will follow if we can show the former satisfies the defining properties of the latter.

Lemma 6.4.3. Let $\sigma_1, \sigma_2 \in \mathsf{SC}_{\mathrm{fo}}$, $\sigma_1 = \mathrm{UNF}(\sigma_1)$, $\sigma_2 = \mathrm{UNF}(\sigma_2)$ and $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$. Then

(i) if $\mathrm{UNF}(\sigma_1) = !\mathtt{t}_1.\sigma_1'$ then $\mathrm{UNF}(\sigma_2) = !\mathtt{t}_2.\sigma_2'$, $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $\sigma_1' \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2'$

(ii) if $\mathrm{UNF}(\sigma_1) = ?\mathtt{t}_1.\sigma_1'$ then $\mathrm{UNF}(\sigma_2) = ?\mathtt{t}_2.\sigma_2'$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and $\sigma_1' \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2'$

(iii) if $\mathrm{UNF}(\sigma_1) = \sum_{i\in I} ?\mathtt{l}_i.\sigma_i^1$ then $\mathrm{UNF}(\sigma_2) = \sum_{j\in J} ?\mathtt{l}_j.\sigma_j^2$, with $I \subseteq J$ and $\sigma_i^1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_i^2$

(iv) if $\mathrm{UNF}(\sigma_1) = \bigoplus_{i\in I} !\mathtt{l}_i.\sigma_i^1$ then $\mathrm{UNF}(\sigma_2) = \bigoplus_{j\in J} !\mathtt{l}_j.\sigma_j^2$, with $J \subseteq I$ and $\sigma_j^1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_j^2$

Proof. The proof is by case analysis on the structure of σ_1 and depends greatly on the restricted syntax of session contracts. We give the details of the first case; the others are analogous.

Suppose $\sigma_1 = !\mathtt{t}_1.\sigma_1'$. Then $?\mathtt{t}_1.\mathtt{1} \dashv \sigma_1$ and because $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$ it follows that $?\mathtt{t}_1.\mathtt{1} \dashv \sigma_2$. Since $?\mathtt{t}_1.\mathtt{1}$ is stable, σ_2 has to engage in an action $!\mathtt{t}_2$ such that $?\mathtt{t}_1 \bowtie_c !\mathtt{t}_2$. It follows $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$. In reason of the syntax and the the hypothesis $\sigma_2 = \mathrm{UNF}(\sigma_2)$, the equality $\sigma_2 = !\mathtt{t}_2.\sigma_2'$ must hold.

We also have to prove that $\sigma_1' \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2'$. Pick a session contract ρ such that $\rho \dashv \sigma_1'$. Clearly $?\mathtt{t}_1.\rho \dashv \sigma_1$, and thus $?\mathtt{t}_1.\rho \dashv \sigma_2$. Since $?\mathtt{t}_1 \bowtie_c !\mathtt{t}_2$, we apply rule [P-Synch] to infer $?\mathtt{t}_1.\rho \parallel \sigma_2 \xrightarrow{\tau} \rho \parallel \sigma_2'$. From the definition of compliance it follows that $\rho \dashv \sigma_2'$. □

Corollary 6.4.4. *The relation $\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}$ is a co-inductive syntactic server pre-order.*

Proof. We prove that $\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}$ is a prefixed point of $\mathcal{F}^{\precsim^{syn}_{\mathrm{SVR}}}$ of Definition 6.1.8, that is $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$ implies $(\sigma_1, \sigma_2) \in \mathcal{F}^{\precsim^{syn}_{\mathrm{SVR}}}(\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}})$. Suppose $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$. Then by Lemma 6.4.2 it follows that $\mathrm{UNF}(\sigma_1) \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \mathrm{UNF}(\sigma_2)$. Now if $\mathrm{UNF}(\sigma_1) = \mathtt{1}$ by definition $(\sigma_1, \sigma_2) \in \mathcal{F}^{\precsim^{syn}_{\mathrm{SVR}}}(\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}})$. Otherwise we can apply Lemma 6.4.3 to the pair $\mathrm{UNF}(\sigma_1)$, $\mathrm{UNF}(\sigma_2)$. This provides the required information to satisfy the requirements (ii) to (v) in Definition 6.1.8, thereby ensuring that $(\sigma_1, \sigma_2) \in \mathcal{F}^{\precsim^{syn}_{\mathrm{SVR}}}(\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}})$. □

A brief comparison between the properties of $\preceq_{\mathrm{SVR}}$ and $\precsim^{syn}_{\mathrm{SVR}}$ is in order. In Example 6.1.2 we have seen that $\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}$ does not satisfy point (1b) of Definition 5.1.7. This implies that the property of $\precsim_{\mathrm{SVR}}$ proven in Lemma 5.1.14 is not enjoyed by $\precsim^{syn}_{\mathrm{SVR}}$.

Example 6.4.5. We prove that $\sigma_1 \precsim^{syn}_{\mathrm{SVR}} \sigma_2$ and $\sigma_2 \xrightarrow{\alpha}$ do not imply that $(\sigma_1 \text{ AFTER } \alpha) \neq \emptyset$. Let $\sigma_1 = ?\mathtt{latte}.\mathtt{1}$ and $\sigma_2 = ?\mathtt{moka}.\mathtt{1} + ?\mathtt{latte}.\mathtt{1}$, and $\mathcal{R} = \{(\sigma_1, \sigma_2), (\mathtt{1}, \mathtt{1})\}$. Since the relation $\mathcal{R}$ is a co-inductive syntactic server pre-order, $\sigma_1 \precsim^{syn}_{\mathrm{SVR}} \sigma_2$. Now observe that $\sigma_2 \xrightarrow{?\mathtt{moka}} \mathtt{1}$; however the set $(\sigma_1 \text{ AFTER} ?\mathtt{moka})$ is empty. □

On the other hand we have the analogous of Lemma 5.1.13.

Lemma 6.4.6. Let $\mathcal{R}$ be a co-inductive syntactic server relation and let $\sigma_1 \mathcal{R} \sigma_2$. If $\sigma_2 \xrightarrow{\tau} \sigma_2'$ then $\sigma_1 \mathcal{R} \sigma_2'$.

Proof. First note that from Definition 6.1.8 it follows that

$$\text{UNF}(\sigma_1) \mathcal{R} \text{UNF}(\sigma_2) \tag{6.7}$$

There are two different cases to be discussed, depending on the unfolding of σ_2 being σ_2 itself or not.

(a) If $\text{UNF}(\sigma_2) \neq \sigma_2$ then σ_2 has a top-most recursion, and therefore we can prove that $\text{UNF}(\sigma_2) = \text{UNF}(\sigma_2')$. This equality and (6.7) above imply that $\text{UNF}(\sigma_1) \mathcal{R} \text{UNF}(\sigma_2')$, which in turn means that $\sigma_1 \mathcal{R} \sigma_2'$.

(b) If $\text{UNF}(\sigma_2) = \sigma_2$ then σ_2 must be an internal sum, say $\sigma_2 = \bigoplus_{i\in I} !1_{\mathtt{i}}.\sigma_i^2$, because σ_2 can perform a silent move and can not unfold.

Since $\sigma_2 = \bigoplus_{i\in I} !1_{\mathtt{i}}.\sigma_i^2$, the term σ_2' is the internal sum $\bigoplus_{k\in K} !1_{\mathtt{k}}.\sigma_k^2$, for some $K \subseteq I$. From Definition 6.1.8 it follows that $\text{UNF}(\sigma_1) = \bigoplus_{j\in J} !1_{\mathtt{j}}.\sigma_j^2$ with $I \subseteq J$. Since $\text{UNF}(\sigma_2') = \sigma_2'$ and $K \subseteq I \subseteq J$ one can prove that $\text{UNF}(\sigma_1) \mathcal{R} \text{UNF}(\sigma_2')$, and thus $\sigma_1 \mathcal{R} \sigma_2$.

□

The main result of this section is the following proposition.

Proposition 6.4.7. [Co-inductive characterisation $\sqsubseteq_{\text{SVR}}^{\mathsf{fo}}$]
For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\text{fo}}$, $\sigma_1 \sqsubseteq_{\text{SVR}}^{\mathsf{fo}} \sigma_2$ if and only if $\sigma_1 \precsim_{\text{SVR}}^{syn} \sigma_2$.

Proof. In view of Corollary 6.4.4 we have to prove only the inclusion $\precsim_{\text{SVR}}^{syn} \subseteq \sqsubseteq_{\text{CLT}}^{\mathsf{fo}}$. It is enough to show that the relation

$$\mathcal{R}' = \{\, (\rho,\, \sigma_2) \mid \rho \dashv \sigma,\ \sigma_1 \precsim_{\text{SVR}}^{syn} \sigma_2,\ \text{for some } \sigma_1 \in \mathsf{SC}_{\text{fo}} \,\}$$

is a co-inductive compliance. Thanks to Lemma 3.3.10 it is enough to prove that the following relation is a syntactic compliance,

$$\mathcal{R} = \{\, (\rho,\, \sigma_2) \mid \rho \dashv^s \sigma,\ \sigma_1 \precsim_{\text{SVR}}^{syn} \sigma_2,\ \text{for some } \sigma_1 \in \mathsf{SC}_{\text{fo}} \,\}$$

Let $\rho \mathcal{R} \sigma$; by definition of $\mathcal{R}$ there exists a σ_1

$$\rho \dashv^s \sigma_1, \qquad \sigma_1 \precsim_{\text{SVR}}^{syn} \sigma_2$$

We have to explain why $(\sigma_2,\, \rho) \in \mathcal{F}^{\dashv^s}(\mathcal{R}, \mathcal{T})$. That is, we have to apply one of the rules in Figure 3.2 to derive

$$\frac{\cdots}{\rho \dashv^s \sigma_2}$$

Lemma 6.1.4 and Lemma 3.3.3 guarantee the follows facts

$$\text{UNF}(\sigma_1) \precsim_{\text{SVR}}^{syn} \text{UNF}(\sigma_2), \quad \text{UNF}(\sigma_1) \dashv^s \text{UNF}(\rho)$$

We first check the depths of σ_2 and ρ. If $\mathit{depth}(\sigma_2) + \mathit{depth}(\rho) > 0$, then It follows that $\text{UNF}(\sigma_2) \mathcal{R}$ $\text{UNF}(\rho)$. We apply [R-Unfold],

$$\frac{\text{UNF}(\sigma_2) \dashv^s \text{UNF}(\rho)}{\rho \dashv^s \sigma_2}\ \mathit{depth}(\sigma_2) + \mathit{depth}(\rho) > 0;\ [\text{R-Unfold}]$$

Suppose now that $\mathit{depth}(\sigma_2) + \mathit{depth}(\rho) = 0$.

The argument is by case analysis on the form of σ_2.

i) If $\sigma_2 = \mathbf{1}$ then we can derive

$$\frac{}{\rho \dashv^s \sigma_2}\ [\text{A-UNIT}]$$

ii) If $\sigma_2 = ?\mathtt{t}_2.\rho_2'$ then $\text{UNF}(\sigma_1) \precsim^{syn}_{\text{SVR}} \sigma_2$ implies that $\text{UNF}(\sigma_1) = ?\mathtt{t}_1.\rho_1'$ with $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and $\rho_1' \precsim^{syn}_{\text{SVR}} \rho_2'$.

The assumption $\rho \dashv^s \text{UNF}(\sigma_1)$ now implies that $\rho = !\mathtt{t}.\rho'$, with $?\mathtt{t}_1 \bowtie_c !\mathtt{t}$, and $\rho' \dashv^s \sigma_1'$. The fact that $?\mathtt{t}_1 \bowtie_c !\mathtt{t}$ implies that $\mathtt{t} \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$.

We have seen that $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$, and that $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}$. Since $\preccurlyeq_{\mathtt{b}}$ is transitive, $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}$. The definition of $\bowtie_c$ guarantees that $!\mathtt{t}_2 \bowtie_c ?\mathtt{t}_2$. We also know that $\rho_1' \precsim^{syn}_{\text{SVR}} \rho_2'$ and that $\rho' \dashv^s \sigma_1'$, thus $\rho_2' \mathcal{R} \rho'$. We have proven enough to derive

$$\frac{\rho' \dashv^s \sigma_2'}{\rho \dashv^s \sigma_2}\ !\mathtt{t}_2 \bowtie_c ?\mathtt{t}_2;\ [\text{R-ALPHA}]$$

iii) An argument similar to the one we used in the previous case can be used if $\sigma_2 = ?\mathtt{t}_2.\rho_2'$.

iv) If $\sigma_2 = \bigoplus_{i\in I} !\mathtt{l}_\mathtt{i}.\sigma_i^2$, then we prove that we can derive

$$\frac{\cdots}{\rho \dashv^s \sigma_2}\ [\text{R-INCH}]$$

We have to prove that $\rho = \sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho_j$, with $I \subseteq J$, and $\rho_i^2 \mathcal{R} \sigma_i$ for every $i \in I$. As $\text{UNF}(\sigma_1) \precsim^{syn}_{\text{SVR}} \sigma_2$, it follows that $\text{UNF}(\sigma_1) = \bigoplus_{k\in K} !\mathtt{l}_\mathtt{k}.\sigma_k^1$, with $I \subseteq K$, and $\rho_i^1 \precsim^{syn}_{\text{SVR}} \rho_i^2$. As $\rho \dashv^s \text{UNF}(\sigma_1)$ it must be $\rho = \sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho_j$, with $K \subseteq J$, and $\rho_k \dashv^s \sigma_k^1$ for every $k \in K$. As $I \subseteq K \subseteq J$, we have $I \subseteq J$, moreover for every $i \in I$ we have

$$\rho_i^1 \precsim^{syn}_{\text{SVR}} \rho_i^2, \qquad \rho_i^1 \dashv^s \rho_i$$

thus $\rho_i^2 \mathcal{R} \rho_i$.

v) If $\sigma_2 = \sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho_i^2$, then the argument is similar to the one used in the previous case.

□

Corollary 6.4.8. *The restricted compliance server pre-order equals the restricted* MUST *server pre-order.*

Proof. It follows from Proposition 6.4.7 and Proposition 6.1.10. □

6.5 Restricted compliance client pre-order

We introduce a new pre-order which compares the capacity of clients to be satisfied by servers. The structure of this sub-section is similar to that of the previous one on the restricted compliance server pre-order.

Definition 6.5.1. [Restricted compliance client pre-order]
For $\rho_1, \rho_2 \in \mathsf{SC}_{\text{fo}}$ let $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2$ whenever $\rho_1 \dashv \sigma$ implies $\rho_2 \dashv \sigma$ for every σ in SC_{fo}. □

Also the restricted compliance client pre-order let us reason modulo unfolding.

We can reason on $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ modulo unfolding.

Lemma 6.5.2. For every $\rho_1, \rho_2 \in \mathsf{ST}_{\text{fo}}$, $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2$ if and only if $\text{UNF}(\rho_1) \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \text{UNF}(\rho_2)$.

Proof. Follows from Corollary 3.2.9. □

$$\frac{}{1 \text{ DUAL } 1} \text{ [A-DUAL]}$$

$$\frac{}{x \text{ DUAL } x} \text{ [A-VAR]}$$

$$\frac{\rho' \text{ DUAL } \sigma'}{!\mathtt{t}.\rho' \text{ DUAL } ?\mathtt{t}.\sigma'} \text{ [R-IN-F]}$$

$$\frac{\rho' \text{ DUAL } \sigma'}{?\mathtt{t}.\rho' \text{ DUAL } !\mathtt{t}.\sigma'} \text{ [R-OUT-F]}$$

$$\frac{\rho_1 \text{ DUAL } \sigma_1 \quad \dots \quad \rho_{|I|} \text{ DUAL } \sigma_{|I|}}{\bigoplus_{i\in I}!\mathtt{l}_i.\rho_i \text{ DUAL } \sum_{i\in I}?\mathtt{l}_i.\sigma_i} \text{ [R-BRANCH]}$$

$$\frac{\rho_1 \text{ DUAL } \sigma_1 \quad \dots \quad \rho_{|I|} \text{ DUAL } \sigma_{|I|}}{\sum_{i\in I}!\mathtt{l}_i.\rho_i \text{ DUAL } \bigoplus_{i\in I}!\mathtt{l}_i.\sigma_i} \text{ [R-CHOICE]}$$

$$\frac{\rho \text{ DUAL } \sigma}{\mu x.\,\rho \text{ DUAL } \mu x.\,\sigma} \text{ [R-REC]}$$

Figure 6.2: Inference rules for the rule functional $\mathcal{F}_{\text{DUAL}}$

We have seen in Lemma 6.1.5 that the session contract 1 is a bottom element in the restricted compliance server pre-order. The compliance client pre-order enjoys the dual property.

Corollary 6.5.3 (Top element)**.**
The pre-order $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ enjoys the following two properties,

(i) it has a top element

(ii) if $\sigma_\top$ is a top element of $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ then $\text{UNF}(\sigma_\top) = 1$

Proof. Since $1 \dashv \sigma$ for every contract σ, the session contract 1 it is a top element in the restricted compliance client pre-order. Moreover, reasoning as in Lemma 6.1.5 we can show that if $\sigma_\top$ is an arbitrary top element then $\text{UNF}(\sigma_\top) = 1$. □

All the client pre-orders we studied thus far have non usable clients. This is not the case for $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$. The definition of compliance and the restrictive syntax of session contracts let us prove that for every ρ there exists some σ such that $\rho \dashv \sigma$. Intuitively, this is true because of two factors,

- the only stable session contract that does not engage in any action is 1;
- the compliance relation allows everlasting computations.

To prove the main result of this section, Proposition 6.5.18, it is necessary to use the duals of session contracts, so now we explain how to construct the dual of every ρ.

Definition 6.5.4. [Dual]
If $\rho \dashv \sigma$ then we say that σ is a *dual* of ρ. □

We want to prove that every session contract has a dual. This means that for every session contract ρ, we can define a server contract σ such that $\rho \dashv \sigma$. We do this reasoning by induction on the terms of $L_{\mathsf{SC}_{\text{fo}}}$.

Definition 6.5.5. [Dual session contracts]
Let $\mathcal{F}^{\text{DUAL}} : \mathcal{P}(L^2_{\mathsf{SC}_{\text{fo}}}) \longrightarrow \mathcal{P}(L^2_{\mathsf{SC}_{\text{fo}}})$ be the rule functional given by the inference rules in Figure 6.2. Lemma C.0.28 and the Knaster-Tarski theorem ensure that there exists the least solution of the

equation $X = \mathcal{F}_{\text{DUAL}}(X)$; we call this solution the *duality relation*, and we denote it DUAL: That is $\text{DUAL} = \mu X.\mathcal{F}_{\text{DUAL}}(X)$. □

We state two basic properties of DUAL.

Lemma 6.5.6. The relation DUAL is a total function.

Proof. We have to prove two things, that is

a) for every $\rho, \sigma,$ and $\sigma' \in \mathsf{SC}_{\text{fo}}$, if ρ DUAL σ and ρ DUAL σ' then $\sigma = \sigma'$

b) for every $\rho \in L_{\mathsf{SC}_{\text{fo}}}$, there exists a σ such that ρ DUAL σ.

The proof of point (a) is by induction on the derivation of ρ DUAL σ. The proof of point (b) is by structural induction on ρ. □

In view of the previous lemma, from now on instead of using the infix notation ρ DUAL σ, we will use the functional notation $\text{DUAL}(\rho) = \sigma$; except when using the inference rules of Figure 6.2.

Lemma 6.5.7. For every $\rho \in L_{\mathsf{SC}_{\text{fo}}}$, if ρ is closed then $\text{DUAL}(\rho)$ is closed.

Proof. The proof is by structural induction on ρ. The only interesting case is $\rho = \mu x.\, \rho'$. By definition $\text{DUAL}(\rho) = \mu x.\, \text{DUAL}(\rho')$. The fact that $\mu x.\, \text{DUAL}(\rho')$ is closed depends on the fact that the function DUAL is an identity on the variables, and preserves the bindings μx. □

Lemma 6.5.8. [Substitution lemma]
For every $\rho, \sigma, \rho', \sigma' \in L_{\mathsf{SC}_{\text{fo}}}$ and variable x, if $\text{DUAL}(\rho) = \sigma$ and $\text{DUAL}(\rho') = \sigma'$, then the equality $\text{DUAL}(\rho \left\{ {}^{\rho'}/_{x} \right\}) = \sigma \left\{ {}^{\sigma'}/_{x} \right\}$ holds true.

Proof. The argument is by structural induction on ρ.

Base cases If $\rho = \mathsf{1}$, then $\rho \left\{ {}^{\rho'}/_{x} \right\} = \rho$, and the hypothesis DUAL $(\rho) = \sigma$ implies that $\sigma = \mathsf{1}$, so $\sigma \left\{ {}^{\sigma'}/_{x} \right\} = \mathsf{1}$. The equalities we have shown ensure that both $\rho \left\{ {}^{\rho'}/_{x} \right\}$ and $\sigma \left\{ {}^{\sigma'}/_{x} \right\}$ are $\mathsf{1}$, so we derive

$$\frac{}{\rho \left\{ {}^{\rho'}/_{x} \right\} \text{ DUAL } \sigma \left\{ {}^{\sigma'}/_{x} \right\}} \text{ [A-DUAL]}$$

If $\rho = y$ and $y \neq x$, then an argument as the previous one, but with an application of [A-VAR], lets us prove that $\rho \left\{ {}^{\rho'}/_{x} \right\}$ DUAL $\sigma \left\{ {}^{\sigma'}/_{x} \right\}$.

If $\rho = x$, then $\rho \left\{ {}^{\rho'}/_{x} \right\} = \rho'$, and the hypothesis $\text{DUAL}(\rho) = \sigma$ implies that $\sigma = x$, so $\sigma \left\{ {}^{\sigma'}/_{x} \right\} = \sigma'$. The hypothesis that $\text{DUAL}(\rho') = \sigma'$ implies that there exists an finite inference tree that proves $\text{DUAL}(\rho \left\{ {}^{\rho'}/_{x} \right\}) = \sigma \left\{ {}^{\sigma'}/_{x} \right\}$.

Inductive cases The arguments for the inductive cases have the same structure, so we discuss only one case.

If $\rho = !\mathtt{t}.\rho''$ then the hypothesis $\text{DUAL}(\rho) = \sigma$ implies that $\sigma = ?\mathtt{t}.\sigma''$ and that $\text{DUAL}(\rho'') = \sigma''$. Since ρ'' is a sub-term of ρ, the inductive hypothesis states the following implication,

$$\begin{gathered}\text{for every } \hat{\rho}, \hat{\sigma}'', \text{ and } \hat{\sigma} \in L_{\mathsf{SC}_{\text{fo}}}, \text{ and variable } y, \text{ if } \text{DUAL}\ (\rho'') = \hat{\sigma}'' \text{ and } \text{DUAL}\ (\hat{\rho}) = \hat{\sigma}, \\ \text{then } \text{DUAL}\ (\rho'' \left\{ {}^{\hat{\rho}}/_{y} \right\}) = \hat{\sigma}'' \left\{ {}^{\hat{\sigma}}/_{y} \right\}.\end{gathered}$$

The fact that $\text{DUAL}\ (\rho'') = \sigma''$ and the hypothesis that $\text{DUAL}\ (\rho') = \sigma'$ let us use the inductive hypothesis: $\text{DUAL}\ (\rho'' \left\{ {}^{\rho'}/_{x} \right\}) = \sigma'' \left\{ {}^{\sigma'}/_{x} \right\}$. This means that there exists a finite inference tree as the following one,

$$\frac{\vdots}{\rho'' \left\{ {}^{\rho'}/_{x} \right\} \text{ DUAL } \sigma'' \left\{ {}^{\sigma'}/_{x} \right\}}$$

By applying rule [R-In-F] we obtain the next tree,

$$\dfrac{\dfrac{\vdots}{\rho''\left\{{}^{\rho'}/_{x}\right\}\ \textsc{Dual}\ \sigma''\left\{{}^{\sigma'}/_{x}\right\}}}{!\mathtt{t}.(\rho''\left\{{}^{\rho'}/_{x}\right\})\ \textsc{Dual}\ ?\mathtt{t}.(\sigma''\left\{{}^{\sigma'}/_{x}\right\})}\ [\text{R-Out-F}]$$

The definition of capture avoiding substitution now implies that $\textsc{Dual}(\rho\left\{{}^{\rho'}/_{x}\right\}) = \sigma\left\{{}^{\sigma'}/_{x}\right\}$. □

Lemma 6.5.9. For every $\rho \in \mathsf{SC}_{\text{fo}}$, $\textsc{Dual}(\rho) = \sigma$ implies that $\textsc{Dual}(\textsc{unf}(\rho)) = \textsc{unf}(\sigma)$.

Proof. The argument is by induction on the depth of ρ.

Base case ($\mathit{depth}(\rho) = 0$) In this case $\textsc{unf}(\rho) = \rho$ and $\rho \neq \mu x.\,\rho'$. The hypothesis that $\textsc{Dual}(\rho) = \sigma$ ensures that $\sigma \neq \mu x.\,\sigma'$, so $\textsc{unf}(\sigma) = \sigma$. The hypothesis now ensures that $\textsc{Dual}(\textsc{unf}(\rho)) = \textsc{unf}(\sigma)$.

Inductive case ($\mathit{depth}(\rho) = n{+}1$) In the inductive case $\rho = \mu x.\,\rho'$, and the hypothesis $\textsc{Dual}(\rho) = \sigma$ implies that $\sigma = \mu x.\,\sigma'$ and $\textsc{Dual}\ (\rho') = \sigma'$.

Consider the term $\hat{\rho} = \rho'\,\{{}^{\rho}/_{x}\}$. Since $\mathit{depth}(\rho) = n + 1$, the depth of $\hat{\rho}$ is n, so the inductive hypothesis ensures that

$$\text{if } \textsc{Dual}(\hat{\rho}) = \hat{\sigma} \text{ then } \textsc{Dual}(\textsc{unf}(\hat{\rho})) = \textsc{unf}(\sigma).$$

Since $\textsc{Dual}\ (\rho') = \sigma'$ and $\textsc{Dual}\ (\rho) = \sigma$, Lemma 6.5.8 implies that $\textsc{Dual}\ (\rho'\,\{{}^{\rho}/_{x}\}) = \sigma'\,\{{}^{\sigma}/_{x}\}$. In turn this means that $\textsc{Dual}\ (\hat{\rho}) = \sigma'\,\{{}^{\sigma}/_{x}\}$. The inductive hypothesis now ensures that

$$\textsc{Dual}\ (\textsc{unf}(\hat{\rho})) = \textsc{unf}(\sigma'\,\{{}^{\sigma}/_{x}\})$$

The definitions of $\textsc{unf}$ and $\hat{\rho}$ imply that $\textsc{unf}(\hat{\rho}) = \textsc{unf}(\rho)$; the equality $\sigma = \mu x.\,\sigma'$ and the definition of $\textsc{unf}$ imply that $\textsc{unf}(\sigma'\,\{{}^{\sigma}/_{x}\}) = \textsc{unf}(\sigma)$; from these equalities it follows that $\textsc{Dual}\ (\textsc{unf}(\rho)) = \textsc{unf}(\sigma)$. □

Lemma 6.5.10. For every $\rho \in \mathsf{SC}_{\text{fo}}$, $\textsc{Dual}(\rho) \in \mathsf{SC}_{\text{fo}}$.

Proof. The proof is by rule induction on the derivation of the dual of ρ. The argument depends on Lemma 6.5.7 and on the fact that the function $\textsc{Dual}$ does not introduce recursive constructors that are not in ρ. □

Lemma 6.5.11. For every $\rho \in \mathsf{SC}_{\text{fo}}$, $\rho \dashv \textsc{Dual}(\rho)$.

Proof. Fix a session contract ρ. Thanks to Lemma 3.3.9, it is enough to prove that $\rho \dashv^{s} \textsc{Dual}(\rho)$. We defined a suitable co-inductive syntactic compliance,

$$\mathcal{R} = \{\,(\rho,\,\sigma)\ \mid\ \textsc{Dual}(\rho) = \sigma,\ \rho, \sigma \in \mathsf{SC}_{\text{fo}}\,\}$$

We prove that $\mathcal{R} \subseteq\ \mathcal{F}^{\dashv^{s}}(\mathcal{R})$. It suffices to show that each pair in $\mathcal{R}$ can be derived applying one of the rules in Figure 6.2, and drawing the premises from $\mathcal{R}$ itself.

Fix a pair $\rho\ \mathcal{R}\ \sigma$. By definition of $\mathcal{R}$ we know that $\textsc{Dual}(\rho) = \sigma$. We reason first on the depth of ρ and σ.

If $\mathit{depth}(\rho) + \mathit{depth}(\sigma) > 0$, then note that $\textsc{unf}(\rho)\ \mathcal{R}\ \textsc{unf}(\sigma)$; this is a consequence Lemma 6.5.9 and Lemma 6.5.10. We know enough to apply [R-Unfold],

$$\dfrac{\textsc{unf}(\rho) \dashv^{s} \textsc{unf}(\sigma)}{\rho \dashv^{s} \sigma}$$

$$\frac{}{\rho_1 \preceq^{syn}_{\text{CLT}} \mathbf{1}}\ [\text{AX-CLT}]$$

Figure 6.3: Same rules of Figure 6.1, but [AX-SRV] is replaced by [AX-CLT]. Inference rules for the rule functional $\mathcal{F}^{\preceq^{syn}}_{\preceq_{\text{CLT}}}$

If $depth(\rho) + depth(\sigma) = 0$, then we reason by case analysis on ρ. The arguments for all the cases are similar, so we discuss only one case.

If $\rho = !\mathtt{t}.\rho'$, then $\text{DUAL}\ (\rho) = \sigma$ implies that $\sigma = ?\mathtt{t}.\sigma'$, and that $\text{DUAL}\ (\rho') = \sigma'$. Since ρ and σ are session contracts, also ρ' and σ' must be session contracts. It follows that $\rho' \mathcal{R} \sigma'$. We know enough to apply rule [R-OUT-F],

$$\frac{\rho' \dashv^s \sigma'}{\rho \dashv^s \sigma}\ !\mathtt{t} \bowtie_c ?\mathtt{t};\ [\text{R-OUT-F}]$$

□

Lemma 6.5.12. For every $\rho_1, \rho_2 \in \mathsf{SC}_{\text{fo}}$, if $\text{UNF}(\rho_2) \neq \mathbf{1}$ and $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2$, then one of the following is true

(i) if $\text{UNF}(\rho_2) = !\mathtt{t}_2.\rho_2'$ then $\text{UNF}(\rho_1) = !\mathtt{t}_1.\rho_1'$, $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $\rho_1' \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2'$

(ii) if $\text{UNF}(\rho_2) = ?\mathtt{t}_2.\rho_2'$ then $\text{UNF}(\rho_1) = ?\mathtt{t}_1.\rho_1'$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and $\rho_1' \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2'$

(iii) if $\text{UNF}(\rho_2) = \sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho_j^2$ then $\text{UNF}(\rho_1) = \sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho_i^1$ with $I \subseteq J$ and $\rho_i^1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_i^2$

(iv) if $\text{UNF}(\rho_2) = \bigoplus_{j\in J} !\mathtt{l}_\mathtt{j}.\rho_j^2$ then $\text{UNF}(\rho_1) = \bigoplus_{i\in I} !\mathtt{l}_\mathtt{i}.\rho_i^1$ with $J \subseteq I$ and $\rho_j^1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_j^2$

Proof. We prove the implication in point (ii). The other cases are analogous.

Suppose that $\text{UNF}(\rho_2) = ?\mathtt{t}_2.\rho_2'$. The hypothesis $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2$ and Lemma 6.5.2 imply that $\text{UNF}(\rho_1) \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \text{UNF}(\rho_2)$. First we show that $\text{UNF}(\rho_1)$ has the required syntax, and then we prove the properties of the base types and the continuations of the contracts.

Point (b) of Lemma 6.5.6 and Lemma 6.5.11 ensure that there exists a σ such that $\text{UNF}(\rho_1) \dashv \sigma$. It follows that $\text{UNF}(\rho_2) \dashv \sigma$. Pick a stable derivative of σ, σ'. Corollary 3.2.7 ensures that $\text{UNF}(\rho_2) \dashv \sigma'$. Since $\text{UNF}(\rho_2) \overset{\checkmark}{\not\longrightarrow}$, Definition 3.2.1 ensures that $\rho_2 \parallel \sigma \xrightarrow{\tau}$. As both terms in the composition $\rho_2 \parallel \sigma$ are stable, they must interact. Thanks to the syntax of session contracts, it follows that $\sigma = !\mathtt{t}.\sigma'$ for some $\mathtt{t}$. As $\text{UNF}(\rho_1) \dashv \sigma$, it follows that $\text{UNF}(\rho_1) = ?\mathtt{t}_1.\rho_1'$.

Now we prove that $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and that $\rho_1' \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2'$. Since ρ_1' is usable there exists a $\hat{\sigma}'$ such that $\rho_1' \dashv \hat{\sigma}'$. Let $\hat{\sigma} = !\mathtt{t}_1.\hat{\sigma}'$; it is relatively easy to see that $\text{UNF}(\rho_1) \dashv \hat{\sigma}$. It follows that $\text{UNF}(\rho_2) \dashv \hat{\sigma}$. Since $\text{UNF}(\rho_2) \overset{\checkmark}{\not\longrightarrow}$ the session contracts $\text{UNF}(\rho_2)$ and $\hat{\sigma}$ must interact; it follows that $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$, and that $\rho_2' \dashv \hat{\sigma}'$. As we have no assumption on $\hat{\sigma}'$, we have proven that $\rho_1' \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2'$. □

Definition 6.5.13. [Syntactic compliance client pre-order]
Let $\mathcal{F}^{\preceq^{syn}}_{\preceq_{\text{CLT}}} : \mathcal{P}(\mathsf{SC}^2_{\text{fo}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{fo}})$ be the rule functional given by the inference rules in Figure 6.3. If $X \subseteq \mathcal{F}^{\preceq^{syn}}_{\preceq_{\text{CLT}}}(X)$, then we say that X is a *syntactic client pre-order* . Lemma C.0.29 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\preceq^{syn}}_{\preceq_{\text{CLT}}}(X)$; we call this solution the *syntactic client pre-order*, and we denote it $\preceq^{syn}_{\text{CLT}}$. That is $\preceq^{syn}_{\text{CLT}} = \nu X.\mathcal{F}^{\preceq^{syn}}_{\preceq_{\text{CLT}}}(X)$.
□

Corollary 6.5.14. *For every session contract ρ_1 and ρ_2, if $\rho_1 \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \rho_2$ then $\rho_1 \preceq^{syn}_{\text{CLT}} \rho_2$.*

Proof. The argument is similar to the one of Corollary 6.4.4, but here we use the function $\mathcal{F}^{\preceq^{syn}}_{\preceq_{\text{CLT}}}$ and Lemma 6.5.12. □

Corollary 6.5.15. *The pre-order $\sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}}$ is contained the restricted* MUST *client pre-order.*

Proof. Definition 6.5.13 and Definition 6.2.10 imply that $\preceq^{syn}_{\mathrm{CLT}}$ is a co-inductive syntactic must client pre-order, and therefore $\preceq^{syn}_{\mathrm{CLT}} \subseteq \precsim^{syn}_{\mathrm{CLT}}$. Consider the following steps,

$$\begin{array}{rcll} \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} & = & \preceq^{syn}_{\mathrm{CLT}} & \text{Because of Corollary 6.5.14} \\ & \subseteq & \precsim^{syn}_{\mathrm{CLT}} & \text{Proven above} \\ & = & \sqsubseteq^{\mathsf{fo}}_{\sim\mathrm{CLT}} & \text{Because of Proposition 6.2.12} \end{array}$$

□

The converse of the previous corollary it is not true. The third inequality of Example 6.2.8 proves this.

Lemma 6.5.16. Let $\mathcal{R}$ be a co-inductive syntactic client pre-order and let $\rho_1 \preceq^{syn}_{\mathrm{CLT}} \rho_2$. If $\rho_2 \xrightarrow{\tau} \rho_2'$ then $\rho_1 \preceq^{syn}_{\mathrm{CLT}} \rho_2'$.

Proof. The proof is similar to the proof of Lemma 6.4.6. □

Recall the symbol $\sqsubseteq_{\mathrm{RS}}$ (Definition 2.3.3).

Lemma 6.5.17. For every $\rho_1, \rho_2 \in \mathsf{SC}_{\mathrm{fo}}$, if $\sigma_1 \preceq^{syn}_{\mathrm{CLT}} \rho_2$, $\mathrm{UNF}(\sigma_2) \neq \mathsf{1}$ and $B \in \mathrm{ACC}(\rho_2, \varepsilon)$, then there exists a set $A \in \mathrm{ACC}(\rho_1, \varepsilon)$ such that $A \sqsubseteq_{\mathrm{RS}} B$.

Proof. The proof is similar to the proof of Lemma 6.1.11 but it relies on the use of Definition 6.5.13. □

Proposition 6.5.18. [Co-inductive characterisation of $\sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}}$]
Let $\rho, \sigma \in \mathsf{SC}_{\mathrm{fo}}$. Then $\rho \preceq^{syn}_{\mathrm{CLT}} \sigma$ if and only if $\rho \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \sigma$.

Proof. In view of Corollary 6.5.14 we have to prove only the inclusion $\preceq^{syn}_{\mathrm{CLT}} \subseteq \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}}$. It is enough to show that the relation

$$\mathcal{R}' = \{\, (\rho_2, \sigma) \mid \rho_1 \preceq^{syn}_{\mathrm{CLT}} \rho_2,\ \rho_1 \dashv \sigma,\ \text{for some } \rho_1 \in \mathsf{SC}_{\mathrm{fo}} \,\}$$

is a co-inductive compliance. Thanks to Lemma 3.3.10 it is enough to prove that the following relation is a syntactic compliance,

$$\mathcal{R} = \{\, (\rho_2, \sigma) \mid \rho_1 \preceq^{syn}_{\mathrm{CLT}} \rho_2,\ \rho_1 \dashv^{s} \sigma,\ \text{for some } \rho_1 \in \mathsf{SC}_{\mathrm{fo}} \,\}$$

The argument is similar to the proof of Proposition 6.4.7. □

Alternative definition model

The intersection of the pre-orders $\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}$ and $\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}}$ provides the same fully abstract model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ that we exhibited in Theorem 6.3.4.

Proposition 6.5.19. For every $\sigma_1, \sigma_2 \in \mathsf{SC}_{\mathrm{fo}}$, $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{P2P}} \sigma_2$ if and only if $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$ and $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \sigma_2$.

Proof. We have to prove two implications,

1. if $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$ and $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \sigma_2$ then $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{P2P}} \sigma_2$
2. if $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{P2P}} \sigma_2$ then $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2$ and $\sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \sigma_2$

The proof of the first implication amounts in three steps,

$$\begin{array}{ll} \sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \sigma_2,\ \sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \sigma_2 & \text{By assumption} \\ \sigma_1 \sqsubseteq^{\mathsf{fo}}_{\sim\mathrm{SVR}} \sigma_2,\ \sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \sigma_2 & \text{By Corollary 6.4.8} \\ \sigma_1 \sqsubseteq^{\mathsf{fo}}_{\sim\mathrm{SVR}} \sigma_2,\ \sigma_1 \sqsubseteq^{\mathsf{fo}}_{\sim\mathrm{CLT}} \sigma_2 & \text{By Corollary 6.5.15} \\ \sigma_1 \sqsubseteq^{\mathsf{fo}}_{\mathrm{P2P}} \sigma_2 & \text{By Definition 6.3.1} \end{array}$$

$$\begin{array}{ccccc} (\sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{SVR}}} & \cap & \sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{CLT}}}) & \cong & \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \\ \| & & \cup & & \\ (\sqsubset^{\mathsf{fo}}_{\preceq_{\mathrm{SVR}}} & \cap & \sqsubset^{\mathsf{fo}}_{\preceq_{\mathrm{CLT}}}) & \cong & \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \end{array}$$

Figure 6.4: Relations between the restricted pre-orders and first-order sub-typing

To prove the second implication we have to show that $\sqsubset^{\mathsf{fo}}_{\mathrm{P2P}} \subseteq \sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{SVR}}}$ and that $\sqsubset^{\mathsf{fo}}_{\mathrm{P2P}} \subseteq \sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{CLT}}}$. The first set inclusion is true because by definition $\sqsubset^{\mathsf{fo}}_{\mathrm{P2P}} \subseteq \sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{CLT}}}$, and because of Corollary 6.4.8.

The proof of the set inclusion $\sqsubset^{\mathsf{fo}}_{\mathrm{P2P}} \subseteq \sqsubset^{\mathsf{fo}}_{\mathrm{CLT}}$ requires more work. Thanks to the Proposition 6.1.10, Proposition 6.2.12 and Proposition 6.5.18, it is enough to prove that $\precsim^{syn}_{\mathrm{SVR}} \cap \precsim^{syn}_{\mathrm{CLT}} \subseteq \preceq^{syn}_{\mathrm{CLT}}$. Definition 6.5.13 ensures that this inclusion follows from the inclusion $\precsim^{syn}_{\mathrm{SVR}} \cap \precsim^{syn}_{\mathrm{CLT}} \subseteq \mathcal{F}^{\preceq^{syn}_{\mathrm{CLT}}}(\precsim^{syn}_{\mathrm{SVR}} \cap \precsim^{syn}_{\mathrm{CLT}})$. We prove that last inclusion: $\precsim^{syn}_{\mathrm{SVR}} \cap \precsim^{syn}_{\mathrm{CLT}}$ is a co-inductive syntactic compliance client pre-order.

Suppose that $\rho_1 \,(\precsim^{syn}_{\mathrm{SVR}} \cap \precsim^{syn}_{\mathrm{CLT}})\, \rho_2$; the argument to prove that the pair (ρ_1, ρ_2) is in the set $\mathcal{F}^{\preceq^{syn}_{\mathrm{CLT}}}(\precsim^{syn}_{\mathrm{SVR}} \cap \precsim^{syn}_{\mathrm{CLT}})$ is by case analysis on the form of $\textsc{unf}(\rho_1)$.

In view of the definitions of $\precsim^{syn}_{\mathrm{SVR}}$, $\precsim^{syn}_{\mathrm{SVR}}$, and $\preceq^{syn}_{\mathrm{CLT}}$, all the cases except one are straightforward; namely, when $\textsc{unf}(\rho_1) = \sum_{i\in I} ?\mathtt{t}_{\mathtt{i}}.\rho_i^1$. If this is the case, then we reason as we did in the proof of Proposition 6.3.3. □

In this chapter we have investigated the server and the client pre-orders that arise from the MUST testing and the compliance relations, if we restrict our attention to the LTS of session contracts. The main results are Theorem 6.3.4 and Proposition 6.5.19.

We summarise our knowledge on the pre-orders for session contracts in Figure 6.4.

Meaning of Theorem 6.3.4 and Proposition 6.5.19 The fully abstract model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ shown in Theorem 6.3.4 shows that the definition of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ is *not* arbitrary. Indeed, Theorem 6.3.4 explains the syntactic relation in terms of the observable behaviours of session contracts; so if $S_1 \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} S_2$ then the behaviours (i.e. the communication patterns) of $\mathcal{M}(S_1)$ and $\mathcal{M}(S_2)$ are related.

Further, Theorem 6.3.4 proves that the theory of first-order session types can be formulated by means of the testing theory, but reasoning on the LTS $\langle\, \mathsf{SC}_{\mathrm{fo}},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$ instead of the more general LTS $\langle\, \mathsf{CCS}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$.

Proposition 6.5.19 proves that the behavioural explanation of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ given by Theorem 6.3.4 does not depend on the relation for satisfaction that we pick; both MUST and $\dashv$ give rise to the same model.

Pre-orders for peers In this chapter we have not studied the pre-orders generated by the symmetric relations for satisfaction $\textsc{must}^{\mathrm{P2P}}$ and $\dashv_{\mathrm{P2P}}$.

It is possible to prove that the pre-order due to the peer compliance coincide with $\sqsubset^{\mathsf{fo}}_{\mathrm{P2P}}$, and so the theory of compliance provides the behavioural counterpart of the theory of first-order session types.

On the contrary, the peer pre-order due to $\textsc{must}^{\mathrm{P2P}}$ does **not** coincide with $\sqsubset^{\mathsf{fo}}_{\mathrm{P2P}}$.

Example 6.5.20. [Restricted MUST peer pre-order]
Also in the setting of first-order session contracts the usability of peers with respect to the MUST testing is not trivial, for instance $\rho = \mu x.\,!\mathtt{bool}.x$ is not usable.

Define $\sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{P2P}}}$ in the obvious way; since ρ is not a usable peer, the proof of $\rho \sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{P2P}}} \mathbf{1}$ is trivial.

This shows that $\sqsubset^{\mathsf{fo}}_{\precsim_{\mathrm{P2P}}} \not\subseteq \sqsubset^{\mathsf{fo}}_{\mathrm{P2P}}$, and so the restricted MUST peer pre-order does not model the sub-typing $\preccurlyeq_{\mathsf{sbt}}$. □

In view of the example above, we can see the testing theory on first-order session types as slightly more general than the compliance theory.

6.6 Related Work

The refinements for session contracts that have been proposed thus far in the literature have been inspired either by MUST testing (and defined using the compliance), or by should testing.

According to this criterion, first we compare the refinement $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ with the pre-orders used in the papers that have influenced us most, namely [Barbanera and de'Liguoro, 2010; Laneve and Padovani, 2008]. The theories presented in these papers are related the compliance relation; thus we will refer to them as *compliance-theories*.

Afterwards, we compare $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ with the refinements of two theories inspired to the fair testing, the one of Bravetti and Zavattaro that we have already discussed, and the theory of "fair sub-typing" proposed by [Padovani, 2011]. We refer to those theories as *fair-theories*. As we will see, the compliance-theories bear some similarities with our results; whereas the fair-theories turns out to generate pre-orders not comparable with $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$.

From now on we reason under the assumption that in our definition of compliance the synchronisation relation $\bowtie$ be the usual co-action relation $\bar{\cdot}$ (so $\bowtie = \{\, (\alpha, \overline{\alpha}) \mid \alpha \in Act \,\}$).

Must-theories

Similar to [Laneve and Padovani, 2007], also the paper [Laneve and Padovani, 2008] uses constrained contracts (see the discussion in Section 3.4). [Laneve and Padovani, 2008] presents the first comparison between a sublanguage of first-order session types and contracts; in particular, it tries to show that the subcontract relation $\preceq^{lp08}$ together with two interpretations similar to $\mathcal{M}$ provide two sound models for the sub-typing. These interpretations are denoted $[\![-]\!]_1$ and $[\![-]\!]_0$. The proposed full abstraction result, [Laneve and Padovani, 2008, see Theorem 2], though, appears not to be true. According to that definition and the interpretation $[\![-]\!]_0$

$$\emptyset[\mathsf{0}] \preceq^{lp08} \{\ell\}[\ell.\,\mathsf{0}]$$

Their Theorem 2 therefore implies $\&\langle\, \ell : \text{END} \,\rangle \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \text{END}$, which is not true. On the other hand if $[\![-]\!]_1$ is used then there are two issues. According to Theorem 2 the pair $(\text{END}, \&\langle\, \ell : \text{END} \,\rangle)$ is interpreted as $(\emptyset[\,\checkmark.\,\mathsf{0}], \{\ell\}[\ell.\,\checkmark.\,\mathsf{0}])$. Then

1. neither $\emptyset[\,\checkmark.\,\mathsf{0}]$ nor $\{\ell\}[\ell.\,\checkmark.\,\mathsf{0}]$ are constrained contracts, because their interfaces do not contain all the action names which appear in the respective behaviours; moreover
2. even if the interpretation was correct, Theorem 2 would be false because

$$\{\,\checkmark\}[\,\checkmark.\,\mathsf{0}] \preceq^{lp08} \{\ell,\,\checkmark\}[\ell.\,\checkmark.\,\mathsf{0}]$$

   while, as stated above, $\&\langle\, \ell : \text{END} \,\rangle \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \text{END}$ is not true.

Our study of the restricted pre-orders on session contracts (Definition 6.4.1 and Definition 6.5.1) is clearly inspired by [Barbanera and de'Liguoro, 2010]. In [Barbanera and de'Liguoro, 2010] the language for session types is the same one as we used, whereas the subset of contracts in which session types are embedded is the set of *session behaviours*. The set of session behaviours is bigger than the set of session contracts because of the lack of distinction between labels and base types. It is possible to write session behaviours as

$$?\mathtt{Int}.\,\mathsf{1} + ?\mathtt{l}_1.\,\mathsf{1}$$

which are image of no session type according to the given interpretation $[\![-]\!]$. Note, though, that $[\![-]\!] = \mathcal{M}$, so the range of $[\![-]\!]$ is the set of session contracts, and our Theorem 6.3.4 proves that $[\![-]\!]$ and their pre-order $\preceq$: [Barbanera and de'Liguoro, 2010, Definition 3.4] provides a complete model for the sub-typing. The completeness of $\preceq$: was only conjectured in [Barbanera and de'Liguoro, 2010].

Their approach is complementary to ours, in that they provide a co-inductive characterisation of the pre-order $\preceq$:, which turns out to equal the intersection of their sub-server and sub-client pre-orders. In contrast, we have studied the (restricted) server and the client pre-orders independently, providing their co-inductive characterisations; we have then (1) explained why it is necessary to use the intersection of the two pre-orders to obtain a fully-abstract model; and (2) proven that the intersection of these pre-orders is a sound *and complete* model of the sub-typing.

Fair-theories

We compare the peer pre-order $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ with the pre-orders proposed in [Padovani, 2011], and [Bravetti and Zavattaro, 2009].

To begin with, observe that the pre-order $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ allows the refinements such as the following one,

$$a \oplus b \sqsubseteq a \tag{6.8}$$

For instance we can prove the following facts

$$\begin{array}{lcl} \mu x.\,(\ !\texttt{espresso}.x \oplus !\texttt{moka}.\,1\ ) & \sqsubseteq^{\mathsf{fo}}_{\text{P2P}} & \mu x.\,!\texttt{espresso}.x \\ \mu X.\ \oplus \langle\, \texttt{livelock}\colon X,\ \texttt{stop}\colon \text{END}\,\rangle & \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} & \mu X.\ \oplus \langle\, \texttt{livelock}\colon X\,\rangle \end{array} \tag{6.9}$$

In [Padovani, 2011] it is pointed out that the refinements shown above are not sound with respect to the fair testing of Rensink and Vogler. Indeed, this lets us prove that the refinements proposed in by Padovani and Bravetti and Zavattaro are not contained in our relation $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$.

In Section 4.4 and Section 5.4 we have already discussed the details of [Bravetti and Zavattaro, 2009], thereby showing that their refinements for peers differ from our refinements. This is the case also in for the pre-order $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$: the relations $\preceq_O^{-1}$ of [Bravetti and Zavattaro, 2009] are not comparable with $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$. Compare the inequalities in Eq. (6.9) with the following one,

$$\mu x.\,(\ \tau.!\texttt{livelock}.x + \tau.!\texttt{stop}.\,1\ ) \not\preceq^{-1}_{\emptyset} \mu x.\,!\texttt{livelock}.x$$

Thus our relation $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ is coarser than $\preceq_{\emptyset}$. As $\preceq^{-1}_{\mathcal{N}}$ relates also terms more general than first-order session contracts, we have the following facts

$$\sqsubseteq^{\mathsf{fo}}_{\text{P2P}} \not\subseteq \preceq^{-1}_{\emptyset}, \quad \preceq^{-1}_{\mathcal{N}} \not\subseteq \sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$$

Similar to the work of Bravetti and Zavattaro, in [Padovani, 2011] the notion of correctness requires *all* the components of a composition to be successful (ie. be able of performing $\checkmark$) *at the same time* in order for the whole composition to be successful. This requirement implies that the compositions which contain terms as

$$\mathsf{0}, \quad \mu x.\,a.x$$

cannot be correct, because the contracts above do not perform $\checkmark$ at all. This phenomenon renders the viability of contracts [Padovani, 2011, Definition 3.1] a non trivial matter; on the contrary, in our theory every peer is viable with respect to $\dashv$.

The language used in [Padovani, 2011] is similar to our session contracts, the differences being that actions are decorated with a role tag $\mathtt{p}$, $\mathtt{q}$, ...; and there is a special session type FAIL. Then sessions are *multiparty*, that is they are general compositions of session types (tagged with a role), for instance

$$\mathtt{p}_1 \colon T_1 \ ||\ \mathtt{p}_2 \colon T_1 \ ||\ \ldots \ ||\ \mathtt{p}_k \colon T_k$$

The notion of correct session type composition is given in [Padovani, 2011, Definition 2.1], and it is used to define a set-theoretical sub-typing relation on session types [Padovani, 2011, Definition 2.2],

which is denoted $\leq$. We can prove

$$\mu x.\,(\,\mathtt{p!livelock}.x \oplus \mathtt{p!stop}.\,1\,) \quad \not\leq \quad \mu x.\,\mathtt{p!livelock}.x$$

because the term $\mu x.\,\mathtt{p!livelock}.x$ cannot reach a successful state at all. This means that (6.8) is not sound for $\leq$.

Also the following inequalities are true:

$$\begin{array}{lll} \mu x.\,\mathtt{p!livelock}.x & \leq & \mu x.\,\mathtt{p!stop}.x \\ \mu x.\,\mathtt{!livelock}.x & \not\sqsubseteq^{\mathsf{fo}}_{\text{P2P}} & \mu x.\,\mathtt{!stop}.x \end{array}$$

The first fact is true because no composition containing the session type $\mu x.\,\mathtt{p!livelock}.x$ can be correct, as this term does not perform $\checkmark$ at all. The second fact is true because from $\mu x.\,\mathtt{!livelock}.x \dashv \mu x.\,\mathtt{?livelock}.x$ and $\mu x.\,\mathtt{!stop}.x \not\dashv \mu x.\,\mathtt{?livelock}.x$.

It follows that the relations $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ and $\leq$ are not comparable,

$$\leq \not\subseteq \sqsubseteq^{\mathsf{fo}}_{\text{P2P}}, \qquad \sqsubseteq^{\mathsf{fo}}_{\text{P2P}} \not\subseteq \leq \tag{6.10}$$

We leave for future work the comparison of the MUST peer pre-order on first-order session contracts, $\sqsubseteq^{\mathsf{fo}}_{\sim\text{P2P}}$, with the fair pre-orders studied by Bravetti and Zavattaro, and Padovani.

Part II

Higher-order theories

Chapter 7

Higher-Order Languages

In the first part of this thesis we have been concerned with theories for first-order languages. Our investigation has lead to two definitions of a fully abstract model of the sub-typing on first-order session types (see Theorem 6.3.4 and Proposition 6.5.19). In particular, we have proven the following isomorphism,

$$\sqsubseteq^{\mathsf{fo}}_{\mathrm{SVR}} \cap \sqsubseteq^{\mathsf{fo}}_{\mathrm{CLT}} \cong \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}} \tag{7.1}$$

From a technical standpoint this result is not yet satisfactory.

Higher-order session types Session types are one of the most studied type systems for concurrent languages; one of their typical application is within the pi-calculus [Sangiorgi and Walker, 2001]. Roughly speaking, the primitive operations in the pi-calculus are the input and output of names *over names*; observe the following interaction of two processes,

$$a!(b).P \;||\; a?(x:S).Q \xrightarrow{\tau} P \;||\; Q\left\{{}^{b}/_{x}\right\}$$

The process on the left, $a!(b).P$, is willing to output the name b through the name a; the process on the right is willing to input over the name a another name, that will replace x in Q.[1] Since an interaction on a can happen, the name b is moved from one process to the other one, by using name a.

Note the type annotation S in the input construct $?(x:S).Q$. The session type S is meant to describe how Q behaves on x, and indeed this is an intuition behind the typing discipline: they are assigned to the names manipulated by processes, and describe how the names are manipulated. What, then, is the session type that describes how a is used by the process $a?(x:S).Q$?

Assuming that Q acts on a according to the type T, then $a?(x:S).Q$ acts on a according to the type $S_a = ?[\,S\,];T$. Noticeably, S_a contains another session types in the input field.

This discussion shows that in order to type names in the pi-calculus, it is necessary to use higher-order session types, that is types that can input/output other types. The following terms are an example,

$$![\,\mu X.\ ?[\,\mathtt{Bool}\,];X\,];\mathrm{END}, \qquad \&\langle\, \mathtt{opt1}\colon\ ?[\,\mathrm{END}\,];\mathrm{END},\ \mathtt{opt2}\colon\ ![\,?[\,\mathrm{END}\,];\mathrm{END}\,];\mathrm{END}\,\rangle$$

Moreover, the co-inductive definition of the sub-typing [Gay and Hole, 2005] on session types is formulated in terms of higher-order types.

To provide a model for the sub-typing à la Gay and Hole, then we have to extend the result shown in Eq. (7.1), to the language of higher-order session types. This is our aim in this part of the thesis. We will extend to the higher-order setting only the model defined using the compliance, and we leave as an open problem the extension of the model due to the MUST testing (see (Q13) in Section 11.2).

[1] Intuitively the input $?(x:S).Q$ binds x in Q as $\lambda x:\mathtt{t}.M$ binds x in M.

S, T	$::=$	**Higher-order session types**
	$\vdots$	see Figure 2.1
	$?[\,T\,]; S$	*Input*
	$![\,T\,]; S$	*Output*

Figure 7.1: Additional terms for higher-order input/output

$$S\mathbf{s} = \begin{cases} \vdots & \\ ![\,T\mathbf{s}\,]; (S'\mathbf{s}) & \text{if } S = ![\,T\,]; S' \\ ?[\,T\mathbf{s}\,]; (S'\mathbf{s}) & \text{if } S = ?[\,T\,]; S' \end{cases}$$

Figure 7.2: Additional cases for substitution on session types.

To model the higher-order session types, we extend the language of first-order session contracts, so as to let them input/output session contracts. This forces us to parametrise the LTS over binary relations $\mathcal{B}$ on session contracts. To this end, we merely transform the relation $\bowtie$ in the side conditions of rule [P-SYNCH], into a function of $\mathcal{B}$: $\bowtie_{\mathcal{B}}$. This allows us to extend smoothly the result in Eq. (7.1) to the higher-order setting.

In Chapter 6 we leveraged the restricted LTS of session contracts to model the sub-typing on first-order session types. As the extension of the model is our only aim in this part of the thesis, we will not be concerned with the LTS of processes any longer.

In this chapter we present two languages, namely the language of higher-order session types, and the language of higher-order session contracts. We adapt to the new setting the definition of sub-typing, and also the operational semantics of session contracts. We also show that our definition of sub-typing and the [Gay and Hole, 2005, Definition 4] generate the same relation (Lemma 7.1.6).

Structure of the chapter. In Section 7.1 we extend the theory of session types so as to include higher-order terms. In Section 7.2 we extend the theory of session contracts, and, most importantly, we parametrise their LTS over the binary relations on session contracts. We also generalise the compliance relation, and prove that its syntactic characterisation is valid also when the LTS is parametrised.

7.1 Session types

Let the language L_T be the set of terms described in Figure 7.1. In Figure 7.2 we adapt the way in which we apply syntactic substitutions to terms.

What we have seen in Section 2.1 about unfolding and guardedness of terms is still true for the extended language, and the definitions of *depth*, UNF and gd do not change.

Definition 7.1.1. [Higher-order session types]
Let $\mathsf{ST}_{\mathrm{ho}}$ denote the set of closed guarded terms of L_T,

$$\mathsf{ST}_{\mathrm{ho}} = \{\, T \in L_{\mathsf{ST}_{\mathrm{ho}}} \mid T \text{ closed}, \; T\,\mathsf{gd} \,\}$$

We refer to the elements in $\mathsf{ST}_{\mathrm{ho}}$ as *higher-order session types*. □

$$\frac{S'_1 \preccurlyeq_{\mathsf{sbt}} S'_2 \quad \hat{S}_1 \preccurlyeq_{\mathsf{sbt}} \hat{S}_2}{?[\,\hat{S}_1\,]; S'_1 \preccurlyeq_{\mathsf{sbt}} ?[\,\hat{S}_2\,]; S'_2}\ [\text{R-IN-H}]$$

$$\frac{S_1 \preccurlyeq_{\mathsf{sbt}} S_2 \quad \hat{S}_2 \preccurlyeq_{\mathsf{sbt}} \hat{S}_1}{![\,\hat{S}_1\,]; S_1 \preccurlyeq_{\mathsf{sbt}} ![\,\hat{S}_2\,]; S_2}\ [\text{R-OUT-H}]$$

Figure 7.3: Additional inference rules for the rule functional $\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}$. See also the rules in Figure 2.6

We amend the definition of the sub-typing relation so as to account for the higher-order terms.

Definition 7.1.2. [Sub-typing]
Let $\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}} : \mathsf{ST}^2_{\mathrm{ho}} \longrightarrow \mathsf{ST}^2_{\mathrm{ho}}$ be the rule functional given by the inference rules in Figure 7.3. If $X \subseteq \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(X)$, then we say that X is a *type simulation*. Lemma C.0.30 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(X)$; we call this solution the *sub-typing relation*, and we denote it $\preccurlyeq_{\mathsf{sbt}}$. That is $\preccurlyeq_{\mathsf{sbt}} = \nu X.\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(X)$. □

Example 7.1.3. [Sub-typing on higher-order types]
Let $S = \mu X.\ ?[\,X\,]; X$ and $T = \mu Y.\ ?[\,Y\,]; ?[\,Y\,]; Y$. In this example we prove that $\mu X.\ ?[\,X\,]; X \preccurlyeq_{\mathsf{sbt}} \mu Y.\ ?[\,Y\,]; ?[\,Y\,]; Y$.

Thanks to the Knaster-Tarski theorem, we have to exhibit a prefixed point of the rule functional $\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}$, that contains the pair (S, T). Consider the following relation

$$\mathcal{R} = \{\, (S, T),\ (?[\,S\,]; S,\ ?[\,T\,]; ?[\,T\,]; T),\ (?[\,S\,]; S,\ ?[\,T\,]; T),\ (S,\ ?[\,T\,]; T) \,\}$$

To show that $\mathcal{R} \subseteq \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(\mathcal{R})$, we have to prove that each pair in $\mathcal{R}$ can be inferred by applying one of the rules in Figure 7.3 using the elements of $\mathcal{R}$ as premises. The following one step derivations show how to infer all the pairs in $\mathcal{R}$.

$$\frac{?[\,S\,]; S \preccurlyeq_{\mathsf{sbt}} ?[\,T\,]; ?[\,T\,]; T}{S \preccurlyeq_{\mathsf{sbt}} T}\ \mathit{depth}(S) + \mathit{depth}(T) > 0;\ [\text{R-UNFOLD}]$$

$$\frac{S \preccurlyeq_{\mathsf{sbt}} ?[\,T\,]; T \quad S \preccurlyeq_{\mathsf{sbt}} T}{?[\,S\,]; S \preccurlyeq_{\mathsf{sbt}} ?[\,T\,]; ?[\,T\,]; T}\ [\text{R-IN-H}]$$

$$\frac{?[\,S\,]; S \preccurlyeq_{\mathsf{sbt}} ?[\,T\,]; T}{S \preccurlyeq_{\mathsf{sbt}} ?[\,T\,]; T}\ \mathit{depth}(S) + \mathit{depth}(T) > 0;\ [\text{R-UNFOLD}]$$

$$\frac{S \preccurlyeq_{\mathsf{sbt}} T \quad S \preccurlyeq_{\mathsf{sbt}} T}{?[\,S\,]; S \preccurlyeq_{\mathsf{sbt}} ?[\,T\,]; T}\ [\text{R-IN-H}]$$

□

The additional inference rules of Figure 7.3 have no impact on the unfolding rule, so Lemma 2.1.16 is true also for $\preccurlyeq_{\mathsf{sbt}}$.

Lemma 7.1.4. [$\preccurlyeq_{\mathsf{sbt}}$ and unfolding]
For every co-inductive type simulation $\mathcal{R}$, and every $S, T \in \mathsf{ST}_{\mathrm{fo}}$, if $S\ \mathcal{R}\ T$ then $\mathrm{UNF}(T)\ \mathcal{R}\ \mathrm{UNF}(T)$.

Proof. The argument is the same used in Lemma 2.1.16. □

Our main aim in this section was to introduce the relation $\preccurlyeq_{\mathsf{sbt}}$. As session types have no semantics, the work we had to do was minimal.

Before proceeding, though, we make sure that our sub-typing $\preccurlyeq_{\mathsf{sbt}}$ coincides with the co-inductive sub-typing $\leq_c$ of [Gay and Hole, 2005], up-to the presence of base types and the restriction to types with only one parameter in the input/output field.

In the sequel, U is another meta variable for higher-order session types.

Definition 7.1.5. [Co-inductive sub-typing]
A relation $\mathcal{R}$ is a *type simulation à la GH* if $(T, U) \in R$ implies the following conditions:

- if $\text{UNF}(T) = \text{END}$ then $\text{UNF}(U) = \text{END}$
- if $\text{UNF}(T) = ?[\,T_1\,]; S_1$ then $\text{UNF}(U) = ?[\,U_1\,]; S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(T_1, U_1) \in \mathcal{R}$.
- if $\text{UNF}(T) = ![\,T_1\,]; S_1$ then $\text{UNF}(U) = ![\,U_1\,]; S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(U_1, T_1) \in \mathcal{R}$
- if $\text{UNF}(T) = \&\langle\, \mathtt{l}_1 \colon S_1, \ldots l_m \colon S_m \,\rangle$ then $\text{UNF}(U) = \&\langle\, \mathtt{l}_1 \colon S'_1, \ldots, \mathtt{l}_n \colon S'_n \,\rangle$ where $m \leq n$ and $(S_i, S'_i) \in \mathcal{R}$ for all $i \in [1, \ldots, m]$
- if $\text{UNF}(T) = \oplus\langle\, \mathtt{l}_1 \colon S_1, \ldots l_m \colon S_m \,\rangle$ then $\text{UNF}(U) = \oplus\langle\, \mathtt{l}_1 \colon S'_1, \ldots, \mathtt{l}_n \colon S'_n \,\rangle$ where $n \leq m$ and $(S_i, S'_i) \in \mathcal{R}$ for all $i \in [1, \ldots, n]$

The *co-inductive sub-typing* relation $\leq_c$ is defined by $T \leq_c U$ if and only if there exists a type simulation $\mathcal{R}$ such that $(T, U) \in \mathcal{R}$. □

Lemma 7.1.6. Let T and U be higher-order session types.

i) if $T \leq_c U$ then $T \preccurlyeq_{\mathsf{sbt}} U$

ii) if T and U contain no base type and $T \preccurlyeq_{\mathsf{sbt}} U$, then $T \leq_c U$

Proof. We have to prove two implications; we begin explaining why i) is true.

To prove that if $T \leq_c U$ then $T \preccurlyeq_{\mathsf{sbt}} U$ is equivalent to showing that $\leq_c \subseteq \preccurlyeq_{\mathsf{sbt}}$. To prove the set inclusion it is enough to show that $\leq_c \subseteq \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(\leq_c)$. We prove that if $T \leq_c U$ then $(T,\, U) \in \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(\leq_c)$.

Fix a pair $T \leq_c U$; we have to prove that an application of one of the inference rules that define $\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}$ (see Figure 7.3) lets us derive $T \preccurlyeq_{\mathsf{sbt}} U$, by using the elements in the relation $\leq_c$ as premises.

We reason first on the depth of the types T and U. Suppose that $\mathit{depth}(T) + \mathit{depth}(U) > 0$; in this case observe that the definition of $\leq_c$ ensures that $\text{UNF}(T) \leq_c \text{UNF}(U)$. We know enough to apply [R-UNFOLD]:

$$\frac{\text{UNF}(R) \preccurlyeq_{\mathsf{sbt}} \text{UNF}(U)}{T \preccurlyeq_{\mathsf{sbt}} U}\ \mathit{depth}(T) + \mathit{depth}(U) > 0;\ [\text{R-UNFOLD}]$$

Suppose now that $\mathit{depth}(T) + \mathit{depth}(U) = 0$. Then $T = \text{UNF}(T)$ and $U = \text{UNF}(U)$, and the argument proceeds by case analysis on T.

- If $T = \text{END}$, then the definition of $\leq_c$ ensures that $u = \text{END}$, so we can derive

$$\frac{}{T \preccurlyeq_{\mathsf{sbt}} U}\ [\text{A-END}]$$

- If $T = ?[\,T_1\,]; S_1$ then $U = ?[\,U_1\,]; S_2$, $T_1 \leq_c U_1$ and $S_1 \leq_c S_2$. We apply rule [R-IN-H]:

$$\frac{S_1 \preccurlyeq_{\mathsf{sbt}} S_2 \quad T_1 \preccurlyeq_{\mathsf{sbt}} U_1}{T \preccurlyeq_{\mathsf{sbt}} U}$$

- If $T = ![\,T_1\,]; S_1$ then the argument is similar to the previous one, but relies on [R-OUT-H].
- If $T = \&\langle\, \mathtt{l}_1 \colon S_1, \ldots l_m \colon S_m \,\rangle$, then then $\text{UNF}(U) = \&\langle\, \mathtt{l}_1 \colon S'_1, \ldots, \mathtt{l}_n \colon S'_n \,\rangle$ where $m \leq n$ and $S_i \leq_c S'_i$ for all $i \in [1, \ldots, m]$. It follows that we can apply rule [R-BRANCH],

$$\frac{S_1 \preccurlyeq_{\mathsf{sbt}} S'_1 \quad \ldots \quad S_m \preccurlyeq_{\mathsf{sbt}} S'_m}{T \preccurlyeq_{\mathsf{sbt}} U}\ m \leq n;\ [\text{R-BRANCH}]$$

- If $T = \oplus\langle \mathtt{l}_1\colon S_1, \ldots l_m\colon S_m \rangle$ the argument is similar to the previous one, but relies on rule [R-Choice].

We have proven that if $T \leq_c U$, then $(T, U) \in \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(\leq_c)$; this means that $\leq_c$ is a prefixed point of $\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}$, and so the definition of $\preccurlyeq_{\mathsf{sbt}}$ ensures that $\leq_c \subseteq \preccurlyeq_{\mathsf{sbt}}$.

We have proven the first implication of the lemma; now we prove the second implication: if T and U contain no base type and $T \preccurlyeq_{\mathsf{sbt}} U$, then $T \leq_c U$. To prove the implication, it is enough to show that the following relation is a type simulation à la GH,

$$\mathcal{R} = \{\, (T, U) \mid T \preccurlyeq_{\mathsf{sbt}} U,\ T, U \text{ contain no base types} \,\}$$

Observe that $\mathcal{R} \subseteq \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(\mathcal{R})$.

Fix a pair $T\ \mathcal{R}\ U$. Lemma 7.1.4 ensures that $\textsc{unf}(T)\ \mathcal{R}\ \textsc{unf}(U)$; and so the set inclusion above implies that $(\textsc{unf}(T), \textsc{unf}(U)) \in \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(\mathcal{R})$. This means that one of the inference rules that define $\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}$ allows us to derive $\textsc{unf}(T) \preccurlyeq_{\mathsf{sbt}} \textsc{unf}(U)$ by using the elements in $\mathcal{R}$ as premises. Since $\mathit{depth}(\textsc{unf}(T)) + \mathit{depth}(\textsc{unf}(U)) = 0$ the rule that lets us derive

$$\textsc{unf}(T) \preccurlyeq_{\mathsf{sbt}} \textsc{unf}(U) \tag{7.2}$$

is not [R-Unfold].

The argument is by case analysis on $\textsc{unf}(T)$.

- If $\textsc{unf}(T) = \textsc{end}$, then Eq. (7.2) must have been derived by using [A-End], so $\textsc{unf}(U) = \textsc{end}$.
- If $\textsc{unf}(T) = ?[\,T_1\,]; S_1$, then Eq. (7.2) must have been derived by using [R-In-H], so $\textsc{unf}(U) = ?[\,U_1\,]; S_2$ and the premises of the rule ensures that $S_1\ \mathcal{R}\ S_2$ and $T_1\ \mathcal{R}\ S_1$.
- If $\textsc{unf}(T) = ![\,T_1\,]; S_1$ then the argument is similar to the previous one, but we use rule [R-Out-H].
- If $\textsc{unf}(T) = \textsc{unf}(T) = \&\langle \mathtt{l}_1\colon S_1, \ldots l_m\colon S_m \rangle$ then Eq. (7.2) must have been derived by using rule [R-Branch]. It follows that $unfoldU = \&\langle \mathtt{l}_1\colon S'_1, \ldots, \mathtt{l}_n\colon S'_n \rangle$, for some $n \in \mathbb{N}$ such that $m \leq n$; moreover the premises of the rule ensure that for every $i \in [1; m]$, $S_i\ \mathcal{R}\ S'_i$.
- If $\textsc{unf}(T) = \oplus\langle \mathtt{l}_1\colon S_1, \ldots l_m\colon S_m \rangle$ the argument is similar to the previous one, but relies on rule [R-Choice]

Note that in the case analysis above we have not considered the cases in which $\textsc{unf}(T)$ performs an input/output on a base type. The definition of $\mathcal{R}$ ensure that these cases cannot happen, for T and U contain no base types, so neither their unfoldings do.

□

In the next section we turn our attention to the higher-order session contracts and their LTSs.

7.2 Session Contracts

In Chapter 2 we introduced session contracts to assign to session types an LTS via a *straightforward* interpretation, namely $\mathcal{M}$, which preserves the sub-typing.

In the new setting, the language $\mathsf{SC}_{\mathsf{fo}}$ does not provide any straightforward way to encode the terms of $\mathsf{ST}_{\mathrm{ho}}$, so as to preserve $\preccurlyeq_{\mathsf{sbt}}$. Nevertheless, we resolved to extend Theorem 6.3.4 to the higher-order setting. To this end, in this section we introduce the language of higher-order session contracts $\mathsf{SC}_{\mathrm{HO}}$. This language provides a natural way to encode session types, thereby assigning them a operational semantics.

ρ, σ	::=	**Higher-order session contracts**
	$\vdots$	see Figure 2.9
	$!(\sigma).\sigma$	*Higher-order output*
	$?(\sigma).\sigma$	*Higher-order input*

Figure 7.4: Additional syntax for higher-order session contracts

The arguments about the syntax of the new language are straightforward; what is noticeably more involved is the LTS that we use. In fact, higher-order session contracts do not have *one* LTS, but an infinite amount of LTSs. Thanks to the restrictive syntax of the language, though, all these LTSs enjoy some properties that we will take advantage of.

After having discussed the syntax and the semantics of higher-order session contracts, we introduce some technicalities that we will need further on. We also adapt to the new setting the definition of compliance and its syntactic characterisation (Lemma 3.3.10).

The language L_{ho} is given by the grammar in Figure 7.4. The depth and the unfoldings of terms in L_{ho} is handled as in Section 2.3.

Definition 7.2.1. [Language of higher-order session contracts]
Let $\mathsf{SC}_{\text{HO}} = \{\, \sigma \in L_{\mathsf{ho}} \mid \sigma \text{ closed}, \sigma\, \mathsf{gd} \,\}$. We refer to the terms in the set SC_{HO} as *session contracts*. □

Let $\eta, \theta, \ldots$ range over the sets $Act \cup \mathsf{SC}_{\text{HO}}$, and μ range over $\mathsf{Act}_{\tau\checkmark} \cup \mathsf{SC}_{\text{HO}}$. From now one we use a series of symbols to range over relations on SC_{HO}; we let

- $\mathcal{B}$ denote a binary relation on SC_{HO}
- $\mathcal{R}$ denote a (reflexive) binary relation on SC_{HO}
- $\mathcal{T}$ denote a transitive relation on SC_{HO}

Operational semantics and interactions

In Section 2.1 we used the rules in Figure 2.8 and Figure 2.10 to define the LTS $\langle \mathsf{SC}_{\text{fo}}, \mathsf{Act}_{\tau\checkmark}, \longrightarrow \rangle$.

To use the same technique here we have to explain when higher-order session contracts can interact; we have to discuss rule [P-Synch]. Since now input and output actions can be session contracts as well, we have to amend the side condition of rule [P-Synch],

$$\frac{q \xrightarrow{\alpha} q' \quad p \xrightarrow{\beta} p'}{q \mid\mid p \xrightarrow{\tau} q' \mid\mid p'}\ \alpha \bowtie \beta;\ \text{[P-Synch]}$$

We would like that also session contracts be related by $\bowtie$. There are two ways to do so. We may define a particular $\bowtie'$, that relates higher-order session contracts in a fixed way; this is similar to what we did with $\bowtie_c$. In Section 2.3 this was a sound idea, for the relation $\preccurlyeq_{\mathsf{b}}$, which we assumed, provides a way to compare base types. Indeed, the relation $\bowtie_c$ depends on $\preccurlyeq_{\mathsf{b}}$. This technique, though, cannot be easily used in the new setting: it is not clear how to assume a priori a pre-order on SC_{HO} that we could use to define a $\bowtie'$. This difficulty leads to the second way to extend $\bowtie$ to higher-order session contracts. Since a priori we do not know when two session contracts, say σ and ρ, should be deemed as "co-contracts", i.e. $\sigma \bowtie \rho$, we make $\bowtie$ depend on a parameter $\mathcal{B}$, and we use the following definition

$$\frac{\rho \xrightarrow{\eta}_{\mathcal{B}} \rho' \quad \sigma \xrightarrow{\theta}_{\mathcal{B}} \sigma'}{\rho \,||\, \rho \xrightarrow{\tau}_{\mathcal{B}} \sigma' \,||\, \sigma'}\ \eta \bowtie_{\mathcal{B}} \theta;\ \text{[P-SYNCH]}$$

Figure 7.5: Operational semantics of recursive session contracts

Parameter of $\bowtie$	Interactions (as per rules in Figure 7.5)
$\mathcal{B} = \emptyset$	$\rho \,\|\|\, \sigma \not\xrightarrow{\tau}_{\mathcal{B}}$
$\mathcal{B} = \{(0,\,0)\}$	$\rho \,\|\|\, \sigma \xrightarrow{\tau}_{\mathcal{B}} 1 \,\|\|\, 0$
$\mathcal{B} = \{(0,\,0), (1,\,1)\}$	$\rho \,\|\|\, \sigma \xrightarrow{\tau}_{\mathcal{B}} 1 \,\|\|\, 0$ $\rho \,\|\|\, \sigma \xrightarrow{\tau}_{\mathcal{B}} 0 \,\|\|\, 0$

Figure 7.6: Interactions of two contracts, as the parameter $\mathcal{B}$ of $\bowtie$ varies

of $\bowtie_{\mathcal{B}}$,

$$\bowtie_{\mathcal{B}} = \begin{cases} (!\mathtt{t}_1,\, ?\mathtt{t}_2) & \text{if } \mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2 \\ (?\mathtt{t}_1,\, !\mathtt{t}_2) & \text{if } \mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1 \\ (!\mathtt{l},\, ?\mathtt{l}) & \text{if } \mathtt{l} \in \mathtt{L} \\ (?\mathtt{l},\, !\mathtt{l}) & \text{if } \mathtt{l} \in \mathtt{L} \\ (!(\sigma_1),\, ?(\sigma_2)) & \text{if } \sigma_1 \;\mathcal{B}\; \sigma_2 \\ (?(\sigma_1),\, !(\sigma_2)) & \text{if } \sigma_2 \;\mathcal{B}\; \sigma_1 \end{cases}$$

The definition of $\bowtie$, implies that the behaviour of a composition depends on the relation $\mathcal{B}$ given to $\bowtie$.

Example 7.2.2. [Contract interactions depend on $\mathcal{B}$]
Let $\rho = !(0).\,1 + ?(1).\,0$ and $\sigma = ?(0).\,0 + !(1).\,0$. In Figure 7.6 we show, for different binary relations $\mathcal{B}$, how the interactions of ρ and σ change. In particular, we show how the interactions in Figure 7.6 are inferred.

The contracts ρ and σ are stable, thus the rules [P-LEFT], and [P-RIGHT] can not be applied to the composition $\rho \,||\, \sigma$. To prove $\rho \,||\, \sigma \xrightarrow{\tau}$ we have to use rule [P-SYNCH].

If $\mathcal{B} = \emptyset$, then the side condition of rule [P-SYNCH] is false, because $!(0) \not\bowtie_{\emptyset} ?(0)$, $!(0) \not\bowtie_{\emptyset} !(1)$, $?(1).\,0 \not\bowtie_{\emptyset} ?(0)$, $?(1).\,0 \not\bowtie_{\emptyset} !(1)$, hence rule [P-SYNCH] can not be applied to $\rho \,||\, \sigma$ either. It follows that $\rho \,||\, \sigma \not\xrightarrow{\tau}_{\mathcal{B}}$.

If $\mathcal{B} = \{(0,\,0)\}$, then the definition of $\bowtie$ implies that $!(0) \bowtie_{\mathcal{B}} ?(0)$, thus we can infer

$$\frac{\rho \xrightarrow{!(0)}_{\mathcal{B}} 1 \quad \sigma \xrightarrow{?(0)}_{\mathcal{B}} 0}{\rho \,||\, \sigma \xrightarrow{\tau}_{\mathcal{B}} 1 \,||\, 0}\ \text{[P-SYNCH]}$$

If $\mathcal{B} = \{(0,\,0), (1,\,1)\}$, then we can again infer $\rho \,||\, \sigma \xrightarrow{\tau}_{\mathcal{B}} 1 \,||\, 0$; but now we also have $?(1) \bowtie_{\mathcal{B}} !(1)$, thus we can derive also

$$\frac{\rho \xrightarrow{?(1)}_{\mathcal{B}} 1 \quad \sigma \xrightarrow{!(1)}_{\mathcal{B}} 0}{\rho \,||\, \sigma \xrightarrow{\tau}_{\mathcal{B}} 0 \,||\, 0}\ \text{[P-SYNCH]}$$

□

Parameter of $\bowtie$	Dependent compliances
$\mathcal{B} = \emptyset$	$\rho \not\dashv_{\mathcal{B}} \sigma$ for every σ
$\mathcal{B} = \{(0, 0)\}$	$\rho \dashv_{\mathcal{B}} \sigma$
$\mathcal{B} = \{(0, 0), (1, 1)\}$	$\rho \not\dashv_{\mathcal{B}} \sigma$

Figure 7.7: The relations $\dashv_{\mathcal{B}}$ vary, as $\mathcal{B}$ does

Note that Lemma 2.3.7, Lemma 2.3.8, Lemma 2.3.9 and Lemma 2.3.10 do not depend on the rule [P-SYNCH], therefore they are true for every LTS parametrised on $\mathcal{B}$.

Recall the function $\mathcal{M}$ of Section 2.3.

To assign the operational semantics provided by the parametrised LTS to session types, we have to amend the definition of $\mathcal{M}$ so as to account for the higher-order terms. Let $\mathcal{M} : \mathsf{ST}_{\text{ho}} \longrightarrow \mathsf{SC}_{\text{HO}}$ be defined as follows,

$$\mathcal{M}(S) = \begin{cases} \vdots & \\ \text{cases in definition of } \mathcal{M}, & \\ !(\mathcal{M}(M)).\mathcal{M}(S) & \text{if } S = ![\,M\,]; S \text{ and } M \in \mathsf{ST}_{\text{ho}}, \\ ?(\mathcal{M}(M)).\mathcal{M}(S) & \text{if } S = ![\,M\,]; S \text{ and } M \in \mathsf{ST}_{\text{ho}} \end{cases}$$

The next proposition can be proven as we did in Section 2.3.

Proposition 7.2.3. The function $\mathcal{M}$ is a bijection.

7.2.1 Dependent compliance relations

As the semantics of parallel composition depends on a parameter $\mathcal{B}$, so do the transitions $\xrightarrow{\tau}$. The definition of compliance has to be genralised so as to mirror this dependency.

Definition 7.2.4. [$\mathcal{B}$-dependent compliance relation]
Let $\mathcal{F}^{\dashv} : \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \times \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{HO}})$ be the rule functional defined so that $(r, p) \in \mathcal{F}^{\dashv}(\mathcal{R}, \mathcal{B})$ whenever the following conditions hold:

(a) if $r \Downarrow$ then $p \Downarrow$

(b) if $r \mid\mid p \not\xrightarrow{\tau}_{\mathcal{B}}$ then $r \xrightarrow{\checkmark}_{\mathcal{B}}$

(c) if $r \mid\mid p \xrightarrow{\tau}_{\mathcal{B}} r' \mid\mid p'$ then $r' \mathcal{R} p'$

Fix a binary relation $\mathcal{B}$. If $X \subseteq \mathcal{F}^{\dashv}(X, \mathcal{B})$, then we say that X is a *co-inductive $\mathcal{B}$-compliance relation*. Lemma C.0.31 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\dashv}(X, \mathcal{B})$; we call this solution the *$\mathcal{B}$-compliance relation*, and we denote it $\dashv_{\mathcal{B}}$. That is $\dashv_{\mathcal{B}} = \nu X.\mathcal{F}^{\dashv}(X, \mathcal{B})$. If $r \dashv_{\mathcal{B}} p$ we say that the process r *$\mathcal{B}$-complies with* the process p. □

All the relations $\dashv_{\mathcal{B}}$ enjoy few properties that do not depend on the specific $\mathcal{B}$ at hand. We list these properties in the next proposition.

Lemma 7.2.5. [Properties of the dependent compliances]
For every binary relation $\mathcal{B} \subseteq \mathsf{SC}^2_{\text{HO}}$, the following statements are true:

i) if $\rho \dashv_{\mathcal{B}} \sigma_1$, $\rho \dashv_{\mathcal{B}} \sigma_2$ then $\rho \dashv_{\mathcal{B}} \sigma_1 \oplus \sigma_2$

$$\frac{\rho' \dashv^s_{\mathcal{B}} \sigma'}{\eta.\rho' \dashv^s_{\mathcal{B}} \theta.\sigma'}\ \eta \bowtie_{\mathcal{B}} \theta,\ \text{and } \eta \text{ contains no label;}\ [\text{R-ETA}]$$

Figure 7.8: Consider the rules in Figure 3.2, and replace [R-ALPHA] with [R-ETA].

ii) if $\rho_1 \dashv_{\mathcal{B}} \sigma$, $\rho_2 \dashv_{\mathcal{B}} \sigma$ then $\rho_1 \oplus \rho_2 \dashv_{\mathcal{B}} \sigma$

iii) $\rho \dashv_{\mathcal{B}} \sigma$ if and only if $\text{UNF}(\rho) \dashv_{\mathcal{B}} \text{UNF}(\sigma)$

Proof. The proofs are almost identical to the ones of Corollary 3.2.9, Proposition 3.2.10, and Lemma 3.2.8. □

Syntactic compliance

The restrictive syntax of session types let us give a syntactic oriented characterisation of the compliance relation (Lemma 3.3.10). This characterisation relies on a co-inductive relation (Definition 3.3.1), that we define now.

Definition 7.2.6. [$\mathcal{B}$-syntactic compliance relation]
Let $\mathcal{F}^{\dashv^s} : \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \times \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{HO}})$ be the rule functional given by the inference rules in Figure 7.8. Fix a relation $\mathcal{B}$. If $X \subseteq \mathcal{F}^{\dashv^s}(X, \mathcal{B})$, then we say that X is a *co-inductive $\mathcal{B}$-syntactic compliance relation*. Lemma C.0.32 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\dashv^s}(X, \mathcal{B})$; we call this solution the *$\mathcal{B}$-syntactic compliance relation*, and we denote it $\dashv^s_{\mathcal{B}}$. That is $\dashv^s_{\mathcal{B}} = \nu X.\mathcal{F}^{\dashv^s}(X, \mathcal{B})$. □

We have the generalisation of Lemma 3.3.3.

Lemma 7.2.7. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, and $\mathcal{R}$-syntactic compliance relation, $\rho\ \mathcal{R}\ \sigma$ if and only if $\text{UNF}(\rho)\ \mathcal{R}\ \text{UNF}(\sigma)$.

Lemma 7.2.8. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, if $\mathcal{R}$ is a co-inductive $\mathcal{B}$-compliance relation, then $\mathcal{R}$ is a co-inductive $\mathcal{B}$-syntactic compliance relation.

Proof. The proof of this lemma proceeds as the proof of Lemma 3.3.4. The only difference is the discussion of point (ii), in which the application of rule [R-ALPHA], must be replaced with rule [R-ETA]. We discuss this case.

ii) Suppose that $\rho = \eta.\rho'$ with η containing no labels. We prove that we can derive (ρ, σ) by using rule [R-ETA].

Plainly $\rho \overset{\checkmark}{\not\longrightarrow}$, thus $(\rho, \sigma) \in \mathcal{F}^{\dashv}(\mathcal{R}, \mathcal{B})$ can have been proved only by applying rule [R-ETA] of Figure 7.8. This means that ρ is not stuck together with σ.

Since $\rho \overset{\tau}{\not\longrightarrow}_{\mathcal{B}}$, either ρ can interact with σ, or $\sigma \xrightarrow{\tau}_{\mathcal{B}}$.

As $\sigma \Longrightarrow_{\mathcal{B}} \hat{\sigma} \overset{\tau}{\not\longrightarrow}$ implies that ρ must interact with $\hat{\sigma}$, the restrictive syntax of session types implies that $\hat{\sigma} = \theta.\sigma'$ and that $\eta \bowtie_{\mathcal{B}} \theta$. The last fact ensures that θ contains no labels; thanks to the syntax of session contracts, we can prove that $\sigma = \hat{\sigma} = \theta.\sigma'$.

Note now that $\rho \mid\mid \sigma \xrightarrow{\tau}_{\mathcal{B}} \rho' \mid\mid \sigma'$, thus Definition 7.2.4 ensures that $\rho'\ \mathcal{R}\ \sigma'$.

We can now derive

$$\frac{\rho' \dashv^s_{\mathcal{B}} \sigma'}{\rho \dashv^s_{\mathcal{B}} \sigma}\ \eta \bowtie_{\mathcal{B}} \theta;\ [\text{E-ETA}]$$

□

The proof of Lemma 3.3.6 does not depend on rule [R-Alpha], so that lemma is true for the LTS of every $\mathcal{B}$. The consequence is the next corollary.

Corollary 7.2.9. *For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, if $\mathcal{R}$ is a co-inductive $\mathcal{B}$-syntactic compliance relation, then following statements hold:*

(a) if $\rho \xrightarrow{\tau}_{\mathcal{B}} \rho'$ then $\rho' \dashv^s_{\mathcal{B}} \sigma$

(b) if $\sigma \xrightarrow{\tau}_{\mathcal{B}} \sigma'$ then $\rho \dashv^s_{\mathcal{B}} \sigma'$

Proof. We prove (a). The argument is a generalisation of the proof of Corollary 3.3.7, in that we account for the parameter $\mathcal{B}$.

$$
\begin{aligned}
\dashv^s_{\mathcal{B}} &\subseteq \mathcal{F}^{\dashv^s}(\dashv^s_{\mathcal{B}}, \mathcal{B}) && \text{By definition}\\
\{(\rho', \sigma)\} \cup \dashv^s_{\mathcal{B}} &\subseteq \mathcal{F}^{\dashv^s}(\{(\rho', \sigma)\} \cup \dashv^s_{\mathcal{B}}, \mathcal{B}) && \text{By Lemma 3.3.6, and definition of SD}\\
\{(\rho', \sigma)\} \cup \dashv^s_{\mathcal{B}} &\subseteq \nu X.\mathcal{F}^{\dashv^s}(X, \mathcal{B}) && \text{By the Knaster-Tarski theorem}\\
&= \dashv^s_{\mathcal{B}} && \text{By definition}
\end{aligned}
$$

From the argument above, it follows that $\rho' \dashv^s_{\mathcal{B}} \sigma$. The proof of (b) is similar. □

We generalise Lemma 3.3.8.

Lemma 7.2.10. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, and every $\mathcal{R}$ such that $\mathcal{R} \subseteq \mathcal{F}^{\dashv^s}(\mathcal{R}, \mathcal{B})$, if $\rho \,\mathcal{R}\, \sigma$ and $\rho \,||\, \sigma \overset{\tau}{\not\longrightarrow}$ then $\rho \xrightarrow{\checkmark}_{\mathcal{B}}$.

Proof. It is enough to use the proof of Lemma 3.3.8, but replace [R-Alpha] with [R-Eta]. □

Lemma 7.2.11. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, the relation $\dashv^s_{\mathcal{B}}$ is a co-inductive $\mathcal{B}$-compliance relation. Formally, $\dashv^s_{\mathcal{B}} \subseteq \mathcal{F}^{\dashv}(\dashv^s_{\mathcal{B}}, \mathcal{B})$

Proof. The argument is similar to the proof of Lemma 3.3.9. The only difference appears in point (i), where we have to use the more general [R-Eta] in place of [R-Alpha]. We give the details of that case.

i) If $\rho = \eta.\rho'$ and η contains no labels, then $(\rho, \sigma) \in \mathcal{F}^{\dashv^s}_{\text{MUST}^s}(\dashv^s)$ must be proven by the derivation

$$\frac{\rho' \dashv^s_{\mathcal{B}} \sigma'}{\eta.\rho' \dashv_{\mathcal{B}} \theta.\sigma'}\ \eta \bowtie_{\mathcal{B}} \theta;\ \text{[R-Eta]}$$

The premises of the rule ensure that $\rho' \dashv \sigma'$.

□

Lemma 7.2.12. [Syntactic characterisation $\mathcal{B}$-compliance relation]
For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, $\rho \dashv_{\mathcal{B}} \sigma$ if and only if $\rho \dashv^s_{\mathcal{B}} \sigma$.

Proof. The result follows from Lemma 7.2.8 and Lemma 7.2.11. The argument is analogous to the proof of Lemma 3.3.10. □

Proposition 7.2.13. If $\mathcal{B} \subseteq \mathcal{B}'$, then $\mathcal{F}^{\dashv^s}(\mathcal{R}, \mathcal{B}) \subseteq \mathcal{F}^{\dashv^s}(\mathcal{R}, \mathcal{B}')$.

7.2.2 Dependent duality

In this section our aim is twofold; we want (a) to understand which conditions are necessary to (b) generalise Lemma 6.5.6 to the LTS parametrised on $\mathcal{B}$. In general this is not possible, there exists relations $\mathcal{B}$ and higher-order session contracts ρ such that $\rho \not\dashv_{\mathcal{B}} \sigma$ for every $\sigma \in \mathsf{SC}_{\text{HO}}$.

We begin by extending the notion of dual, and defining two properties of binary relations.

$$\frac{\rho' \text{ DUAL } \sigma'}{?(\hat{\rho}).\rho' \text{ DUAL } !(\hat{\sigma}).\sigma'} \ \hat{\rho}\ \mathcal{B}\ \hat{\sigma}; [\text{R-IN-H}]$$

$$\frac{\rho' \text{ DUAL } \sigma'}{?(\hat{\rho}).\rho' \text{ DUAL } !(\hat{\sigma}).\sigma'} \ \hat{\sigma}\ \mathcal{B}\ \hat{\rho}; [\text{R-OUT-H}]$$

Figure 7.9: $\mathcal{B}$-Inference rules for the rule functional $\mathcal{F}_{\text{DUAL}}$. Take the rules in Figure 6.2, and add the rules above

Definition 7.2.14. [$\mathcal{B}$-dual session contract]
If $\rho \dashv\!\vdash^{s}_{\mathcal{B}} \sigma$ then we say that σ is a $\mathcal{B}$-*dual* of ρ. □

Definition 7.2.15. [Strongly total]
We say that a relation $\mathcal{R} \subseteq A \times A$ is *total* if for every $a \in A$ there exists a $a' \in A$ such that $a\ \mathcal{R}\ a'$. We say that $\mathcal{R}$ is *strongly total* if and only if for every $a \in A$ there exists $a', a'' \in A$ such that $a\ \mathcal{R}\ a'$, and $a''\ \mathcal{R}\ a$; that is if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are total. □

We are ready to define a functional that, under suitable hypothesis, will let us prove that every session contract has a $\mathcal{B}$-dual.

Definition 7.2.16. [$\mathcal{B}$-dual session contracts]
Let $\mathcal{F}_{\text{DUAL}} : \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \times \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{HO}})$ be the rule functional given by the inference rules in Figure 7.9. Fix a relation $\mathcal{B}$. Lemma C.0.33 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_{\text{DUAL}}(X, \mathcal{B})$; we call this solution the $\mathcal{B}$-*duality relation*, and we denote it $\text{DUAL}(\mathcal{B})$: That is $\text{DUAL}(\mathcal{B}) = \mu X.\mathcal{F}_{\text{DUAL}}(X, \mathcal{B})$. □

All the properties of the function DUAL that studied in Section 6.5. Most of them do not depend on the parameter $\mathcal{B}$ at all.

The next proposition shows under which hypothesis $\text{DUAL}(\mathcal{B})$ is a total function.

Lemma 7.2.17. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, if $\mathcal{B}$ is strongly total then $\text{DUAL}(\mathcal{B})$ is total.

Proof. The proof is analogous to the proof of point (b) in Lemma 6.5.6. Since $\mathcal{B}$ is strongly total we know that if we need to apply [R-IN-H] or [R-OUT-H], the side conditions are true. □

Lemma 7.2.18. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, the relation $\text{DUAL}(\mathcal{B})$ is a co-inductive $\mathcal{B}$-compliance relation.

Proof. Similar to the proof of Lemma 6.5.11. □

Corollary 7.2.19. *If $\mathcal{R}$ is a reflexive relation, then* $\text{DUAL}(\mathcal{R})$ *is a total $\mathcal{R}$-compliance relation.*

Proof. It follows from Lemma 7.2.17, Lemma 7.2.18 and the fact that a reflexive relation is strongly total. □

7.3 Transitive closures

In the next chapter we will use the the transitive closures of binary relations on session contracts. We explain here how to build inductive transitive closures.

Let $\mathcal{F}_{+} : \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \times \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{HO}})$ be the rule functional given by the inference rules in Figure 7.10. Proposition C.0.37 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_{+}(X, \mathcal{B})$; we call this solution the *transitive closure of* $\mathcal{B}$, and we denote it $[\mathcal{B}]^{+}$: That is $[\mathcal{B}]^{+} = \mu X.\mathcal{F}_{+}(X, \mathcal{B})$. □

$$\frac{}{a\ [\mathcal{B}]^+\ b}\ a\ \mathcal{B}\ b;\ [\text{TRC-A}] \qquad \frac{a\ [B]^+\ b \quad b\ [\mathcal{B}]^+\ c}{a\ [\mathcal{B}]^+\ c}\ [\text{TRC-R}]$$

Figure 7.10: Inference rule for the rule functional $\mathcal{F}_+$

Example 7.3.1. In this example we justify the use of a fixed point in the definition of transitive closure. In particular, we show that the operation adding to a relation the pairs necessary to prove its transitivity has to be iterated.

Consider the relations

$$T_1 = \{(?\mathtt{l}_1.\,1,\,0), (1, ?\mathtt{l}_1.\,1)\}$$
$$T_2 = \{(0,\,!\mathtt{l}_2.\,1)\}$$

If we add to the relation $T_1 \cup T_2$ all the pairs according to rule [TRC-R], then we get

$$T_1 \cup T_2 \cup \{(?\mathtt{l}_1.\,1,\,!\mathtt{l}_2.\,1), (1, 0)\}$$

Note, though, that this relation is not transitive, as

$$\{(1, ?\mathtt{l}_1.\,1), (?\mathtt{l}_1.\,1,\,!\mathtt{l}_2.\,1)\} \subseteq T_1 \cup T_2 \cup \{(?\mathtt{l}_1.\,1,\,!\mathtt{l}_2.\,1)\}$$

while $(1,\,!\mathtt{l}_2.\,1) \notin T_1 \cup T_2 \cup \{(?\mathtt{l}_1.\,1,\,!\mathtt{l}_2.\,1)\}$.

On the other hand, we can prove that the relation $[\mathcal{B}]^+$ is

$$T_1 \cup T_2 \cup \{(?\mathtt{l}_1.\,1,\,!\mathtt{l}_2.\,1), (1,\,0), (1,\,!\mathtt{l}_2.\,1)\}$$

and indeed it is transitive.

Lemma 7.3.2. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, the transitive closure of $\mathcal{B}$ contains $\mathcal{B}$: $\mathcal{B} \subseteq [\mathcal{B}]^+$.

Lemma 7.3.3. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$, the relation $[\mathcal{B}]^+$ is transitive.

Proof. Let $a\ [\mathcal{B}]^+\ b$ and $b\ [\mathcal{B}]^+\ c$; we have to prove that $a\ [\mathcal{B}]^+\ c$.

The hypothesis that $a\ [\mathcal{B}]^+\ b$ and $b\ [\mathcal{B}]^+\ c$ ensure that we have the finite inference trees

$$\frac{\vdots}{a\ [\mathcal{B}]^+\ b} \qquad \frac{\vdots}{b\ [\mathcal{B}]^+\ c}$$

We can now apply rule [TRC-R] to obtain the finite inference tree

$$\frac{\dfrac{\vdots}{a\ [\mathcal{B}]^+\ b} \quad \dfrac{\vdots}{b\ [\mathcal{B}]^+\ c}}{a\ [\mathcal{B}]^+\ c}\ [\text{TRC-R}]$$

This is enough to prove that $a\ [\mathcal{B}]^+\ c$. □

7.4 Related Work

The higher-order session types that we introduced in Definition 7.1.1 in the literature are know just as “session types”, and they have been introduced by [Takeuchi et al., 1994].

We comment on the distinction between first-order and higher-order session types that we introduced. The purpose of the distinction was to ease the transition of our reasoning from processes to session types. The labels in the transitions of processes are actions of $\mathsf{Act}_{\tau\checkmark}$, so to move from from processes to first-order session contracts does not require a complete change in our reasoning. After having established the full abstraction result (Theorem 6.3.4) in the first-order setting, we moved to the higher-order setting, that is the notion of session type used throughout the literature.

The definition of the sub-typing à la Gay and Hole differs from our Definition 7.1.2. In particular, in Figure 2.6 we have inference rules to explicitly (un)fold terms, and these rules have side conditions. Definition 4 of Gay and Hole, on the other hand, is given by case analysis on the unfoldings of types. Lemma 7.1.6 shows that the difference has no impact in the relation obtained. The first presentation of sub-typing for session types was put forth also by Gay and Hole in 1999, and is algorithmic. That formulation is given by using inference rules (see Figure 1 of that paper), and the (un)folding of terms is treated by two rules, namely [AS-Rec-L] and [AS-Rec-R]. The role played by the side conditions of our rule [r-Fold!!!!!!!!] and [r-Unfold], there is played by the type environment and the form of [AS-Rec-L], [AS-Rec-R]. Also, the way in which unfoldings are treated by the mentioned rules is reminiscent of the rule [Coind] work by [Brandt and Henglein, 1998, see Figure 9].

Chapter 8

Modelling higher-order session-types

Higher-order session contracts allow us to easily assign an operational semantics to higher-order session types, but they come at a cost. We had to parametrise the transitions of session contracts on binary relations $\mathcal{B}$, thereby obtaining an infinite amount of LTSs.

In this chapter we purse two aims. First, we study the client and the server pre-orders on the LTSs

$$\langle\, \mathsf{SC}_{\text{HO}},\ \mathsf{Act}_{\tau\checkmark} \cup \mathsf{SC}_{\text{HO}} \ \longrightarrow_{\mathcal{B}} \,\rangle$$

We do this in two steps, first we parametrise the relations $\precsim^{syn}_{\text{SVR}}$ and $\preceq^{syn}_{\text{CLT}}$ over the relations $\mathcal{B}$, and introduce the *dependent pre-order*

$$\sqsubseteq^{\mathcal{B}}_{\text{SVR}}, \qquad \sqsubseteq^{\mathsf{ho}}_{\text{P2P}}\ \mathcal{B}$$

Then we show that as long as the parameter $\mathcal{B}$ is a pre-order on session contracts, Proposition 6.4.7 and Proposition 6.5.18 extend to the higher-order setting,

$$\sqsubseteq^{\mathcal{B}}_{\text{SVR}} = \precsim^{\mathcal{B}}_{\text{SVR}}, \qquad \sqsubseteq^{\mathcal{B}}_{\text{CLT}} = \preceq^{\mathcal{B}}_{\text{CLT}}$$

Pre-orders on higher-order session contracts will be so important that we will denote the their set with the symbol $\textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$.

Our investigation, of the dependent pre-orders, lets us cut the Gordian knot represented by the parameter $\mathcal{B}$. At the end of Section 8.1 and of Section 8.2, we prove that if we restrict our attention the the pre-orders on SC_{HO}, then the functions $\lambda X.\ \sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ and $\lambda X.\ \sqsubseteq^{X}_{\text{SVR}}$ are *monotone endofunctions* , so the Knaster-Tarski theorem ensure that their greatest fixed points exist; we denote them respectively

$$\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}, \qquad \sqsubseteq^{\mathsf{ho}}_{\text{SVR}} \tag{8.1}$$

The pre-orders in Eq. (8.1) generalise the pre-orders $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ and $\sqsubseteq^{\mathsf{fo}}_{\text{SVR}}$, and let us generalise Eq. (7.1). In Theorem 8.4.9) we show the following isomorphism,

$$\sqsubseteq^{\mathsf{ho}}_{\text{CLT}} \cap \sqsubseteq^{\mathsf{ho}}_{\text{SVR}} \cong \preccurlyeq_{\mathsf{sbt}}$$

Structure of the chapter. In Section 8.1 we study the dependent client pre-orders, and we show the conditions required to extend the syntactic characterisation of $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ to the new setting. We also prove the results that ultimately lead to the proof that $\lambda X.\ \sqsubseteq^{X}_{\text{CLT}}$ is monotone if $X \in \textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$. Section 8.2 follows the structure of Section 8.1 and we state the results about the dependent server pre-orders without proving them. In Section 8.4 we adapt the proofs of Section 6.3 to prove that the

full abstraction result holds true also in the higher-order setting.

8.1 Client pre-orders

In this section we generalise the results of Section 6.5 to the dependent LTSs. Also in this context we study when the client ρ_2 is better, with respect to the dependent compliance $\dashv_{\mathcal{B}}$, than a client ρ_1.

Following this intuition, in this section we define a *family* of binary relations on session contracts (Definition 8.1.1), the dependent client pre-orders:

$$\{\sqsubseteq^{\mathcal{B}}_{\text{CLT}}\}_{\mathcal{B}\in\mathcal{P}(\mathsf{SC}^2_{\text{HO}})}$$

We will be concerned, at first, with the properties of these pre-orders; and then with their syntactic characterisation (Proposition 8.1.11). As we will see, the results that we will accumulate to prove Proposition 8.1.11, imply also that the abstraction $\lambda X.\ \sqsubseteq^{X}_{\text{CLT}}$ is monotone as long as X is a reflexive and transitive relation. This fact will be crucial later on (see Section 8.4).

Definition 8.1.1. [$\mathcal{B}$-dependent client pre-order]
Given a binary relation on session contracts $\mathcal{B} \subseteq \mathsf{SC}^2_{\text{HO}}$, we write $\rho_1 \sqsubseteq^{\mathcal{B}}_{\text{CLT}} \rho_2$ whenever $\rho_1 \dashv_{\mathcal{B}} \sigma$ implies that $\rho_2 \dashv_{\mathcal{B}} \sigma$ for every $\sigma \in \mathsf{SC}_{\text{HO}}$. We refer to the symbol $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ as the the *$\mathcal{B}$-dependent client pre-order.* □

We call the relations $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ "dependent", because in general their properties depend on the parameter $\mathcal{B}$. We give an example of this fact.

Example 8.1.2. Let $\mathcal{B} = \{(\hat{\rho}, \hat{\rho})\}$. We prove that

a) $?(\hat{\rho}).\,\mathtt{1} \sqsubseteq^{\emptyset}_{\text{CLT}} !\mathtt{1}.\,\mathtt{1}$

b) $?(\hat{\rho}).\,\mathtt{1} \not\sqsubseteq^{\mathcal{B}}_{\text{CLT}} !\mathtt{1}.\,\mathtt{1}$

To show point (a), we prove that $?(\hat{\rho}).\,\mathtt{1} \not\dashv_{\emptyset} \sigma$ for every stable σ. Let σ be a stable session contract. Plainly $?(\hat{\rho}).\,\mathtt{1} \parallel \sigma \overset{\tau}{\not\longrightarrow}_{\emptyset}$, because $\hat{\rho} \not\bowtie_{\emptyset} \sigma$, and $?(\hat{\rho}).\,\mathtt{1} \overset{\checkmark}{\not\longrightarrow}_{\emptyset}$, so $?(\hat{\rho}).\,\mathtt{1} \not\dashv_{\emptyset} \sigma$. It follows that $?(\hat{\rho}).\,\mathtt{1} \sqsubseteq^{\emptyset}_{\text{CLT}} !\mathtt{1}.\,\mathtt{1}$.

For every action θ we have that $?(\hat{\rho}) \not\bowtie_{\emptyset} \theta$; this let us prove that $\rho \parallel \sigma \overset{\tau}{\not\longrightarrow}_{\emptyset}$; at the same time, $?(\hat{\rho}).\,\mathtt{1} \overset{\checkmark}{\not\longrightarrow}_{\emptyset}$, so condition (b) of Definition 7.2.4 lets us prove that $?(\hat{\rho}).\,\mathtt{1} \not\dashv_{\emptyset} \sigma$. Since we have no particular assumption on σ, other than its being stable, we have proven that $?(\hat{\rho}).\,\mathtt{1} \not\dashv_{\emptyset} \sigma$ for every stable session contract σ. This can be used to prove the following equality,

$$\{\,\sigma \in \mathsf{SC}_{\text{HO}} \mid ?(\hat{\rho}).\,\mathtt{1} \dashv_{\emptyset} \sigma\,\} = \emptyset$$

The equality above ensures that $?(\hat{\rho}).\,\mathtt{1} \sqsubseteq^{\emptyset}_{\text{CLT}} \sigma$ is trivially true for every session contract σ. It follows that $?(\hat{\rho}).\,\mathtt{1} \sqsubseteq^{\emptyset}_{\text{CLT}} !\mathtt{1}.\,\mathtt{1}$, so point (a) is proven.

Now we show point (b). Let $\mathcal{R} = \{\,(?(\hat{\rho}).\,\mathtt{1},\ !(\hat{\rho}).\,\mathtt{1}),\ (\mathtt{1},\ \mathtt{1})\,\}$. The relation $\mathcal{R}$ is a co-inductive $\mathcal{B}$-syntactic compliance relation (that is $\mathcal{R} \subseteq \mathcal{F}_{\dashv^s}(\mathcal{R}, \mathcal{B})$); the derivation that proves this is

$$\dfrac{\dfrac{}{\mathtt{1} \dashv^s_{\mathcal{B}} \mathtt{1}}\ [\text{A-Unit}]}{?(\hat{\rho}).\,\mathtt{1} \dashv^s_{\mathcal{B}} !(\hat{\rho}).\,\mathtt{1}}\ ?(\hat{\rho}) \bowtie_{\mathcal{B}} !(\hat{\rho});\ [\text{R-Eta}]$$

Lemma 3.3.10 guarantees that $?(\hat{\rho}).\,\mathtt{1} \dashv_{\mathcal{B}} !(\hat{\rho}).\sigma$.

We prove that $!\mathtt{1}.\,\mathtt{1} \not\dashv_{\mathcal{B}} !(\hat{\rho}).\,\mathtt{1}$; this is true because $!\mathtt{1} \not\bowtie_{\mathcal{B}} !(\hat{\rho})$, so $!\mathtt{1}.\,\mathtt{1} \parallel !(\hat{\rho}).\,\mathtt{1} \overset{\tau}{\not\longrightarrow}$, and $!\mathtt{1}.\,\mathtt{1} \overset{\checkmark}{\not\longrightarrow}_{\mathcal{B}}$.
□

Even though, in general, the properties of $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ depend on $\mathcal{B}$, all the pre-orders $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ enjoy some properties that do not depend on the $\mathcal{B}$'s. This is a consequence of the properties of the dependant compliance relations which do not depend on any $\mathcal{B}$'s (Lemma 7.2.5).

For instance, we generalise Corollary 6.5.3.

Corollary 8.1.3 (Top elements)**.**
Let $\mathcal{B}$ be a binary relation on session contracts.

(a) the term $\mathbf{1}$ is a top element of $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$

(b) if $\mathbf{1} \sqsubseteq^{\mathcal{B}}_{\text{CLT}} \rho_2$ then $\text{UNF}(\rho_2) = \mathbf{1}$.

Proof. The proof of this corollary is similar to the proof of Corollary 6.5.3, but relies on point (iii) of Lemma 7.2.5. □

The corollary above ensures that for every $\mathcal{B}$, the relation $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ is not a sound model of $\preccurlyeq_{\mathtt{sbt}}$. The problem is still the one exhibited in Example 6.2.7.

Corollary 8.1.4. $\rho_1 \sqsubseteq^{\mathcal{B}}_{\text{CLT}} \rho_2$ *if and only* $\text{UNF}(\rho_1) \sqsubseteq^{\mathcal{B}}_{\text{CLT}} \text{UNF}(\rho_2)$.

Under suitable hypothesis on $\mathcal{B}$, we can give a syntactic characterisation of $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$.

Lemma 8.1.5. Let $\mathcal{R}$ be a reflexive relation on session contracts, and let $\rho_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_2$. One of the following is true:

(i) $\text{UNF}(\rho_2) = \mathsf{1}$

(ii) $\text{UNF}(\rho_2) = !(\hat{\rho}_2).\rho'_2$, $\text{UNF}(\rho_1) = !(\hat{\rho}_1).\rho'_1$, $\hat{\rho}_2 \mathrel{\mathcal{R}} \hat{\rho}_1$, and $\rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho'_2$

(iii) $\text{UNF}(\rho_2) = ?(\hat{\rho}_2).\rho'_2$, $\text{UNF}(\rho_1) = ?(\hat{\rho}_1).\rho'_1$, $\hat{\rho}_1 \mathrel{\mathcal{R}} \hat{\rho}_2$, and $\rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho'_2$

(iv) if $\text{UNF}(\rho_2) = !\mathtt{t}_2.\rho'_2$ then $\text{UNF}(\rho_1) = !\mathtt{t}_1.\rho'_1$, $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $\rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho'_2$

(v) if $\text{UNF}(\rho_2) = ?\mathtt{t}_2.\rho'_2$ then $\text{UNF}(\rho_1) = ?\mathtt{t}_1.\rho'_1$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and $\rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho'_2$

(vi) if $\text{UNF}(\rho_2) = \sum_{j\in J} ?\mathtt{l}_j.\rho^2_j$ then $\text{UNF}(\rho_1) = \sum_{i\in I} ?\mathtt{l}_i.\rho^1_i$ with $I \subseteq J$ and $\rho^1_i \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho^2_i$

(vii) if $\text{UNF}(\rho_2) = \bigoplus_{j\in J} !\mathtt{l}_j.\rho^2_j$ then $\text{UNF}(\rho_1) = \bigoplus_{i\in I} !\mathtt{l}_i.\rho^1_i$ with $J \subseteq I$ and $\rho^1_j \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho^2_j$

Proof. The proof of this lemma is similar to the proof of Lemma 6.5.12. We have to discuss only that cases that involve higher-order terms, that is point (ii) and point (iii).

We prove that if $\rho_2 = !(\hat{\rho}_2).\rho'_2$ then point (ii) is true. We prove the following facts:

(b.1) if $\rho_2 = !(\hat{\rho}_2).\rho'_2$, then $\rho_1 = !(\hat{\rho}_1).\rho'_1$

(b.2) if $\rho_1 = !(\hat{\rho}_1).\rho'_1$, then $\rho_2 = !(\hat{\rho}_2).\rho'_2$, $\hat{\rho}_2 \mathrel{\mathcal{R}} \hat{\rho}_1$, and $\rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho'_2$

We give the proofs in order. Suppose $\rho_2 = !(\hat{\rho}_2).\rho'_2$ and let $\rho_1 \dashv_{\mathcal{R}} \sigma$; thanks to Corollary 7.2.19 and the reflexivity of $\mathcal{R}$ we know that there exist such a σ (ie. $\overline{\rho_1}$).

The hypothesis $\rho_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_2$ ensures that $\rho_2 \dashv_{\mathcal{R}} \sigma$. The last fact, given the form of ρ_2, can be proved only by using rule [R-Eta], thus the premises must be true: $\sigma = \theta.\sigma'$, $\hat{\rho}_2 \bowtie_{\mathcal{R}} \theta$, and $\rho'_2 \dashv_{\mathcal{R}} \sigma'$. The assumption $\rho_1 \dashv_{\mathcal{R}} \sigma$ now implies that $\rho_1 = !(\hat{\rho}_1).\rho'_1$. We have proven (1), and now we prove (2).

Let $\rho_1 = !(\hat{\rho}_1).\rho'_1$; thanks to Corollary 7.2.19 and the reflexivity of $\mathcal{R}$ we know that there exist a σ' such that $\rho_1 \dashv^s_{\mathcal{R}} \sigma'$ (ie. $\overline{\rho_1}$). As $\mathcal{R}$ is reflexive, we can prove that $!(\hat{\rho}_1) \bowtie_{\mathcal{B}} ?(\hat{\rho}_1)$, thus we can prove that $\rho_1 \dashv^s_{\mathcal{R}} ?(\hat{\rho}_1).\sigma'$. The hypothesis implies that $\rho_2 \dashv^s_{\mathcal{R}} ?(\hat{\rho}_1).\sigma'$, and this can be proven only by using rule [R-Eta]; thus it must be $\rho_2 = \eta_2.\rho'_2$, $\eta_2 = !(\hat{\rho}_2)$, $\hat{\rho}_2 \mathrel{\mathcal{R}} \hat{\rho}_1$, and $\rho'_2 \dashv^s_{\mathcal{R}} \sigma'$; as there is no particular assumption on σ', the last fact implies that $\rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho'_2$.

The proof that point (iii) is true is similar to the argument above. □

$$\frac{\rho_1' \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2'}{!(\hat{\rho}_1).\rho_1' \preceq^{\mathcal{B}}_{\text{CLT}} !(\hat{\rho}_2).\rho_2'}\ \hat{\rho}_2\ \mathcal{B}\ \hat{\rho}_1;\ [\text{R-OUT-H}]$$

$$\frac{\rho_1' \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2'}{?(\hat{\rho}_1).\rho_1' \preceq^{\mathcal{B}}_{\text{CLT}} ?(\hat{\rho}_2).\rho_2'}\ \hat{\rho}_1\ \mathcal{B}\ \hat{\rho}_2;\ [\text{R-IN-H}]$$

Figure 8.1: Additional inference rules for the rule functional $\mathcal{F}^{\preceq_{\text{CLT}}}$. The other rules are in Figure 6.3

In Lemma 8.1.5, the hypothesis of $\mathcal{R}$ being reflexive is necessary. We explain why in Example B.0.6.

Definition 8.1.6. [$\mathcal{B}$-syntactic client pre-order]
Let $\mathcal{F}^{\preceq_{\text{CLT}}} : \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \times \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \mathsf{SC}^2_{\text{HO}}$ be the rule functional given by the inference rules in Figure 8.1. Fix a binary relation $\mathcal{B}$. If $X \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(X, \mathcal{B})$, then we say that X is a *co-inductive $\mathcal{B}$-syntactic client pre-order*. Lemma C.0.35 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\preceq_{\text{CLT}}}(X, \mathcal{B})$; we call this solution the *$\mathcal{B}$-syntactic client pre-order*, and we denote it $\preceq^{\mathcal{B}}_{\text{CLT}}$. That is $\preceq^{\mathcal{B}}_{\text{CLT}} = \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, \mathcal{B})$. □

We can reason on the pre-orders $\preceq^{\mathcal{B}}_{\text{CLT}}$ up-to unfolding.

Lemma 8.1.7. For every $\mathcal{B} \subseteq \mathsf{SC}_{\text{HO}}$ and $\rho_1, \rho_2 \in \mathsf{SC}_{\text{HO}}$, $\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2$ if and only if $\text{UNF}(\rho_1) \preceq^{\mathcal{B}}_{\text{CLT}} \text{UNF}(\rho_2)$.

Proof. Analogous to the proof of Lemma 2.1.16. □

The next result is crucial for our aims.

Proposition 8.1.8. The rule functional $\mathcal{F}^{\preceq_{\text{CLT}}}$ is monotone in its second variable.

Proof. Let $\mathcal{R} \subseteq \mathcal{R}'$ and $\mathcal{S}, \mathcal{R}, \mathcal{R}' \subseteq \mathsf{SC}_{\text{HO}} \times \mathsf{SC}_{\text{HO}}$. We have to prove the following set inclusion

$$\mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{S}, \mathcal{R}) \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{S}, \mathcal{R}')$$

Element-wise, we are required to prove that if $(\rho_1, \rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{S}, \mathcal{R})$, then $(\rho_1, \rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{S}, \mathcal{R}')$.

Suppose that $(\rho_1, \rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{S}, \mathcal{R})$, this means that there exists a one step derivation

$$\frac{\cdots}{\rho_1 \preceq^{\mathcal{R}'}_{\text{CLT}} \rho_2}$$

generated by instantiating one of the rules in Figure 8.1. Note that if the side conditions and the premises of the rule used do not depend on $\mathcal{R}$ (this the case if the rule used is, for instance, [R-CHOICE] or [R-IN-FO]), then the derivation above proves that $(\rho_1, \rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{S}, \mathcal{R}')$. Suppose now that the derivation is due to rule [R-OUT-H] or [R-IN-H]; these are the only two rules whose premises and side conditions depend on the second variable of $\mathcal{F}^{\preceq_{\text{CLT}}}$. The derivation above must have the form

$$\frac{\rho_1 \preceq^{\mathcal{R}}_{\text{CLT}} \rho_2}{!(\hat{\rho}_1).\rho_1 \preceq^{\mathcal{R}}_{\text{CLT}} !(\hat{\rho}_2).\rho_2}\ \hat{\rho}_2\ \mathcal{R}\ \hat{\rho}_1;\ [\text{R-OUT-HO}]$$

The hypothesis that $\mathcal{R} \subseteq \mathcal{R}'$ and the fact that $\rho_2\ \mathcal{R}\ \hat{\rho}_1$ ensure that $\rho_2\ \mathcal{R}'\ \hat{\rho}_1$, thus we can derive

$$\frac{\rho_1 \preceq^{\mathcal{R}'}_{\text{CLT}} \rho_2}{!(\hat{\rho}_1).\rho_1 \preceq^{\mathcal{R}'}_{\text{CLT}} !(\hat{\rho}_2).\rho_2}\ \hat{\rho}_2\ \mathcal{R}'\ \hat{\rho}_1;\ [\text{R-OUT-HO}]$$

This derivation proves that $(\rho_1, \rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{S}, \mathcal{R}')$. The argument for rule [R-IN-HO] is similar. □

The proof of the previous proposition ultimately relies on the restrictive syntax of session contracts, and it is relatively easy to see that if we parametrise the more generous LTS of processes, then the foregoing proposition is not true.

Corollary 8.1.9. *Let $\mathcal{B}, \mathcal{B}'$ be two binary relations on session contracts. If $\mathcal{B} \subseteq \mathcal{B}'$ then $\preceq^{\mathcal{B}}_{\text{CLT}} \subseteq \preceq^{\mathcal{B}'}_{\text{CLT}}$.*

Proof. Our definitions imply two equalities,

$$\begin{aligned} \preceq^{\mathcal{B}}_{\text{CLT}} &= \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, \mathcal{B}) && \text{By definition} \\ &= \mathcal{F}^{\preceq_{\text{CLT}}}(\preceq^{\mathcal{B}}_{\text{CLT}}, \mathcal{B}) && \text{By definition of fixed point} \\ &\subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\preceq^{\mathcal{B}}_{\text{CLT}}, \mathcal{B}) && \text{By weakening} \end{aligned}$$

We have proven that $\preceq^{\mathcal{B}}_{\text{CLT}}$ is a prefixed point of $\mathcal{F}^{\preceq_{\text{CLT}}}(X, \mathcal{B}')$. The Knaster-Tarski theorem implies that $\preceq^{\mathcal{B}}_{\text{CLT}} \subseteq \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, \mathcal{B}')$, because $\nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, \mathcal{B}') = \bigcup\{ \mathcal{R} \,|\, \mathcal{R} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{R}, \mathcal{B}') \}$. The definition of $\preceq^{\mathcal{B}'}_{\text{CLT}}$ ensures now that $\preceq^{\mathcal{B}}_{\text{CLT}} \subseteq \preceq^{\mathcal{B}'}_{\text{CLT}}$. □

Lemma 8.1.10. If $\mathcal{T}$ is a transitive binary relation on session contracts, then $\preceq^{\mathcal{T}}_{\text{CLT}} \subseteq \sqsubseteq^{\mathcal{T}}_{\text{CLT}}$

Proof. The proof of this lemma is similar to the proof of Proposition 6.5.18, but it relies on [R-ETA] in place of [R-ALPHA]. We have to add to the original proof the discussion for the higher-order terms. We discuss one case. The only difference with the first-order argument is that the transitivity of $\preccurlyeq_{\mathfrak{b}}$ is replaced by the transitivity of $\mathcal{T}$, which is true by hypothesis.

If $\rho_2 = !(\hat{\rho}_2).\rho'_2$ then $\text{UNF}(\rho_1) \preceq^{\mathcal{T}}_{\text{CLT}} \rho_2$ implies that $\text{UNF}(\rho_1) = \eta_1.\rho'_1$ and $\rho'_1 \preceq^{\mathcal{T}}_{\text{CLT}} \rho'_2$, and $\hat{\rho}_2 \; \mathcal{T} \; \hat{\rho}_1$. The assumption $\rho_1 \dashv^s_{\mathcal{T}} \sigma$ now implies that $\sigma = \theta.\sigma'$, with $!(\hat{\rho}_1) \bowtie_{\mathcal{T}} \theta$, and $\rho'_1 \dashv^s_{\mathcal{T}} \sigma'$. The fact that $!(\hat{\rho}_1) \bowtie_{\mathcal{T}} \theta$ implies that $\theta = ?(\hat{\sigma})$ for some $\hat{\sigma}$ such that $\hat{\rho}_1 \; \mathcal{T} \; \hat{\sigma}$. We have seen that $\hat{\rho}_2 \; \mathcal{T} \; \hat{\rho}_1$, and that $\hat{\rho}_1 \; \mathcal{T} \; \hat{\sigma}$, thus the hypothesis of $\mathcal{T}$ being transitive ensures that $\hat{\rho}_2 \; \mathcal{T} \; \hat{\sigma}$. The definition of $\bowtie$ guarantees that $!(\hat{\rho}_2) \bowtie_{\mathcal{T}} ?(\hat{\sigma})$. We also know that $\rho'_1 \preceq^{\mathcal{T}}_{\text{CLT}} \rho'_2$ and that $\rho'_1 \dashv^s_{\mathcal{T}} \sigma'$, thus $\rho'_2 \; \mathcal{R} \; \sigma'$. We have proven enough to derive

$$\frac{\rho'_2 \dashv^s_{\mathcal{T}} \sigma'}{\rho_2 \dashv^s_{\mathcal{T}} \sigma} \; !(\hat{\rho}_2) \bowtie_{\mathcal{T}} ?(\hat{\sigma}); \; [\text{R-ETA}]$$

If $\rho_2 = ?(\hat{\rho}_2).\rho'_2$ then the argument is similar to the previous one. □

The hypothesis of the previous lemma can not be weakened. We explain why in Example B.0.7.

Notation Let us denote the set of pre-orders on session contracts with the symbol $\text{PRE}(\mathsf{SC}^2_{\text{HO}})$, and its elements with the symbols $\mathcal{O}, \mathcal{P}$.

We are ready to give the alternative characterisation of the dependent client pre-orders.

Proposition 8.1.11. [Alternative characterisation dependent client pre-orders]
Let $\mathcal{O}$ be a pre-order on session contracts; $\preceq^{\mathcal{O}}_{\text{CLT}} = \sqsubseteq^{\mathcal{O}}_{\text{CLT}}$

Proof. The set inclusion $\sqsubseteq^{\mathcal{O}}_{\text{CLT}} \subseteq \preceq^{\mathcal{O}}_{\text{CLT}}$ follows from Lemma 8.1.5, $\mathcal{O}$ being reflexive, and the Knaster-Tarski theorem. The set inclusion $\preceq^{\mathcal{O}}_{\text{CLT}} \subseteq \sqsubseteq^{\mathcal{O}}_{\text{CLT}}$ follows from Lemma 8.1.10, and $\mathcal{O}$ being transitive. □

8.1.1 Syntactic client pre-orders and transitivity

In the next subsection we will need few technical results which we devise here. We dwell on the relation between the rule functional $\mathcal{F}^{\preceq_{\text{CLT}}}$, the transitivity of its arguments, and the transitivity of its images.

In order to prove that an image of $\mathcal{F}^{\preceq_{\text{CLT}}}$ is transitive, it is not enough to take into account only one of its parameters. It is *necessary* that both parameters of $\mathcal{F}^{\preceq_{\text{CLT}}}$ be transitive.

The next lemma is a standard result of lattice theory.

Lemma 8.1.12. [Golden lemma [Arnold and Niwiński, 2001, Proposition 1.3.2]]
Let the symbol θ range over μ and ν. Let E be a complete lattice, and $h : E \times E \longrightarrow E$ a function monotonic with respect to all its arguments. The ensuing equalities are true,

$$\theta x.\theta y.h(x,y) = \theta x.h(x,x) = \theta y.\theta x.h(x,y)$$

Proof. We give the proof for $\theta = \mu$. The proof for $\theta = \nu$ is similar, by the principle of symmetry.

Let $h'(x) = \mu y.h(x,y)$. The definition of fixed point implies that

$$h(x, h'(x)) = h(x, \mu y.h(x,y)) \tag{8.2}$$

Let $a = \mu x.h'(x)$; it follows that $a = \mu x.\mu y.h(x,y)$; and let $b = \mu x.h(x,x)$. We have

$$\begin{array}{lll} a & = h'(a) & \text{By definition of fixed point} \\ & = h(a, h'(a)) & \text{In view of Eq. (8.2)} \\ & = h(a,a) & \text{By definition of } h' \end{array}$$

It follows that $b \leq a$. On the other hand, $b = h(b,b)$, hence, $b \geq \mu y.h(b,y)$, and $b \geq \mu x.\mu y.h(x,y) = a$. □

In the next lemma we show a set of prefixed points of $\mathcal{F}^{\preceq_{\text{CLT}}}$ that are closed with respect to transitive closure.

Proposition 8.1.13. For every $\mathcal{B} \subseteq \mathsf{SC}^2_{\text{HO}}$, if $\mathcal{B} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{B})$ then $[\mathcal{B}]^+ \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}([\mathcal{B}]^+,[\mathcal{B}]^+)$.

Proof. We have to prove that if $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$, then it is also in $\mathcal{F}^{\preceq_{\text{CLT}}}([\mathcal{B}]^+,[\mathcal{B}]^+)$. The main argument is by rule induction on the proof of $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$.

In the base case we have the derivation

$$\frac{}{\sigma_1\ [\mathcal{B}]^+\ \sigma_3}\ \sigma_1\ \mathcal{B}\ \sigma_3;\ [\text{TRC-A}]$$

The side conditions of rule [TRC-A] and the hypothesis ensure that $(\sigma_1,\ \sigma_3) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{B})$. Lemma C.0.32, Corollary 8.1.9, and $\mathcal{B} \subseteq [\mathcal{B}]^+$ (Proposition C.0.37) imply that $(\sigma_1,\ \sigma_3) \in \mathcal{F}^{\preceq_{\text{CLT}}}([\mathcal{B}]^+,[\mathcal{B}]^+)$.

In the inductive case, the derivation which proves that $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$ has the form

$$\frac{\dfrac{\vdots}{\sigma_1\ [\mathcal{B}]^+\ \sigma_2} \quad \dfrac{\vdots}{\sigma_2\ [\mathcal{B}]^+\ \sigma_3}}{\sigma_1\ [\mathcal{B}]^+\ \sigma_3}\ [\text{TRC-R}]$$

By inductive hypothesis we have that (a) $(\sigma_1,\ \sigma_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}([\mathcal{B}]^+,[\mathcal{B}]^+)$ and (b) $(\sigma_2,\ \sigma_3) \in \mathcal{F}^{\preceq_{\text{CLT}}}([\mathcal{B}]^+,[\mathcal{B}]^+)$. To prove that $(\sigma_1,\ \sigma_3) \in \mathcal{F}^{\preceq_{\text{CLT}}}([\mathcal{B}]^+,[\mathcal{B}]^+)$ we show a derivation

$$\frac{\cdots}{\sigma_1 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_3}$$

done using the rules in Figure 8.1. We proceed by case analysis on σ_3. If $\sigma_3 = \mathbf{1}$, then we can derive

$$\frac{}{\sigma_1 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_3}\ [\text{A-Goal-C}]$$

Before discussing the other cases, note that the (a) and (b) means that we can use the inference rules in Figure 8.1 to derive the following

$$\frac{\cdots}{\sigma_1 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_2} \tag{8.3}$$

$$\frac{\cdots}{\sigma_2 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_3} \tag{8.4}$$

Note that the rules instantiated depend on the form of σ_3.

If $\sigma_3 = !(\hat{\sigma}_3).\sigma_3'$, then the derivation in (8.4) has to be due to rule [R-Out-H], so $\sigma_2 = !(\hat{\sigma}_2).\sigma_2'$ and

$$\frac{\sigma_2' \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_3' \quad \hat{\sigma}_3 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \hat{\sigma}_2}{\sigma_2 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_3} \text{ [R-Out-H]}$$

Similarly, the form of σ_2 and the derivation in (8.3) ensure that $\sigma_1 = !(\hat{\sigma}_1).\sigma_1'$ and that the derivation is as follows

$$\frac{\sigma_1' \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_2' \quad \hat{\sigma}_2 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \hat{\sigma}_1}{\sigma_1 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_2} \text{ [R-Out-H]}$$

So far, we have proven that

$$\sigma_1' \; [\mathcal{B}]^+ \; \sigma_2' \; [\mathcal{B}]^+ \; \sigma_3', \quad \hat{\sigma}_3 \; [\mathcal{B}]^+ \; \sigma_2' \; [\mathcal{B}]^+ \; \sigma_1'$$

The transitivity of $[\mathcal{B}]^+$ (Lemma 7.3.3) ensures that

$$\sigma_1' \; [\mathcal{B}]^+ \; \sigma_3', \quad \hat{\sigma}_3 \; [\mathcal{B}]^+ \; \sigma_1'$$

We are ready to apply rule [R-Out-H], to derive (σ_1, σ_3):

$$\frac{\sigma_1' \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_3' \quad \hat{\sigma}_3 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \hat{\sigma}_1}{\sigma_1 \preceq_{\text{CLT}}^{[\mathcal{B}]^+} \sigma_3} \text{ [R-Out-H]}$$

This derivation proves that $(\sigma_1, \sigma_3) \in \mathcal{F}^{\preceq_{\text{CLT}}}([\mathcal{B}]^+, [\mathcal{B}]^+)$.

The arguments for the other cases are analogous. □

Corollary 8.1.14. *The fixed point $\nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X)$ is a transitive relation.*

Proof. To prove that a relation is transitive, it is enough to show that it contains its transitive closure; so to prove that $\nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X)$ is transitive, it suffices to show that

$$[(\nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X))]^+ \subseteq \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X)$$

Let $A = \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X)$; we have to show the ensuing set inclusion,

$$[A]^+ \subseteq \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X)$$

The definition of fixed point guarantees that $A = \mathcal{F}^{\preceq_{\text{CLT}}}(A, A)$, so, by weakening, we have $A \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(A, A)$. Proposition 8.1.13 ensures that

$$[A]^+ \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}([A]^+, [A]^+)$$

We proceed as follows.

$$\begin{aligned}
[A]^+ &\subseteq \mathcal{F}^{\preceq_{\text{CLT}}}([A]^+, [A]^+) && \text{Proven above} \\
&\subseteq \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, [A]^+) && \text{By the Knaster-Tarski theorem} \\
&\subseteq \nu Y.\nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, Y) && \text{By the Knaster-Tarski theorem} \\
&= \nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X) && \text{By Lemma 8.1.12}
\end{aligned}$$

□

Monotone functions

The dependent client pre-orders $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ are parametrised over a binary relation $\mathcal{B}$, so we have a family of these pre-orders $\{\sqsubseteq^{\mathcal{B}}_{\text{CLT}}\}_{\mathcal{B}\in\mathcal{P}(\mathsf{SC}^2_{\text{HO}})}$; for instance,[1]

$$\begin{array}{ccccc} \mathcal{B}_1 & \mathcal{B}_2 & \mathcal{B}_3 & \mathcal{B}_4 & \dots \\ \downarrow & \downarrow & \downarrow & \downarrow & \\ \sqsubseteq^{\mathcal{B}_1}_{\text{CLT}} & \sqsubseteq^{\mathcal{B}_2}_{\text{CLT}} & \sqsubseteq^{\mathcal{B}_3}_{\text{CLT}} & \sqsubseteq^{\mathcal{B}_4}_{\text{CLT}} & \dots \end{array}$$

It is natural to abstract away from the relations $\mathcal{B}$'s, and study the monotonicity of the functions given by the function $\lambda X.\ \sqsubseteq^{X}_{\text{CLT}}$. In view of Proposition 8.1.8, and Proposition 8.1.11 it is easy to see that this function is monotone, as long as $X \in \textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$.

Lemma 8.1.15. Let $\mathcal{R}$ be a reflexive relation on session contracts, and let $\mathcal{T}$ be a transitive relation on session contracts. If $\mathcal{R} \subseteq \mathcal{T}$ then $\sqsubseteq^{\mathcal{R}}_{\text{CLT}} \subseteq \sqsubseteq^{\mathcal{T}}_{\text{CLT}}$.

Proof.

$$\begin{array}{lcll} \sqsubseteq^{\mathcal{R}}_{\text{CLT}} & \subseteq & \preceq^{\mathcal{R}}_{\text{CLT}} & \text{By Lemma 8.1.5} \\ & \subseteq & \preceq^{\mathcal{T}}_{\text{CLT}} & \text{By Proposition 8.1.8} \\ & \subseteq & \sqsubseteq^{\mathcal{T}}_{\text{CLT}} & \text{By Lemma 8.1.10} \end{array}$$

□

Corollary 8.1.16. *Let* $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}} : \textsc{Pre}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$ *be the function*

$$\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}(\mathcal{O}) = \sqsubseteq^{\mathcal{O}}_{\text{CLT}}$$

The function $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ *is a monotone endofunction.*

Proof. The function $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ is monotone because of Lemma 8.1.15 and of the fact that pre-orders are reflexive and transitive. The function $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ is indeed an endofunction because $\sqsubseteq^{\mathcal{O}}_{\text{CLT}}$ is a pre-order for every $\mathcal{O}$. □

8.2 Server pre-orders

In this section we extend to the parametrised LTS the restricted compliance server pre-order that we studied in Section 6.4. The results are similar to ones we have proven in the previous section, so we do not discuss them at length.

Definition 8.2.1. [$\mathcal{B}$-dependent server pre-order]
Given a binary relation on session contracts $\mathcal{B} \subseteq \mathsf{SC}^2_{\text{HO}}$, we write $\sigma_1 \sqsubseteq^{\mathcal{B}}_{\text{SVR}} \sigma_2$ whenever $\rho \dashv_{\mathcal{B}} \sigma_1$ implies that $\rho \dashv_{\mathcal{B}} \sigma_2$, for every $\rho \in \mathsf{SC}_{\text{HO}}$. We refer to the symbol $\sqsubseteq^{\mathcal{B}}_{\text{SVR}}$ as the the *$\mathcal{B}$-dependent server pre-order.*
□

Example 8.1.2 can be adapted to prove that as $\mathcal{B}$ varies also the dependent pre-orders $\sqsubseteq^{\mathcal{B}}_{\text{SVR}}$ changes; some properties of the pre-orders $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$, though, do not depend on the parameter $\mathcal{B}$.

The dual of the property Corollary 8.1.3 is true.

Corollary 8.2.2 (Bottom elements)**.**
Let $\mathcal{B}$ be a binary relation on session contracts.

(a) the term **1** *is a bottom element of* $\sqsubseteq^{\mathcal{B}}_{\text{SVR}}$

[1] We use natural numbers to distinguish the $\mathcal{B}$'s, not to say that they are countable.

$$\frac{\rho_1' \preceq^{\mathcal{B}}_{\text{SVR}} \rho_2'}{!(\hat{\rho}_1).\rho_1' \preceq^{\mathcal{B}}_{\text{SVR}} !(\hat{\rho}_2).\rho_2'}\ \hat{\rho}_2\ \mathcal{B}\ \hat{\rho}_1;\ [\text{R-OUT-H}]$$

$$\frac{\rho_1' \preceq^{\mathcal{B}}_{\text{SVR}} \rho_2'}{?(\hat{\rho}_1).\rho_1' \preceq^{\mathcal{B}}_{\text{SVR}} ?(\hat{\rho}_2).\rho_2'}\ \hat{\rho}_1\ \mathcal{B}\ \hat{\rho}_2;\ [\text{R-IN-H}]$$

Figure 8.2: Additional inference rules for the rule functional $\mathcal{F}^{\preceq_{\text{CLT}}}$. The other rules are in Figure 6.3

(b) if $\sigma_1 \sqsubseteq^{\mathcal{B}}_{\text{SVR}} \mathbf{1}$ *then* $\sigma_1 = \mathbf{1}$.

Proof. The proof is similar to the proof of Lemma 6.1.5. □

Corollary 8.2.3. $\rho_1 \sqsubseteq^{\mathcal{B}}_{\text{SVR}} \rho_2$ *if and only* $\text{UNF}(\rho_1) \sqsubseteq^{\mathcal{B}}_{\text{SVR}} \text{UNF}(\rho_2)$.

Lemma 8.2.4. Let $\mathcal{R}$ be a reflexive relation on session contracts. If $\sigma_1 \sqsubseteq^{\mathcal{R}}_{\text{SVR}} \sigma_2$ then one of the following is true.

(a) $\text{UNF}(\sigma_1) = \mathbf{1}$

(b) $\text{UNF}(\sigma_1) = !(\hat{\rho}_1).\sigma_1'$, $\text{UNF}(\sigma_2) = !(\hat{\rho}_2).\sigma_2'$, $\hat{\rho}_2\ \mathcal{B}\ \hat{\rho}_1$ and $\sigma_1' \sqsubseteq^{\mathcal{R}}_{\text{SVR}} \sigma_2'$

(c) $\text{UNF}(\sigma_1) = ?(\hat{\rho}_1).\sigma_1'$, $\text{UNF}(\sigma_2) = ?(\hat{\rho}_2).\sigma_2'$, $\hat{\rho}_1\ \mathcal{B}\ \hat{\rho}_2$ and $\sigma_1' \sqsubseteq^{\mathcal{R}}_{\text{SVR}} \sigma_2'$

(d) $\text{UNF}(\sigma_1) = !\mathtt{t}_1.\sigma_1'$, $\text{UNF}(\sigma_2) = !\mathtt{t}_2.\sigma_2'$, $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$ and $\sigma_1' \sqsubseteq^{\mathcal{R}}_{\text{SVR}} \sigma_2'$

(e) $\text{UNF}(\sigma_1) = ?\mathtt{t}_1.\sigma_1'$, $\text{UNF}(\sigma_2) = ?\mathtt{t}_2.\sigma_2'$, $\mathtt{t}_1 \preccurlyeq_{\mathtt{b}} \mathtt{t}_2$ and $\sigma_1' \sqsubseteq^{\mathcal{R}}_{\text{SVR}} \sigma_2'$

(f) $\text{UNF}(\sigma_1) = \sum_{i\in I} ?\mathtt{l}_{\mathtt{i}}.\sigma_i^1$, $\text{UNF}(\sigma_2) = \sum_{j\in J} ?\mathtt{l}_{\mathtt{j}}.\sigma_j^2$, with $I \subseteq J$ and $\sigma_i^1 \sqsubseteq^{\mathcal{R}}_{\text{SVR}} \sigma_i^2$ for every $i \in I$

(g) $\text{UNF}(\sigma_1) = \bigoplus_{i\in I} !\mathtt{l}_{\mathtt{i}}.\sigma_i^1$, $\text{UNF}(\sigma_2) = \bigoplus_{j\in J} !\mathtt{l}_{\mathtt{j}}.\sigma_j^2$, with $J \subseteq I$ and $\sigma_j^2 \sqsubseteq^{\mathcal{R}}_{\text{SVR}} \sigma_j^2$ for every $j \in J$

Proof. The proof is similar to the one of Lemma 8.1.5. □

Definition 8.2.5. [$\mathcal{B}$-syntactic dependent server pre-order]
Let $\mathcal{F}^{\preceq_{\text{SVR}}} : \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \times \mathcal{P}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \mathcal{P}(\mathsf{SC}^2_{\text{HO}})$ be the rule functional given by the inference rules in Figure 8.2.

Fix a binary relation $\mathcal{B}$. If $X \subseteq \mathcal{F}^{\preceq_{\text{SVR}}}(X, \mathcal{B})$, then we say that X is a *co-inductive $\mathcal{B}$-syntactic server pre-order*. Lemma C.0.36 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\preceq_{\text{SVR}}}(X, \mathcal{B})$; we call this solution the *$\mathcal{B}$-syntactic server pre-order*, and we denote it $\preceq^{\mathcal{B}}_{\text{SVR}}$. That is $\preceq^{\mathcal{B}}_{\text{SVR}} = \nu X.\mathcal{F}^{\preceq_{\text{SVR}}}(X, \mathcal{B})$. □

Lemma 8.2.6. Let $\mathcal{T}$ be a transitive on session contracts. The set inclusion $\preceq^{\mathcal{B}}_{\text{SVR}} \subseteq \sqsubseteq^{\mathcal{B}}_{\text{SVR}}$ holds true.

Proof. The argument is analogous to the proof of Lemma 8.1.10. □

Proposition 8.2.7. [Alternative characterisation dependent server pre-order]
Let $\mathcal{O}$ be a pre-order on session contracts. We have the equality

$$\sqsubseteq^{\mathcal{O}}_{\text{SVR}} = \preceq^{\mathcal{O}}_{\text{SVR}}$$

Proof. The argument is analogous to the one used to prove Proposition 8.1.11. □

Lemma 8.2.8. Let $\mathcal{F}^{\text{SRV}} : \text{PRE}(\mathsf{SC}^2_{\text{HO}}) \longrightarrow \text{PRE}(\mathsf{SC}^2_{\text{HO}})$ be the function

$$\mathcal{F}^{\text{SRV}}(\mathcal{O}) = \sqsubseteq^{\mathcal{O}}_{\text{SVR}}$$

The function $\mathcal{F}^{\text{SRV}}$ is monotone endofunction.

Proof. The argument is similar to the proof of Corollary 8.1.16. □

8.3 Client and server pre-orders

Thus far, in Section 8.1 and Section 8.2 we have introduced two families of pre-orders, namely the dependent client ones and the dependent server ones:

$$\{\sqsubseteq^{\mathcal{B}}_{\text{CLT}}\}_{\mathcal{B}\in\mathcal{P}(\mathsf{SC}^2_{\text{HO}})}, \quad \{\sqsubseteq^{\mathcal{B}}_{\text{SVR}}\}_{\mathcal{B}\in\mathcal{P}(\mathsf{SC}^2_{\text{HO}})}$$

After having studied the elements of these families, we have turned our attention the functions that map binary relations $\mathcal{B}$ to the pre-orders $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$ or $\sqsubseteq^{\mathcal{B}}_{\text{SVR}}$; and we have shown when these functions are monotone. The outcome of our study are the monotone endofunctions $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ and $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}$. Thanks to the Knaster-Tarski theorem these functions have fixed points, in particular the greatest ones (Definition 8.3.2, Definition 8.3.6). These objects turns out to depend only on themselves (Lemma 8.3.3, Lemma 8.3.7). In a sense, these fixed points embody the notion of "absolute" client and of "absolute" server pre-orders, and they give us a *non-arbitrary* relation to let session contracts interact.

We have argued that, a priori, we cannot fix a relation that formalises when a higher-order action should interact with another higher-order action; and so we have introduced the dependency of the LTS on the relations $\mathcal{B}$'s.

The intersection of the fixed points of $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ and $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}$ will give us an LTS that does not depend on any binary relation on session contracts other than the intersection itself, thus these fixed points free us from the dependency of the LTS on $\mathcal{B}$.

The chief results of this section are the properties of the mentioned fixed points, and two properties of their intersection (Lemma 8.3.10, Lemma 8.3.12).

We are first concerned with the client pre-order; and then move on to the server pre-order. After the suitable definitions, for each pre-order we show that it depends on itself (Lemma 8.3.3, Lemma 8.3.7), we give a syntactic characterisation (Lemma 8.3.4, Lemma 8.3.8) and a proof method (Lemma 8.3.5, Lemma 8.3.9).

First, we establish a technicality; the last result that we need in order to apply the Knaster-Tarski theorem. We defer the proof to Appendix A.

Proposition 8.3.1. The pre-order $\langle\, \text{PRE}(\mathsf{SC}^2_{\text{HO}}), \subseteq \,\rangle$ is a complete lattice.

Proof. See Lemma A.0.4. □

Definition 8.3.2. [Client pre-order]
Recall the function $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ defined in Corollary 8.1.16. If $X \subseteq \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}(X)$, then we say that X is a *co-inductive client pre-order*. Corollary 8.1.16 and the Knaster-Tarski theorem ensure that there exists the greatest solution of the equation $X = \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}(X)$; we call this solution the *client pre-order*, and we denote it $\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}$. That is $\sqsubseteq^{\mathsf{ho}}_{\text{CLT}} = \nu X.\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}(X)$. □

Lemma 8.3.3. The client pre-order is a dependent client pre-order: $\sqsubseteq^{\mathsf{ho}}_{\text{CLT}} = \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}_{\text{CLT}}$.

Proof. Consider the following equalities.

$$\begin{array}{lcll} \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} & = & \nu\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}} & \text{By definition} \\ & = & \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}(\nu\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}) & \text{By definition of fixed point} \\ & = & \sqsubseteq^{\nu\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}}_{\mathrm{CLT}} & \text{By definition of } \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}} \\ & = & \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}_{\mathrm{CLT}} & \text{By definition of } \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} \end{array}$$

□

Now we show how the client pre-order is related to the rule functional $\mathcal{F}^{\preceq_{\mathrm{CLT}}}$.

Lemma 8.3.4. $\sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} = \nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, X)$

Proof. Let $A = \nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, X)$.

We have to show two set inclusions, namely

$$A \subseteq \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}, \quad \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} \subseteq A$$

We prove the left inclusion.

$$\begin{array}{lcll} \nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, X) & = & \nu Y\nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, Y) & \text{By Lemma 8.1.12} \\ & = & \nu Y.\ \preceq^{Y}_{\mathrm{CLT}} & \text{By definition of } \preceq_{\mathrm{CLT}} \\ & = & \preceq^{\nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X,X)}_{\mathrm{CLT}} & \text{By definition of fixed point} \\ & \subseteq & \sqsubseteq^{\nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X,X)}_{\mathrm{CLT}} & \text{By Lemma 8.1.10, and Corollary 8.1.14,} \\ & \subseteq & \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} & \text{By the Knaster-Tarski theorem} \end{array}$$

Now we prove the right set inclusion.

$$\begin{array}{lcll} \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} & = & \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}_{\mathrm{CLT}} & \text{By definition} \\ & \subseteq & \preceq^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}_{\mathrm{CLT}} & \text{By Lemma 8.1.5, } \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} \text{ being reflexive} \\ & = & \nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}) & \text{By definition} \\ & \subseteq & \nu Y\nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, Y) & \text{By the Knaster-Tarski theorem} \\ & = & \nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, X) & \text{By Lemma 8.1.12} \end{array}$$

□

Lemma 8.3.5. [Proof method for $\sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}$]
If $\mathcal{R}$ is a co-inductive $\mathcal{R}$-syntactic client pre-order, then $\mathcal{R} \subseteq \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}$.

Proof. Consider the ensuing passages.

$$\begin{array}{lcll} \mathcal{R} & \subseteq & \mathcal{F}^{\preceq_{\mathrm{CLT}}}(\mathcal{R}, \mathcal{R}) & \text{By hypothesis} \\ [R]^{+} & \subseteq & \mathcal{F}^{\preceq_{\mathrm{CLT}}}([\mathcal{R}]^{+}, [\mathcal{R}]^{+}) & \text{By Proposition 8.1.13} \\ & \subseteq & \nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, [\mathcal{R}]^{+}) & \text{By the Knaster-Tarski theorem} \\ & \subseteq & \nu X\nu Y.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, Y) & \text{By the Knaster-Tarski theorem} \\ & = & \nu X.\mathcal{F}^{\preceq_{\mathrm{CLT}}}(X, X) & \text{By Lemma 8.1.12} \\ & = & \sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}} & \text{By Lemma 8.3.4} \end{array}$$

□

As to the server pre-order, we proceed similarly to what we did for $\sqsubseteq^{\mathsf{ho}}_{\mathrm{CLT}}$.

Definition 8.3.6. [Server pre-order]
Recall the definition of $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{SVR}}}$. If $X \subseteq \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{SVR}}}(X)$, then we say that X is a *co-inductive server pre-order*. Lemma 8.2.8 and the Knaster-Tarski theorem ensure that there exists the greatest solution of

the equation $X = \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}(X)$; we call this solution the *server pre-order*, and we denote it $\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}$. That is $\sqsubseteq^{\mathsf{ho}}_{\text{SVR}} = \nu X.\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}(X)$. □

Lemma 8.3.7. The server pre-order is a dependent server pre-order: $\sqsubseteq^{\mathsf{ho}}_{\text{SVR}} = \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}_{\text{SVR}}$.

Proof. The proof is analogous to the one of Lemma 8.3.3. □

Lemma 8.3.8. $\sqsubseteq^{\mathsf{ho}}_{\text{SVR}} = \nu X.\mathcal{F}^{\preceq_{\text{SVR}}}(X, X)$

Proof. The proof is analogous to the one of Lemma 8.3.4. □

Lemma 8.3.9.
If $\mathcal{R}$ is a co-inductive $\mathcal{R}$-syntactic server pre-order, then $\mathcal{R} \subseteq \sqsubseteq^{\mathsf{ho}}_{\text{SVR}}$.

Proof. Similar to the proof of Lemma 8.3.5. □

In the remaining part of this section we study some properties of the intersection of the fixed points of $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ and $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}$. The corollaries (Corollary 8.3.11, Corollary 8.3.13)of the results (Lemma 8.3.10, Lemma 8.3.12) are necessary to prove Proposition 8.4.2 in Section 8.4.

Lemma 8.3.10. Let $\mathcal{R}$ be a fixed point of $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$, and $\mathcal{S}$ be a fixed point of $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}$. The relation $\mathcal{R} \cap \mathcal{S}$ is a co-inductive $(\mathcal{R} \cap \mathcal{S})$-syntactic client pre-order.

Proof. Let $\mathcal{B} = \mathcal{R} \cap \mathcal{S}$. We have to prove the set inclusion $\mathcal{B} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B}, \mathcal{B})$. The definitions of $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ and $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}$ and the hypothesis imply the following equality,

$$\mathcal{B} = \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \cap \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \tag{8.5}$$

so it is enough to show that

$$\sqsubseteq^{\mathcal{R}}_{\text{CLT}} \cap \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B}, \mathcal{B})$$

Fix a pair $(\rho_1, \rho_2) \in \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \cap \sqsubseteq^{\mathcal{S}}_{\text{SVR}}$; we have to prove that

$$(\rho_1, \rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B}, \mathcal{B})$$

If $depth(\rho_1) + depth(\sigma_2) > 0$ then we use Corollary 8.2.3 and Corollary 8.1.4; they ensure that $\text{UNF}(\rho_1) \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \text{UNF}(\rho_2)$ and that $\text{UNF}(\rho_1) \sqsubseteq^{\mathcal{S}}_{\text{CLT}} \text{UNF}(\rho_2)$. It follows that we can apply [R-Unfold],

$$\frac{\text{UNF}(\rho_1) \preceq^{\mathcal{B}}_{\text{CLT}} \text{UNF}(\rho_2)}{\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2} \; depth(\rho_1) + depth(\rho_2) > 0 \;\; [\text{R-Unfold}]$$

If $depth(\rho_1) + depth(\rho_2) = 0$ then we reason differently. The relations $\sqsubseteq^{\mathcal{R}}_{\text{CLT}}$ and $\sqsubseteq^{\mathcal{R}}_{\text{SVR}}$ are pre-orders, so they are reflexive; it follows that $\mathcal{R}$ and $\mathcal{S}$ are reflexive as well. This allows us to use Lemma 8.1.5 and Lemma 8.2.4.

We proceed reasoning by case analysis on ρ_2.

(a) If $\rho_2 = \mathbf{1}$, then we have the derivation

$$\frac{}{\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2} \; [\text{A-Goal-C}]$$

(b) If $\rho_2 = !(\hat{\rho}_2).\rho'_2$, then $\rho_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_2$ implies that $\rho_1 = !(\hat{\rho}_1).\rho'_1$, with $\hat{\rho}_2 \; \mathcal{R} \; \hat{\rho}_1$, and $\rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho'_2$; while $\rho_1 \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \rho_2$ implies that $\hat{\rho}_2 \; \mathcal{S} \; \hat{\rho}_1$, and $\rho'_1 \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \rho'_2$. We know enough to state that

$$\hat{\rho}_2 \; \mathcal{B} \; \hat{\rho}_1, \quad \rho'_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \cap \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \rho'_2$$

Thanks to (8.5) we know that $\rho_1' \mathcal{B} \rho_2'$. Now we can infer

$$\frac{\rho_1' \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2' \quad \hat{\rho}_2 \preceq^{\mathcal{B}}_{\text{CLT}} \hat{\rho}_1}{\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2} \text{ [R-OUT-H]}$$

(c) If $\rho_2 = ?(\hat{\rho}_2).\rho_2'$, then argument is analogous to the one we used in the previous case.

(d) If $\rho_2 = !\mathtt{t}_2.\rho_2'$, then $\rho_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_2$ implies that $\rho_1 = !\mathtt{t}_1.\rho_1'$, with $\mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1$, and $\rho_1' \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_2'$; the assumption $\rho_1 \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \rho_2$ that $\rho_1' \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \rho_2'$. Now we can infer

$$\frac{\rho_1' \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2'}{\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2} \ \mathtt{t}_2 \preccurlyeq_{\mathtt{b}} \mathtt{t}_1 \text{ [R-PUT-F]}$$

(e) If $\rho_2 = ?\mathtt{t}_2.\rho_2'$, the the argument is analogous to the one used in case (d).

(f) If $\rho_2 = \sum_{j\in J} \mathtt{l}_j.\rho_j^2$, then $\rho_1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_2$ and Lemma 8.1.5 implies that $\rho_2 = \sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho_i^1$ with $I \subseteq J$ and $\rho_i^1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_i^2$. Since we know also that $\rho_1 \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \rho_2$, Lemma 8.2.4 implies that $\rho_i^1 \sqsubseteq^{\mathcal{S}}_{\text{SVR}} \rho_i^2$ for every $i \in I$. It follows that for every $i \in I$, $\rho_i^1 \mathcal{B} \rho_i^2$; and so we can apply [R-BRANCH],

$$\frac{\rho_1^1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_1^2 \ \cdots \ \rho_{|I|}^1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_{|I|}^2}{\sum_{i\in I} ?\mathtt{l}_\mathtt{i}.\rho_i^1 \preceq^{\mathcal{B}}_{\text{CLT}} \sum_{j\in J} ?\mathtt{l}_\mathtt{j}.\rho_j^2,} \ I \subseteq J \text{ [R-BRANCH]}$$

(g) If $\rho_2 = \bigoplus_{j\in J} \mathtt{l}_j.\rho_j^2$, then the argument is similar to the one for the previous case.

□

We need the previous lemma to prove what really interests us.

Corollary 8.3.11. *If $\mathcal{R} = \sqsubseteq^{\mathcal{R}}_{\text{CLT}}$ and $\mathcal{S} = \sqsubseteq^{\mathcal{S}}_{\text{SVR}}$, then the relation $\mathcal{R} \cap \mathcal{S}$ is contained in $\sqsubseteq^{\mathcal{R}\cap\mathcal{S}}_{\text{CLT}}$.*

Proof. Lemma 8.3.10 ensures that $\mathcal{R} \cap \mathcal{S} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{R} \cap \mathcal{S}, \mathcal{R} \cap \mathcal{S})$, hence Lemma 8.3.9 guarantees that $\mathcal{R} \cap \mathcal{S} \subseteq \sqsubseteq^{\mathcal{R}\cap\mathcal{S}}_{\text{CLT}}$. □

Note that Corollary 8.3.11 is not obvious; the obvious statement is

$$\mathcal{R} = \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \text{ implies } \mathcal{R} \cap \mathcal{S} \subseteq \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \tag{8.6}$$

in the sense that (8.6) is true by definition, and the consequences tell us nothing more than the hypothesis. On the contrary, Corollary 8.3.11 states that

$$\mathcal{S} = \sqsubseteq^{\mathcal{S}}_{\text{SVR}}, \mathcal{R} = \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \text{ imply } \mathcal{R} \cap \mathcal{S} \subseteq \sqsubseteq^{\mathcal{R}\cap\mathcal{S}}_{\text{CLT}} \tag{8.7}$$

The statement (8.7) is not obvious because the parameter of $\sqsubseteq_{\text{CLT}}$ in the consequences (i.e. $\mathcal{R} \cap \mathcal{S}$) differs from the parameter that appear in the hypothesis (i.e. $\mathcal{R}$). Moreover, the corollary means that we can use a dependent *server* pre-order to infer some of the properties of a dependent *client* pre-order.

Lemma 8.3.12. Let $\mathcal{R}$ be a fixed point of $\mathcal{F}^{\sqsubseteq^{\text{ho}}_{\text{CLT}}}$, and $\mathcal{S}$ be a fixed point of $\mathcal{F}^{\sqsubseteq^{\text{ho}}_{\text{SVR}}}$. The pre-order $\mathcal{R} \cap \mathcal{S}$ is a co-inductive $\mathcal{R} \cap \mathcal{S}$-syntactic server pre-order.

Proof. the proof of this lemma is similar to the proof of Lemma 8.3.10. □

Corollary 8.3.13. *If $\mathcal{R} = \sqsubseteq^{\mathcal{R}}_{\text{CLT}}$ and $\mathcal{S} = \sqsubseteq^{\mathcal{S}}_{\text{SVR}}$, then the relation $\mathcal{R} \cap \mathcal{S}$ is contained in $\sqsubseteq^{\mathcal{R}\cap\mathcal{S}}_{\text{SVR}}$.*

Proof. Lemma 8.3.12 states that$\mathcal{R} \cap \mathcal{S} \subseteq \mathcal{F}^{\preceq_{\text{SVR}}}(\mathcal{R} \cap \mathcal{S}, \mathcal{R} \cap \mathcal{S})$. From Lemma 8.3.9, it follows that $\mathcal{R} \cap \mathcal{S} \subseteq \sqsubseteq^{\mathcal{R}\cap\mathcal{S}}_{\text{SVR}}$. □

8.4 A behavioural model of sub-typing

In the previous section we have defined two pre-orders, namely $\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}$ and $\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}$; these pre-orders generalise the first-order pre-orders $\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ and $\sqsubseteq^{\mathsf{fo}}_{\text{SVR}}$, in the following sense

$$\begin{array}{ccccc} \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} & \subseteq & \sqsubseteq^{\emptyset}_{\text{CLT}} & \subseteq & \sqsubseteq^{\mathsf{ho}}_{\text{CLT}} \\ \sqsubseteq^{\mathsf{fo}}_{\text{SVR}} & \subseteq & \sqsubseteq^{\emptyset}_{\text{SVR}} & \subseteq & \sqsubseteq^{\mathsf{ho}}_{\text{SVR}} \end{array}$$

In this section we prove that the intersection of the pre-orders $\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}$ and $\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}$ is a fully abstract model of the sub-typing $\preccurlyeq_{\mathsf{sbt}}$ via the interpretation $\mathcal{M}$. The proofs are similar to the ones we saw in Section 6.3.

Definition 8.4.1. [Session contract pre-order]
The *session pre-order* $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$ is defined as $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} = \sqsubseteq^{\mathsf{ho}}_{\text{CLT}} \cap \sqsubseteq^{\mathsf{ho}}_{\text{SVR}}$. We say that a relation $\mathcal{R}$ is a co-inductive session contract pre-order if and only if $\mathcal{R} \subseteq \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \cap \sqsubseteq^{\mathcal{R}}_{\text{SVR}}$. □

Proposition 8.4.2. $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \subseteq \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{CLT}} \cap \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{SVR}}$.

Proof. By definition $\sqsubseteq^{\mathsf{ho}}_{\text{CLT}} = \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}(\sqsubseteq^{\mathsf{ho}}_{\text{CLT}})$ and $\sqsubseteq^{\mathsf{ho}}_{\text{SVR}} = \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}}(\sqsubseteq^{\mathsf{ho}}_{\text{SVR}})$, hence we can apply Corollary 8.3.11 and Corollary 8.3.13, to state that 1) $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \subseteq \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{CLT}}$, and that 2) $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \subseteq \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{SVR}}$. This is enough to see that $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \subseteq \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{CLT}} \cap \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{SVR}}$. □

Corollary 8.4.3. *The intersection of the greatest fixed points of $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ and $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}}$ is the greatest fixed point of the the intersection of the two functionals. Formally,*

$$\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} = \nu X.(\sqsubseteq^{X}_{\text{CLT}} \cap \sqsubseteq^{X}_{\text{SVR}})$$

Proof. On the one hand, $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \subseteq \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{CLT}} \cap \sqsubseteq^{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}}_{\text{SVR}}$, thus $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \subseteq \nu X.\ \sqsubseteq^{X}_{\text{CLT}} \cap \sqsubseteq^{X}_{\text{SVR}}$. On the other hand, if we let $A = \nu X.\ \sqsubseteq^{X}_{\text{CLT}} \cap \sqsubseteq^{X}_{\text{SVR}}$, then $A \subseteq \sqsubseteq^{A}_{\text{CLT}}$, and $A \subseteq \sqsubseteq^{A}_{\text{SVR}}$, thus $A \subseteq \sqsubseteq^{\mathsf{ho}}_{\text{CLT}}$ and $A \subseteq \sqsubseteq^{\mathsf{ho}}_{\text{SVR}}$; and so $A \subseteq \sqsubseteq^{\mathsf{ho}}_{\text{CLT}} \cap \sqsubseteq^{\mathsf{ho}}_{\text{SVR}} = \sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$. □

Corollary 8.4.4. *Every co-inductive session contract pre-order $\mathcal{R}$ is contained in $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$.*

Proof. The Knaster-Tarski theorem ensures that $\mathcal{R} \subseteq \nu X.\ \sqsubseteq^{X}_{\text{CLT}} \cap \sqsubseteq^{X}_{\text{SVR}}$, and Corollary 8.4.3 ensures that $\nu X.\ \sqsubseteq^{X}_{\text{CLT}} \cap \sqsubseteq^{X}_{\text{SVR}} = \sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$. The transitivity of $\subseteq$ ensures that $\mathcal{R} \subseteq \sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$. □

Lemma 8.4.5. Let $\mathcal{R}$ be a co-inductive session contract pre-order, and let

$$\mathcal{T} = \{\, (\mathcal{M}^{-1}(\sigma_1),\, \mathcal{M}^{-1}(\sigma_2)) \mid \sigma_1\ \mathcal{R}\ \sigma_2 \,\}$$

The relation $\mathcal{T}$ is a type simulation.

Proof. We have to prove that $\mathcal{T} \subseteq \mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}(\mathcal{T})$. Fix a $S_1\ \mathcal{T}\ S_2$. By definition there exists $\sigma_1\ \mathcal{R}\ \sigma_2$, such that

1) $S_1 = \mathcal{M}^{-1}(\sigma_1)$

2) $S_2 = \mathcal{M}^{-1}(\sigma_2)$

The proof is similar to the proof of Proposition 6.3.3, and amounts to a case analysis on S_1. The only difference with that lemma is that we have two more cases to discuss; namely the ones that involve higher-order. We discuss only one of the two cases, for the other is analogous.

a) If $S_1 = ![\,S\,]; S_1'$ then $\sigma_1 = !(\mathcal{M}^{-1}(S)).\mathcal{M}^{-1}(S_1')$, thus point (b) of Lemma 8.2.4 implies that $\text{UNF}(\sigma_2) = !(\hat{\sigma}_2).\sigma_2'$, with $\mathcal{M}^{-1}(S)\ \mathcal{R}\ \hat{\sigma}_2$, and $\mathcal{M}^{-1}(S_1')\ \mathcal{R}\ \sigma_2'$. The definition of $\mathcal{M}$ and the

construction of $\mathcal{T}$ ensure that $\text{UNF}(S_2) = ![\,\hat{S}\,]; S_2'$, for some $\hat{S}$ and S_2' such that $S \;\mathcal{T}\; \hat{S}$ and $S_1' \;\mathcal{T}\; S_2'$. Now we can infer

$$\frac{S_1' \preccurlyeq_{\mathsf{sbt}} S_2' \quad S \preccurlyeq_{\mathsf{sbt}} \hat{S}}{![\,S\,]; S_1' \preccurlyeq_{\mathsf{sbt}} ![\,\hat{S}\,]; S_2'}\ [\text{R-OUT}]$$

The arguments for the other cases are in Proposition 6.3.3 □

Lemma 8.4.6. Let $\mathcal{T}$ be a type simulation, and let

$$\mathcal{R} = \{\, (\mathcal{M}(S_1), \mathcal{M}(S_2)) \mid S_1 \;\mathcal{T}\; S_2 \,\}$$

the relation $\mathcal{R}$ is a $\mathcal{R}$-syntactic client pre-order.

Proof. We have to prove that $\mathcal{R} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{R}, \mathcal{R})$. Let $\rho_1 \;\mathcal{R}\; \rho_2$; by construction there exist two session types S_1, and S_2 such that $\rho_1 = \mathcal{M}(S_1)$, $\rho_2 = \mathcal{M}(S_1)$, and $S_1 \;\mathcal{T}\; S_2$.

The proof is similar to the argument described in Theorem 6.3.4. We proceed by case analysis on ρ_2; all the cases that do not involve higher-order terms are exactly as in Theorem 6.3.4. We discuss only one of the higher-order cases.

If $\rho_2 = !(\hat{\rho}_2).\rho_2$, then the definition of $\mathcal{M}$ implies that $S_2 = ![\,\hat{S}_2\,]; S_2'$ for some $\hat{S}_2$ and S_2' such that $\hat{\rho}_2 = \mathcal{M}(\hat{S}_2')$ and $\rho_2' = \mathcal{M}(S_2')$; since $\mathcal{T}$ is a type simulation it follows that $S_1 = ![\,\hat{S}_1\,]; S_1'$, $\hat{S}_2 \;\mathcal{T}\; \hat{S}_1$, $S_1' \;\mathcal{T}\; S_2'$. The equality $\rho_1 = \mathcal{M}(S_1)$ ensures that $\rho_1 = !\mathcal{M}(\hat{S}_1)\mathcal{M}(S_1')$ By construction it follows that $\hat{\rho}_2 \;\mathcal{R}\; \mathcal{M}(\hat{S}_1)$ and $\mathcal{M}(S_1') \;\mathcal{R}\; \rho_2'$, thus we can infer

$$\frac{\mathcal{M}(S_1') \preccurlyeq_{\mathsf{sbt}} \rho_2' \quad \hat{\rho}_2 \preccurlyeq_{\mathsf{sbt}} \mathcal{M}(\hat{S}_1)}{\rho_1 \preccurlyeq_{\mathsf{sbt}} \rho_2}\ [\text{R-OUT-H}]$$

□

Lemma 8.4.7. Let $\mathcal{T}$ be a type simulation, and let

$$\mathcal{R} = \{\, (\mathcal{M}(S_1), \mathcal{M}(S_2)) \mid S_1 \;\mathcal{T}\; S_2 \,\}$$

the relation $\mathcal{R}$ is a $\mathcal{R}$-syntactic server pre-order.

Proof. We have to prove that $\mathcal{R} \subseteq \mathcal{F}^{\preceq_{\text{SVR}}}(\mathcal{R})$; this to aim fix a pair $\sigma_1 \;\mathcal{R}\; \sigma_2$; by construction there exist two session types S_1, and S_2 such that $\rho_1 = \mathcal{M}(S_1)$, $\rho_2 = \mathcal{M}(S_1)$, and $S_1 \;\mathcal{T}\; S_2$.

The proof proceeds as in Lemma 8.4.6. We reason by case analysis on σ_1.

□

Corollary 8.4.8. *Let $\mathcal{T}$ be a type simulation, and let*

$$\mathcal{R} = \{\, (\mathcal{M}(S_1), \mathcal{M}(S_2)) \mid S_1 \;\mathcal{T}\; S_2 \,\}$$

the relation $\mathcal{R}$ is a co-inductive session contract pre-order.

Proof. We want to prove that $\mathcal{R} \subseteq \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \cap \sqsubseteq^{\mathcal{R}}_{\text{SVR}}$; thus, in view of Proposition 8.1.11 and Proposition 8.2.7, it suffices to prove that $\mathcal{R} \subseteq \preceq^{\mathcal{R}}_{\text{CLT}} \cap \preceq^{\mathcal{R}}_{\text{SVR}}$. This set inclusion follows from $\mathcal{R} \subseteq \preceq^{\mathcal{R}}_{\text{CLT}}$ and $\mathcal{R} \subseteq \preceq^{\mathcal{R}}_{\text{SVR}}$, and, by definition, to prove these inclusions it is enough to show that

(1) $\mathcal{R} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{R}, \mathcal{R})$

(2) $\mathcal{R} \subseteq \mathcal{F}^{\preceq_{\text{SVR}}}(\mathcal{R}, \mathcal{R})$

The set inclusions are proven in Lemma 8.4.6 and Lemma 8.4.7. □

Theorem 8.4.9. [Full abstraction]
Let S_1 and S_2 be strict session types. $\mathcal{M}(S_1) \sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \mathcal{M}(\mathcal{S}_2)$ if and only if $S_1 \preccurlyeq_{\mathsf{sbt}} S_2$.

Proof. The two inclusions that we are required to prove follow from Lemma 8.4.5 and Corollary 8.4.8. □

Theorem 8.4.9 extends to the whole theory of session types sub-typing à la Gay and Hole the behavioural model due to the compliance relation.

Throughout this chapter we have shown how to adapt the proof that

$$\sqsubseteq^{\mathsf{fo}}_{\text{SVR}} \cap \sqsubseteq^{\mathsf{fo}}_{\text{CLT}} \cong \preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$$

in order to prove that

$$\sqsubseteq^{\mathsf{ho}}_{\text{CLT}} \cap \sqsubseteq^{\mathsf{ho}}_{\text{SVR}} \cong \preccurlyeq_{\mathsf{sbt}}$$

At the end of Chapter 6 we have already commented on the meaning of such a model as the one exhibited by Theorem 8.4.9. Here we remark just that our models show that the the standard sub-typing for session types is a refinement for peers. This means that the endpoints of a communication channel, say a^- and a^+, should not be referred to as client and server, for they are not used according to a client/server logic. If two processes interact correctly via the endpoints a^- and a^+, then both processes have to be equally satisfied by the interactions that take place on a.

8.5 Related Work

In order to model the sub-typing on higher-order types we had to face a technical difficulty: to remove in a non-arbitrary way the parameter $\mathcal{B}$ from the LTS of session contracts. To do so we followed the approach of [Padovani, 2013], and studied the monotonicity of the (endo)functions that map pre-orders on session contracts to the dependent client and server. Once established the monotonicity, the main results followed just by tailoring the proof of Theorem 6.3.4.

In Section 6.6 we have seen that $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$ and the fair pre-order of [Padovani, 2011] are not comparable (see Eq. (6.10)). This result extends to the higher-order setting:

$$\leq \not\subseteq \sqsubseteq^{\mathsf{ho}}_{\text{P2P}}, \qquad \sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \not\subseteq \leq \tag{8.8}$$

The right inequality follows form $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}} \subseteq \sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$ and $\sqsubseteq^{\mathsf{fo}}_{\text{P2P}} \not\subseteq le$.

The left inequality is true because $\mu x.\, \mathtt{p!livelock}.x \leq \mu x.\, \mathtt{p!stop}.x$, while $\mu x.\, \mathtt{!livelock}.x \not\sqsubseteq^{\mathsf{ho}}_{\text{P2P}} \mu x.\, \mathtt{!stop}.x$.

Recently Dardha et al. have shown a fully abstract encoding of session types into the standard types of pi-calculus. Theorem 3 of [Dardha et al., 2012] show that the sub-typing on types of the pi-calculus, <:, captures exactly the sub-typing à la Gay and Hole,

Theorem 3 For every session type S, T, $S \preccurlyeq_{\mathsf{sbt}} T$ if and only if $[\![S]\!] <: [\![T]\!]$.

Our Theorem 8.4.9 justifies and explains in behavioural terms the relation $\preccurlyeq_{\mathsf{sbt}}$; essentially it states that the relation $\preccurlyeq_{\mathsf{sbt}}$ is *the* peer pre-order given by the compliance in the LTS

$$\langle\, \mathsf{SC}_{\text{HO}},\ \mathsf{Act}_{\tau\checkmark} \cup \mathsf{SC}_{\text{HO}},\ \longrightarrow_{\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}} \,\rangle$$

Theorem 3 above, instead, shows that there are other syntactic means to define this peer pre-order. The combination of the two theorems establishes a connection between the standard sub-typing <:, a subset of the types for pi-calculus, the pre-orders given by the compliance relation, and the higher-order session contracts.

To what extent the compliance relation can be used to explain the standard pre-order $<:$ becomes a natural problem to address (see (Q16) in Section 11.2).

Chapter 9

Ongoing work: session contracts as types

We concluded Chapter 8 by showing that higher-order session contracts and the pre-order $\sqsubseteq^{\mathsf{ho}}_{\mathrm{P2P}}$ are a fully abstract model of higher-order session types, and the sub-typing $\preccurlyeq_{\mathsf{sbt}}$ (Theorem 8.4.9). This result justifies in a behavioural fashion the definition of $\preccurlyeq_{\mathsf{sbt}}$: within the theory of compliance, the sub-typing à la Gay and Hole not only *looks* natural, but it *is* natural, in the sense that there is no syntactical definition of $\sqsubseteq^{\mathsf{ho}}_{\mathrm{P2P}}$ other than the definition of $\preccurlyeq_{\mathsf{sbt}}$.

Now we shift our standpoint: the full abstraction result lets us think of higher-order session contracts as types, and of the relation $\sqsubseteq^{\mathsf{ho}}_{\mathrm{P2P}}$ as a sound sub-typing relation. The natural concern that arises from this shift, is whether session contracts can help us in advancing the existing type systems based on session types.

In this chapter we briefly address this issue. First we sketch a type system for a dialect of the pi-calculus, where types are session contracts. Afterwards, by means of the type system, we propose a way to ensure that the observable behaviour of well-typed processes enjoys certain properties. Our approach relies on the connection between the observable behaviour of types (i.e. session contracts) assigned to session end-points, and the behaviour of processes on these end-points (see Conjecture 9.2.18 and Example 9.2.19).

This chapter is merely exploratory and should be taken as a proof of concept. The conjectures that we state are currently under investigation, so the discussion is based on examples.

Structure of the chapter. In Section 9.1 we define the syntax and the reduction semantics of $\pi\mathsf{SC}$, a dialect the pi-calculus. Our version of the pi-calculus resembles the one of [Gay and Hole, 2005], but we use recursion instead of replication. In Section 9.2 we present a type discipline for $\pi\mathsf{SC}$, and discuss it in a series of examples. We also discuss the results that at present we are after. As this chapter contain no new results, we omit the related work section.

9.1 Pi-calculus with session contracts

In this section we define a dialect of the pi-calculus [Milner, 1999]. We assume a finite set of ground types, $\mathtt{BT}$; with the proviso that $\mathtt{BT}$ contains the types `Bool` and `Int`:

$$\{\mathtt{Bool}, \mathtt{Int}\} \subseteq \mathtt{BT}$$

The semantics of the type `Bool` is the set $\{\mathsf{true}, \mathsf{false}\}$, and this set provides the basic cases for the grammar of boolean expressions B (see Figure 9.1). The semantics of type `Int` is a finite subset of

the natural numbers, and provides the base elements for the arithmetic expressions A.

Further, we assume the existence of the following denumerable sets,

- a set of names $\mathcal{N}$, whose elements we range over with $a, b, c, \ldots, x, y, z, \ldots$
- a set of process variables, that we denote $\chi_1, \chi_2, \chi_3, \ldots$

The set $\mathtt{V}$ of *values* is defined as the union of the semantics of the ground types: $\mathtt{V} = \bigcup_{\mathtt{t} \in \mathtt{BT}} [\![\mathtt{t}]\!]$. We let u range over names and values ($\mathcal{N} \cup \mathtt{V}$), and $\mathtt{v}$ range over values ($\mathtt{V}$).

We use names to represent sessions, which are as private connections, characterised by *two* different and *complementary* end-points. To denote the end-points of a session a, we decorate a with *polarities*.

Definition 9.1.1. [Polarities]
The sets of polarities and of optional polarities are defined respectively as

$$\begin{array}{rcl} \mathtt{P} & = & \{-, +\} \\ \mathtt{O} & = & \{-, +, \epsilon\} \end{array}$$

We range over $\mathtt{P}$ with the symbol p, which denotes a *polarity*; and we range over $\mathtt{O}$ with the symbol o, which denotes an *optional* polarity. The complement of the (optional) polarity o, is denoted $\overline{o}$, and defined as follows,

$$\overline{o} = \begin{cases} + & \text{if } o = - \\ - & \text{if } o = + \\ \epsilon & \text{if } o = \epsilon \end{cases}$$

□

Let L_π be the language defined by the grammar in Figure 9.1. Also in this case we use the standard notions of capture avoiding substitution, depth, guarded terms and unfolding (see Section 2.1).

Definition 9.1.2. [π-processes with session contracts]
Let

$$\pi\mathsf{SC} = \{\, P \in L_\pi \mid P \text{ closed}, P\,\mathsf{gd} \,\}$$

We refer to the terms in $\pi\mathsf{SC}$ as *processes*. □

The type t that appears in the input construct in Figure 9.1 ranges over the set $\mathsf{SC}_{\text{HO}} \cup \mathtt{BT}$. We will discuss types in Section 9.2.

The set of free names of a term P, denoted $\text{FN}(P)$, is defined in the standard manner.

The non deterministic outputs in Figure 9.1 are not a standard construct, so we briefly comment on it.

Example 9.1.3. [Non-deterministic terms]
The non-deterministic sums let us render explicitly the non-determinism typical of choices. For the time being, we impose some restrictions on the syntax of these terms. For instance, if the names a and b are different, then the ensuing term is not a non-deterministic process,

$$a![\,l_1\,].\mathbf{0} \oplus b![\,l_2\,].\mathbf{0}$$

whereas the following term is a non-deterministic process

$$a![\,l_1\,].\mathbf{0} \oplus a![\,l_2\,].\mathbf{0}$$

because it is generated by the grammar for N_a. □

P, Q	::=		**Processes**
		$\mathbf{0}$	*Empty process*
		$u^p![\,v^o\,].P$	*Value output*
		N_u	*Nondeterministic output on u*
		$u^p?[\,x^o\colon t\,].P$	*Binding input*
		$u^p \rhd \{\,\mathtt{l_1}\colon P_1, \ldots, \mathtt{l_n}\colon P_n\,\}$	*Offer*
		$\text{IF}\,(B)\,\text{THEN}\,P\,\text{ELSE}\,Q$	*If then else*
		$(\nu a)\,P$	*Session creation*
		$P \mid\mid Q$	*Parallel composition*
		χ	*Process variable*
		$\mu\chi.\,P$	*Recursive process*
N_u	::=		**Non deterministic outputs**
		$u^p![\,\mathtt{l}\,].P$	*Label output*
		$N_u \oplus N_u$	*Non deterministic sum*
B	::=		**Boolean expressions**
		true	
		false	
		$u = u'$	
		$u > u$	
A	::=		**Arithmetic expressions**
		$1, 2, 3 \ldots$	*Natural numbers*

With the proviso that in the terms N_u (for every u) the labels are pair-wise distinct.

Figure 9.1: Grammar for processes

$P \mid\mid \mathbf{0} \equiv P$	[Par-Zero]
$P \mid\mid Q \equiv Q \mid\mid P$	[Par-Comm]
$P \mid\mid (Q \mid\mid R) \equiv (P \mid\mid Q) \mid\mid R$	[Par-Assoc]
$(\nu a)\,\mathbf{0} \equiv \mathbf{0}$	[Scp-Void]
$(\nu a)\,P \mid\mid Q \equiv (\nu a)\,(P \mid\mid Q)$, if a not free in Q	[Scp-Extr]
$(\nu a)\,(\nu b)\,P \equiv (\nu b)\,(\nu a)\,P$	[Scp-Flip]

Figure 9.2: Axioms for structural congruence

Reduction semantics We assume an evaluation relation $\Downarrow$ for the boolean expressions $\mathcal{B}$. The reduction semantics of the language depends on a structural congruence relation and on $\downarrow$. The structural congruence relation $\equiv$ is the *least* relation that satisfies the axioms in Figure 9.2, while the reduction semantics is the *least* relation $\longrightarrow$ that satisfies the rules in Figure 9.3.

From now on we will assume that on private names (i.e. session channels), only other private names are communicated. The intuition behind this assumption is that the names which are not private are public, thus known to everybody, and there is no need to communicate them.

9.1.1 Runtime errors

The reductions semantics tells us how to execute processes; the execution of processes, though, may stop because of some issues.

In general, we can think of a predicate $\longrightarrow_{\mathsf{err}}$, such that if $P \longrightarrow_{\mathsf{err}}$, then the computation of P cannot proceed: $P \overset{\tau}{\not\longrightarrow}$. If $P \longrightarrow_{\mathsf{err}}$ then we say that P *reduces into an error*.

Rather than defining $\longrightarrow_{\mathsf{err}}$, in this section we exhibit some archetypal processes that reduce into errors.

$$\frac{}{u^{o'}?[\,x^o\colon T\,].P \;||\; u^{\overline{o'}}![\,\mathtt{v}^o\,].Q \;||\; R \longrightarrow P\left\{{}^{\mathtt{v}^o}/_{x^o}\right\} \;||\; Q \;||\; R}\ [\text{A-Comm}]$$

$$\frac{}{u^p \rhd \{\,\mathtt{l}\colon P_1, \ldots, \mathtt{l_n}\colon P_n\,\} \;||\; u^{\overline{p}}![\,\mathtt{l_i}\,].Q \;||\; R \longrightarrow P_i \;||\; Q \;||\; R}\ [\text{A-Synch}]$$

$$\frac{}{\mu\chi.\,P \longrightarrow \text{UNF}(\mu\chi.\,P)}\ [\text{A-Unf}]$$

$$\frac{}{N_u \oplus N'_u \longrightarrow N_u}\ [\text{A-Inch-L}]$$

$$\frac{}{N_u \oplus N'_u \longrightarrow N'_u}\ [\text{A-Inch-R}]$$

$$\frac{}{\text{IF}\,(B)\ \text{THEN}\ P\ \text{ELSE}\ Q \longrightarrow P}\ B \Downarrow \mathsf{true};\ [\text{A-True}]$$

$$\frac{}{\text{IF}\,(B)\ \text{THEN}\ P\ \text{ELSE}\ Q \longrightarrow Q}\ B \Downarrow \mathsf{false};\ [\text{A-False}]$$

$$\frac{P \longrightarrow Q}{(\nu a)\,P \longrightarrow (\nu a)\,Q}\ [\text{R-Res}]$$

$$\frac{P \longrightarrow Q}{P \;||\; R \longrightarrow Q \;||\; R}\ [\text{R-Par}]$$

$$\frac{P' \longrightarrow Q'}{P \longrightarrow Q}\ P \equiv P', Q \equiv Q';\ [\text{R-Struct}]$$

Figure 9.3: Rules for the reduction semantics

In the language $\pi\mathsf{SC}$ there are terms that use names, variables, and polarities in ways that are not coherent with the intuitions behind the end-points of sessions. We show few of these "malformed" processes in the next example.

Example 9.1.4. [Malformed processes]
Observe the ensuing processes

$$\begin{aligned}
P_1 &= a^-![\,3\,].a^+![\,3\,].\mathbf{0}\\
P_2 &= a^-![\,3\,].\mathbf{0} \oplus a^+![\,3\,].\mathbf{0}\\
P_3 &= \text{IF}\,(v) = v'\,\text{THEN}\ a^-![\,true\,].\mathbf{0}\ \text{ELSE}\ a^+![\,false\,].\mathbf{0}\\
P_4 &= a^p \rhd \{\,\mathtt{l}_1\colon b^+![\,1\,].\mathbf{0}, \mathtt{l}_2\colon b^-![\,2\,].\mathbf{0}\,\}
\end{aligned}$$

Let us think of the end-points a^- and a^+ as resources. These two resources represents the two ends of a peer to peer connections, and are meant to let two *distinct* processes interact with each other. Thus a process that owns a^+ should not own a^- and vice-versa. Each one of the process above, on the contrary, owns both endpoints of the connection a, so we deem those terms as malformed. □

The issue shared by all the processes in the previous example, is that both end-points of a connection appear in them. This does not mirror the reality, as normally no program sends messages to itself. Moreover, the presence of both endpoints of a connection in a process means that the logic of the process changes as interactions take place; this appears not to mirror the reality, as the client of a connection and the server of the same connection have well distinguished logics.

Other errors that may take place during communications are type mismatches, mismatches in the polarities of names, and the non-linear usage of session endpoints.

Example 9.1.5. [Data mismatch]
Consider the following processes,

$$P = s?[\,y^-\colon ?\texttt{Int}.!\texttt{Bool}.\mathbf{0}\,].y^-?[\,x\colon \texttt{Int}\,].\textsc{if}\,(x > 0)\ \textsc{then}\ y^-![\,\mathsf{true}\,].\mathbf{0}\ \textsc{else}\ y^-![\,\mathsf{false}\,].\mathbf{0}$$
$$Q = (\nu a)\,(s![\,a^+\,].a^+![\,\mathsf{true}\,].\mathbf{0})$$

Intuitively, we should not accept the composition $Q \mid\mid P$ as a "good" process, as its reductions lead to the term

$$(\nu a)\,(\mathbf{0} \mid\mid \textsc{if}\,(\mathsf{true} > 0)\ \textsc{then}\ a![\,1\,].\mathbf{0}\ \textsc{else}\ a![\,0\,].\mathbf{0})$$

As it stands, it is not clear how to make sense of this term. The problem lies in the fact that the type of the formal parameters of $>$, is not the type of the actual parameter true. □

Example 9.1.6. [Polarity mismatches]
In Figure 9.3 rule [A-Comm] allows a communication to take place only if the polarity of the value sent, $\mathtt{v}^o$, matched the polarity at which the value is expected, x^o.

On the one hand, this ensures complementary end-points cannot be mixed because of communication, and show explicitly which one of the end-points of a session is sent, and which can be received. On the other hand, the requirement that the polarities match may let processes reduce into an error. The following composition is an example of the phenomenon,

$$a^-![\,b^-\,].b^+?[\,y\colon \texttt{Int}\,].P \mid\mid a^+?[\,x^+\colon !\texttt{Int}.\sigma\,].x^+![\,3\,].Q$$

Even though the process on the left is willing to perform an output via the end-point a^-, and the process on the right is ready to read via the end-point a^+, the overall composition is stable, because the polarity of b^- does not match with the polarity of x^+, so rule [A-Comm] cannot be applied. □

Example 9.1.7. [Linearity of session channels]
Consider the processes

$$P = s?[\,x^-\colon ?(?\texttt{Int}.\,1).\,1\,].x^-?[\,y\colon \texttt{Int}\,].\mathbf{0}$$
$$Q = (\nu a)\,(s![\,a^-\,].(a^+![\,3\,].\mathbf{0} \mid\mid a^+![\,3\,].\mathbf{0}))$$

up-to structural equivalence, the composition $P \mid\mid Q$ reduces to the term

$$S = (\nu a)\,(a^+![\,3\,].\mathbf{0})$$

which is *not* structurally equivalent to $\mathbf{0}$, and is stable. The problem here amounts to the fact that in Q there are two threads communicating with P, and this leads to a sort of "misalignment" between the session as seen by P and the session as seen by the two threads in Q. In particular, when P has finished communicating, one of the processes in Q is "left behind", and is still expecting to perform an output on a^+. □

9.2 Type system

Let $\mathtt{P}$ denote the set of polarities, Let $\mathtt{O}$ denote the set of optional polarities. Let $\mathcal{T} = \mathtt{O} \times (\mathsf{SC}_{\text{HO}} \cup \mathtt{BT})$. Types T are pairs in the set $\mathcal{T}$, while the set $\mathsf{SC}_{\text{HO}} \cup \mathtt{BT}$ is ranged over by t. Let $\mathsf{pol}((p,t)) = p$ and let $\mathsf{body}((p,t)) = t$.

Type environments We denote with Γ the *functions* from polarised names to types,

$$\Gamma : (\mathcal{N} \times \mathtt{P}) \longrightarrow \mathcal{T}$$

with the additional requirement that the polarities of the names be equal to the polarities of the relative types. Formally

$$T = \Gamma(a^p) \text{ implies } \mathsf{pol}(T) = p \tag{9.1}$$

Example 9.2.1. The following relation is not a Γ, because it is not a function

$$\mathcal{R} = \{(a^-, (-, !\mathtt{1}.\,\mathtt{1})), (a^-, (-, \mathtt{1}))\}$$

The following relation is not a Γ because it does not satisfy Eq. (9.1),

$$\{(a^-, (+, !\mathtt{1}.\,\mathtt{1}))\}$$

□

We denote with Δ the *relations* from optionally polarised names to types, that is

$$\Delta \subseteq (\mathcal{N} \times \mathtt{0}) \times \mathcal{T}$$

with the additional proviso that the Δ's satisfy the condition in Eq. (9.1). The set Δu^o is defined as $\{\, T \mid (u^o, T) \in \Delta \,\}$.

Example 9.2.2. The relation $\mathcal{R}$ defined In the previous example we defined a relation $\mathcal{R}$ which is not a Γ; that relation, though, is a valid Δ. The following relation is not a Δ because it does not satisfy Eq. (9.1),

$$\{(a^-, (+, !\mathtt{1}.\,\mathtt{1}))\}$$

□

Environments for process variables We need a last ingredient in our type environments. We denote with Z any *function* from process variables $\chi, \chi', \ldots$ to pairs $\Gamma; \Delta$. We write $Z - \chi$ to denote the function $Z \setminus \{(\chi, Z(\chi))\}$.

Definition 9.2.3. [Type environment]
We refer to the triple $Z; \Gamma; \Delta$ as *type environment*. Moreover, we let E denote the set of type environments. □

The purpose of having a set Γ and a set Δ is that we are going to put names and variables to be treated in a linear manner in Γ, and the other ones in Δ.

Notation We let $a^o : T$ denote the pair $(a^o, (o, t))$; so, for instance, instead of writing

$$\{(a^-, (-, !\mathtt{1}.\,\mathtt{1})), (a^-, (-, \mathtt{1}))\}$$

we will write

$$\{a^- : \mathtt{1}.\,\mathtt{1}, a^- : \mathtt{1}\}$$

We also let $dom(\Gamma; \Delta) = dom(\Gamma) \cup dom(\Delta)$, and let the symbol $=_{\text{P2P}}$ denote equivalence generated by the pre-order $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$. We lift the relation $=_{\text{P2P}}$ from session contracts to environments in the following way. For every sets of types A and B we write $A =_{\text{P2P}} B$ whenever

1. $(u^o, t) \in A$ if and only $(u^o, t') \in B$
2. $(u^o, t) \in A$ and $(u^o, t') \in B$ imply that either $t =_{\text{P2P}} t'$ or $t = t'$

We stipulate that $\Gamma; \Delta =_{\text{P2P}} \Gamma'; \Delta'$ whenever $\Gamma =_{\text{P2P}} \Gamma'$ and $\Delta =_{\text{P2P}} \Delta'$.

Definition 9.2.4. [Completed]
We write $Z;\Gamma\text{Completed}$ and say that $Z;\Gamma$ is *completed*, if $u^p \in dom(\Gamma)$ implies that one of the following conditions is true,

a) $u^p \in dom(Z(\chi))$ for some χ

b) $\Gamma(u^p) = 1$. □

Intuitively, a type environment is completed if the communications on the end-points of the sessions terminated (i.e. have been completed), and Z accounts for the end-points on which the communications are not terminated.

We will need an operation to manipulate in a sound way the linear environment Γ. To this end we define $+$.

Definition 9.2.5. The addition of a typed name to an environment is defined by

$$\Gamma + \{u^p : T\} = \Gamma \cup \{u^p : T\}$$

if $u^p \notin dom(\Gamma)$, and is undefined in all other cases. □

Definition 9.2.6. [Type relation]
Let $\mathcal{F}_{\vdash} : \mathcal{P}(E \times L_\pi) \longrightarrow \mathcal{P}(E \times L_\pi)$ be the rule functional given by the inference rules in Figure 9.4. Lemma C.0.38 and the Knaster-Tarski theorem ensure that there exists the least solution of the equation $X = \mathcal{F}_{\vdash}(X)$; we call this solution the *type relation*, and we denote it $\vdash$: That is $\vdash = \mu X.\mathcal{F}_{\vdash}(X)$. □

Type judgements have the following form

$$Z;\Gamma;\Delta \vdash P$$

While Γ and Δ are the environments that associate types to the names and the values that appear in P, Z expresses how the free process variables of P, once substituted, will be typed. We will show the role played by Z in a series of examples about typing recursive processes.

Almost all the inference rules in Figure 9.4 are standard. The only ones that need explanation are [T-NonD-L], and the rules for recursive terms.

The two rules [T-NonD-L] together are meant to to generalise the usual rules for the processes that perform a choice; see for instance rule [T-Choose] of [Gay and Hole, 2005],

$$\frac{l = l_i \in \{\,\mathtt{l_1}, \ldots, \mathtt{l_n}\,\} \quad \Gamma, x^p : T_i \vdash P}{\Gamma, x^p : \ \oplus\ \langle\, \mathtt{l_i} : T_i \,\rangle_{1 \le i \le n} \vdash x^p \lhd \mathtt{l}.P}\ \text{[T-Choose]} \tag{9.2}$$

The rule in Eq. (9.2) can type processes that choose only one label. The rules [T-NonD-L], [T-Out-L] generalise [T-Choose] in the sense that they can type processes that can choose different labels.

Example 9.2.7. [Multiple choices]
Consider the following processes,

$$\begin{aligned} P' &= a^+![\,\mathtt{sign}\,].a^+![\,3\,].a^+?[\,x : \mathtt{Int}\,].\mathbf{0} \\ P &= a^+![\,\mathtt{close}\,].\mathbf{0} \oplus P' \end{aligned}$$

The process P expects to be offered a choice along channel a, by some peer that owns a^-; P is free to choose either to close the connection, or the ask to invoke the function `sign`, send the parameter

$$\frac{}{Z;\Gamma;\Delta \vdash \mathbf{0}}\ Z;\Gamma \textsc{Completed};\ [\textsc{T-Void}]$$

$$\frac{Z;\Gamma + \{a^p\colon \sigma, a^{\overline{p}}\colon \sigma'\};\Delta \vdash P}{Z;\Gamma;\Delta \vdash (\nu a)\,P}\ \sigma \dashv_{\textsc{p2p}} \sigma';\ [\textsc{T-Res}]$$

$$\frac{Z;\Gamma_1;\Delta \vdash P \quad Z;\Gamma_2;\Delta \vdash Q}{Z;\Gamma;\Delta \vdash P \mid\mid Q}\ \Gamma = \Gamma_1 + \Gamma_2;\ [\textsc{T-Par}]$$

$$\frac{Z;\Gamma;\Delta \vdash P \quad Z;\Gamma;\Delta \vdash Q}{Z;\Gamma;\Delta \vdash \textsc{if}\,(B)\ \textsc{then}\ P\ \textsc{else}\ Q}\ [\textsc{T-If}]$$

Rules for branches and choices

$$\frac{Z;\Gamma + \{u^p\colon \sigma_1\};\Delta \vdash P_1 \quad \ldots \quad Z;\Gamma + \{u^p\colon \sigma_n\};\Delta \vdash P_m}{Z;\Gamma + \{u^p\colon \textstyle\sum_{i\in[1;m]} ?\mathtt{l_i}.\sigma_i\};\Delta \vdash u^p \rhd \{\,\mathtt{l}_1\colon P_1, \ldots, \mathtt{l}_n\colon P_n\,\}}\ 1 \le m \le n;\ [\textsc{T-Branch}]$$

$$\frac{Z;\Gamma + \{u^p\colon \sigma_1\};\Delta \vdash P_1 \quad \ldots \quad Z;\Gamma + \{u^p\colon \sigma_n\};\Delta \vdash P_n}{Z;\Gamma + \{u^p\colon \bigoplus_{i\in[1;n]} !\mathtt{l_i}.\sigma_i\};\Delta \vdash u^p![\,\mathtt{l}_1\,].P_1 \oplus \ldots \oplus u^p![\,\mathtt{l}_n\,].P_n}\ [\textsc{T-NonD-L}]$$

Input and output of values

$$\frac{Z;\Gamma + \{x^p\colon \sigma', y^q\colon \sigma\};\Delta \vdash P}{Z;\Gamma + \{x^p\colon ?(\sigma'').\sigma'\};\Delta \vdash x^p?[\,y^q\colon \sigma\,].P}\ \sigma'' \sqsubseteq^{\mathsf{ho}}_{\textsc{p2p}} \sigma;\ [\textsc{T-In-Ss}]$$

$$\frac{Z;\Gamma + \{x^p\colon \sigma\};\Delta \cup \{y\colon \mathtt{t}\} \vdash P}{Z;\Gamma + \{x^p\colon ?\mathtt{t}'.\sigma\};\Delta \vdash x^p?[\,y\colon \mathtt{t}\,].P}\ \mathtt{t}' \preccurlyeq_{\mathtt{b}} \mathtt{t};\ [\textsc{T-In-Sv}]$$

$$\frac{Z;\Gamma;\Delta \cup \{y\colon \mathtt{t}, x\colon \sigma\} \vdash P}{Z;\Gamma;\Delta \cup \{x\colon ?\mathtt{t}'.\sigma\} \vdash x?[\,y\colon \mathtt{t}\,].P}\ \mathtt{t}' \preccurlyeq_{\mathtt{b}} \mathtt{t};\ [\textsc{T-In-Uv}]$$

$$\frac{Z;\Gamma + \{x^p\colon \sigma\} \vdash P}{Z;\Gamma' + \{x^p\colon !(\sigma'').\sigma, u^{p'}\colon \sigma'\};\Delta \vdash x^p![\,u^{p'}\,].P}\ \sigma' \sqsubseteq^{\mathsf{ho}}_{\textsc{p2p}} \sigma'';\ [\textsc{T-Out-S}]$$

$$\frac{Z;\Gamma' + \{x^p\colon \sigma\};\Delta \vdash P}{Z;\Gamma' + \{x^p\colon !(\mathtt{t}').\sigma\};\Delta \cup \{\mathtt{v}\colon \mathtt{t}\} \vdash x^p![\,\mathtt{v}\,].P}\ \mathtt{t} \preccurlyeq_{\mathtt{b}} \mathtt{t}';\ [\textsc{T-Out-V}]$$

Rules for recursive terms

$$\frac{}{Z;\Gamma;\Delta \vdash \chi}\ \chi \in dom(Z),\ Z;\Gamma \textsc{Completed};\ [\textsc{T-Var}]$$

$$\frac{Z;Z(\chi) \vdash P}{Z - \chi;\Gamma;\Delta \vdash \mu\chi.\,P}\ Z(\chi) =_{\textsc{p2p}} \Gamma;\Delta;\ [\textsc{T-Rec}]$$

Figure 9.4: Inference rules for the rule functional $\mathcal{F}_{\vdash}$. The polarities of types are understood; they coincide with the polarities of the names that types are assigned to

for the function, and read the outcome. Let $\sigma_1 = !\texttt{close}.\texttt{1}$ and $\sigma_2 = !\texttt{sign}.!\texttt{Int}.?\texttt{Int}.\texttt{1}$ The inference tree below sketched the type derivation for P; note that we omit Z as P contains no process variable,

$$\dfrac{\dfrac{}{\{u^+\colon \texttt{1}\};\{\texttt{3}\colon \texttt{Int}\} \vdash \mathbf{0}}\ [\text{T-Void}] \quad \dfrac{\dfrac{}{\{u^+\colon \texttt{1}\};\emptyset \vdash \mathbf{0}}\ [\text{T-Void}]}{\{u^+\colon ?\texttt{Int}.\texttt{1}\};\emptyset \vdash a^+?[x\colon \texttt{Int}].\mathbf{0}}\ [\text{T-In-Sv}]}{\{u^+\colon !\texttt{Int}.?\texttt{Int}.\texttt{1}\};\{\texttt{3}\colon \texttt{Int}\} \vdash a^+![\texttt{3}].a^+?[x\colon \texttt{Int}].\mathbf{0}}\ [\text{T-Out-V}]}{\{u^+\colon \sigma_1 \oplus \sigma_2\};\{\texttt{3}\colon \texttt{Int}\} \vdash P}\ [\text{T-NonD-L}]$$

□

In a series of examples we show how the rules for recursive terms and process variables are meant to be used. The axiom [T-Var] and the rule [T-Rec] are inspired by the type discipline of [Honda et al., 1998] and [Demangeon and Honda, 2011]. Observe that in the side condition of [T-Rec], the equality $=_{\text{P2P}}$ allows us to unfold the session contracts in the environments.

Example 9.2.8. [Typing a recursive process]
Let $P = \mu\chi.\, a^p![\texttt{3}].a^p![\texttt{1}].\chi$, and let

$$\begin{array}{lcl} \sigma & = & \mu x.\, !\texttt{Int}.x \\ \Delta & = & \{\texttt{3}\colon \texttt{Int}, \texttt{1}\colon \texttt{Int}\} \\ Z(\chi) & = & \{a^p\colon !\texttt{Int}.!\texttt{Int}.\sigma\};\Delta \end{array}$$

Intuitively, the behaviour shown by the process P on the channel a^p is described by σ. This is formalised by the following type derivation.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{Z;\{a^p\colon \sigma\};\emptyset \vdash \chi}\ [\text{T-Var}]}{Z;\{a^p\colon \sigma\};\{\texttt{1}\colon \texttt{Int}\} \vdash a^p![\texttt{1}].\chi}\ [\text{T-Out-Sv}]}{Z;\{a^p\colon !\texttt{Int}.!\texttt{Int}.\sigma\};\Delta \vdash a^p![\texttt{3}].a^p![\texttt{1}].\chi}\ [\text{T-Out-Sv}]}{Z;\{a^p\colon \sigma\};\Delta \vdash P}\ Z(\chi) =_{\text{P2P}} \{a^p\colon \sigma\};\Delta\ [\text{T-Rec}]$$

□

Example 9.2.9. [Typing nested recursion]
Let $P = \mu\chi.\, a^- \rhd \{\,\texttt{neg}\colon a^-?[x\colon \texttt{Bool}].a^-![\mathit{not}\ x].\chi,\ \texttt{k}\colon \mu\chi'.\, a^-![\texttt{3}].\chi'\,\}$. The process P recursively offers two choices to the peer that owns the end-point a^+. The label `neg` represents the negation function, and P after `neg` interacts on a^+ accordingly. The label `k` represents the choice of a constant, so P after `k` sends a constant number.

Now let

$$\begin{array}{lcl} \sigma' & = & \mu y.\, !\texttt{Int}.y \\ \sigma & = & \mu x.\, ?\texttt{neg}.?\texttt{Bool}.!\texttt{Bool}.x + !\texttt{k}.\sigma' \\ \Delta & = & \{\texttt{3}\colon \texttt{Int}\} \\ \Gamma & = & \{a^-\colon \sigma\} \\ Z(\chi) & = & \{a^-\colon \text{UNF}(\sigma)\};\Delta \\ Z(\chi') & = & \{a^-\colon \text{UNF}(\sigma')\};\Delta \end{array}$$

The following inference tree shows how to type P. Note that $Z(\chi) =_{\text{P2P}} \Gamma;\Delta$, so we can apply rule

[T-UNF] at the bottom of the tree.

$$\dfrac{\dfrac{\dfrac{\overline{Z;\{a^-:\sigma'\};\emptyset\vdash\chi'}\ [\text{T-VAR}]}{Z;\{a^-:\,!\texttt{Int}.\sigma'\};\emptyset\vdash\chi'}\ \texttt{Int}\preccurlyeq_{\mathfrak{b}}\texttt{Int};\ [\text{T-OUT-SV}]}{Z;\{a^-:\sigma'\};\Delta\vdash\mu\chi'.\,a^-![3].\chi'}\ Z(\chi')=_{\text{P2P}}\{a^-:\sigma'\};\Delta\ [\text{T-REC}]}{(A)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{Z;\{a^-:\sigma\};\emptyset\vdash\chi}\ [\text{T-VAR}]}{Z;\{a^-:\,!\texttt{Bool}.\sigma\};\{x:\texttt{Bool}\}\vdash a^-![not\ x].\chi}\ \texttt{Bool}\preccurlyeq_{\mathfrak{b}}\texttt{Bool};\ [\text{T-OUT-SV}]}{Z;\{a^-:\,?\texttt{Bool}.!\texttt{Bool}.\sigma\};\Delta\vdash a^-?[x:\texttt{Bool}].a^-![not\ x].\chi}\ [\text{T-IN-SV}] \qquad \dfrac{\vdots}{(A)}}{Z;\{a^-:\,\text{UNF}(\sigma)\};\Delta\vdash a^-\rhd\{\texttt{neg}:\,a^-?[x:\texttt{Bool}].a^-![not\ x].\chi,\texttt{k}:\,\mu\chi'.\,a^-![3].\chi'\}}\ [\text{T-BRANCH}]}{Z;\Gamma;\Delta\vdash P}\ [\text{T-REC}]$$

□

In the next example we will see how to type a process that recursively reads a name and interacts over it.

Example 9.2.10. [Recursively read name]
The process that we want to type in this example is $P=\mu\chi.\,z^+?[y^-:\sigma].y^-![3].\chi$, where $\sigma=\mu x.\,?(\rho).x$ and $\rho=!\texttt{Int}.1$. In the process P the end-point y^- depends on the name x^+, for y^- it is read through x^+. The recursive behaviour is shown on the "outermost" channel x^+. The variable y^- is bound afresh at each recursive loop, so the end-point that replaces y^- is used only for a finite communication.

Let

$$\begin{array}{lcl}\Gamma & = & \{z^+,\sigma\}\\ \Gamma' & = & \{z^+,?(\rho).\sigma\}\\ \Delta & = & \{3:\texttt{Int}\}\\ Z(\chi) & = & \Gamma';\Delta\end{array}$$

Note that $\Gamma=_{\text{P2P}}\Gamma'$. The inference tree that types P is the following one,

$$\dfrac{\dfrac{\dfrac{\overline{Z;\{z^+:\,\mu x.\,?(\rho).x;y^-:\,1\};\emptyset\vdash\chi}\ [\text{T-VAR}]}{Z;\{z^+:\,\mu x.\,?(\rho).x;y^-:\,!\texttt{Int}.1\};\Delta\vdash y^-![3].\chi}\ [\text{T-OUT-SV}]}{Z;\Gamma';\Delta\vdash z^+?[y^-:\sigma].y^-![3].\chi}\ !\texttt{Int}.\sigma\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}\sigma;\ [\text{T-IN-SS}]}{Z;\Gamma;\Delta\vdash P}\ Z(\chi)=_{\text{P2P}}\Gamma;\Delta\ [\text{T-REC}]$$

□

In order to prove the valuable properties of the type system based on higher-order session contracts, we have to establish the normal results of type systems.

Lemma 9.2.11. [Free names]
If $Z;\Gamma;\Delta\vdash P$, then $x^p\in dom(\Gamma)$ implies $u^p\in\text{FN}(P)$ or $\Gamma(u^p)=1$ or $u^p\in dom(Z(P))$.

As the symbol T ranges over ground types and session contracts, we have to state the substitution lemma with a case analysis.

Conjecture 9.2.12 (Substitution lemma). *If $Z;\Gamma+\{x^o:T_2\};\Delta\vdash P$, and $\Gamma+\{u^o:T_1\}$ is defined, then*

1. *$\mathit{body}(T_1)\preccurlyeq_{\mathfrak{b}}\mathit{body}(T_2)$ implies that $Z;\Gamma+\{u^o:T_1\};\Delta\vdash P\left\{{}^{u^o}/_{x^o}\right\}$*

2. $\mathit{body}(T_1) \sqsubseteq^{\mathsf{ho}}_{\mathrm{P2P}} \mathit{body}(T_2)$ *implies that* $Z; \Gamma + \{u^o : T_1\}; \Delta \vdash P \left\{ {}^{u^o}/_{x^o} \right\}$

Lemma 9.2.13. The operation + affords the following properties

a) $\Gamma = \Gamma + \emptyset$

b) $\Gamma + \Gamma' = \Gamma' + \Gamma$

c) $(\Gamma + \Gamma') + \Gamma'' = \Gamma + (\Gamma' + \Gamma'')$

The previous lemma is necessary to prove the following fact.

Lemma 9.2.14. [Compatibility with $\equiv$]
If $Z; \Gamma; \Delta \vdash P$ and $P \equiv Q$, then $Z; \Gamma; \Delta \vdash Q$.

Conjecture 9.2.15 (Subject reduction). *If* $Z; \Gamma; \Delta \vdash P$ *and* $P \longrightarrow Q$ *then* $\Gamma'; \Delta' \vdash Q$ *for some* $Z'; \Gamma'; \Delta'$ *such that* $dom(\Gamma') \subseteq dom(\Gamma)$, $dom(\Delta') \subseteq dom(\Delta)$, *and* $dom(Z') \subseteq dom(Z)$.

9.2.1 Conjectures

Recall from Section 9.1.1 the idea of runtime error and the symbol $\longrightarrow_{\mathsf{err}}$. One of the properties that we wish to prove for the proposed type system (possibly amending the rules), is type safety.

Conjecture 9.2.16 (Type safety). *For every* $P \in \pi\mathsf{SC}$, *if* $Z; \emptyset; \emptyset \vdash P$, *then* $P \not\longrightarrow_{\mathsf{err}}$.

It is well-known that session types guarantee type safety, so to prove Conjecture 9.2.16 we expect to use standard techniques, and we do not consider such a result a novelty.

In Section 9.1.1 we have shown in a series of examples few processes that reduce into errors. Let us revisit one of those examples.

Example 9.2.17. Consider the processes P and Q of Example 9.1.5; we argued there that the composition $P \mid\mid Q$ reduces to a term that does not make sense, so $P \mid\mid Q \longrightarrow_{\mathsf{err}}$. According to Conjecture 9.2.16, we should not be able to type check $P \mid\mid Q$. In this example we argue that this is indeed the case.

The type discipline shows that the composition above should be regarded as a badly formed process, for it cannot be typed.

Intuitively, we can provide types the channels of P,[1]

$$\{y^- : ?\mathtt{Int}.!\mathtt{Bool}.\mathtt{1}\}; \{s : ?(?\mathtt{Int}.!\mathtt{Bool}.\mathrm{END}).\mathtt{1}\} \vdash P$$

and similarly for the other peer Q,

$$\{a^+ : !\mathtt{Bool}.\mathtt{1}\}; \{s : !(!\mathtt{Bool}.\mathtt{1}).\mathtt{1}\} \vdash Q$$

Now we see clearly the mismatch between the way in which P uses y, and the way in which Q uses a; there are *two* kinds of mismatches;

1) the first mismatch is that on y a datum at type `Int` is read, whereas on a a datum of type `Bool` is written

2) the second mismatch is that after the tentative input on y^-, P is willing to write on y^-; the process Q, on the contrary, after the output on a^+ stops using a^+.

These mismatches do not allow us to type check the composition $P \mid\mid Q$.

A similar argument can be applied to the process $P \mid\mid Q$ of Example 9.1.7. The composition $P \mid\mid Q$ cannot be typed using session contracts, because rule [T-RES] cannot be applied to it, so it is ruled out as a malformed program; indeed $P \mid\mid Q \longrightarrow_{\mathsf{err}}$ □

[1] We omit Z as there are no process variables in P and Q.

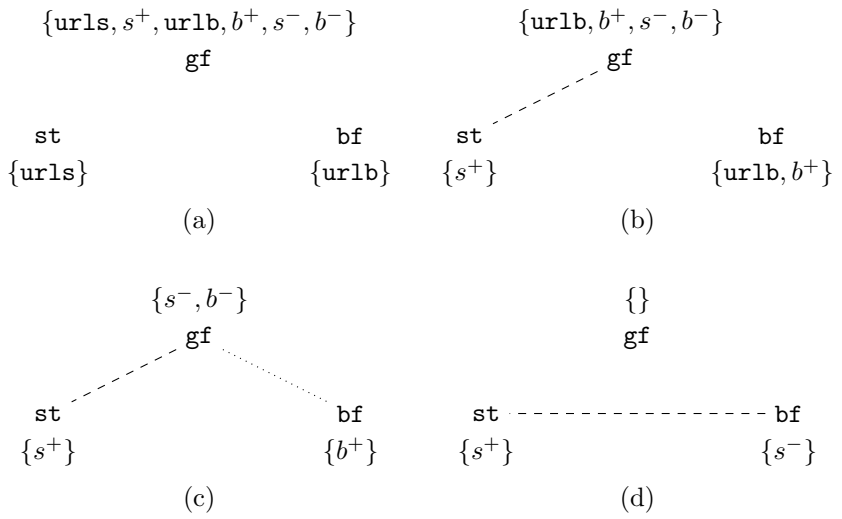


Figure 9.5: Evolution of the states of the sessions, during the computation described in Example 9.2.19

What we believe to be the main advance that session contracts may lead to, is the verification of behavioural properties of processes.

Protocol specification

Session contracts are equipped with an operational semantics similar to the one of CCS. If we specify a communication protocol by using a session contract σ, thanks to the semantics, we can check whether the protocol enjoys given properties: by writing a property as a formula φ of the Hennessy-Milner Logic [Aceto et al., 2007], we can prove whether $\sigma \models \varphi$. If this is the case, then the protocol specified by σ enjoys the property φ.

Roughly speaking, the typing rules in Figure 9.4 ensure that if $Z;\Gamma;\Delta \vdash P$, and a^o appears in Γ, then there is relation between the behaviour of $\Gamma(a^o)$ and the behaviour of P on the name a^o.

We are currently investigating the relation between the behaviours of types and the behaviours of processes, so as to define a function lift suitable to prove the following theorem.

Conjecture 9.2.18 (Behavioural properties of processes via types)**.**
For every $P \in \pi\mathsf{SC}$ such that $\text{FN}(P) = \{a\}$, for some $a \in \mathcal{N}$, if $Z;\Gamma;\Delta \vdash P$ and $\Gamma(a^o) \models \varphi$, then $P \models \mathsf{lift}(\varphi, a^o)$.

The intuitive meaning of Conjecture 9.2.18 is that under certain conditions, if P is typed by a triple $Z;\Gamma;\Delta$, then the behavioural properties of the session contracts in Γ, should imply analogous properties, "lifted" to the behaviour of processes.

If we think of session contracts as protocol specifications, then we can take processes of $\pi\mathsf{SC}$ to be protocol implementations. Conjecture 9.2.18 then guarantees that if a specification σ enjoys a property φ, then the implementations of σ enjoy $\mathsf{lift}(\varphi, a^o)$ for some polarised name α. Such a result provides a way to ensure that the way in which programs communicate over peer to peer connections adhere to formally specified properties $\varphi_1, \varphi_2, \ldots$.

We conclude this chapter motivating Conjecture 9.2.18 by means of an example.

Example 9.2.19. In this example we show how the observable behaviour that processes show on sessions a, b, c, ... is described by the session contracts that type the end-points of those sessions (i.e. a, b, c, ...).

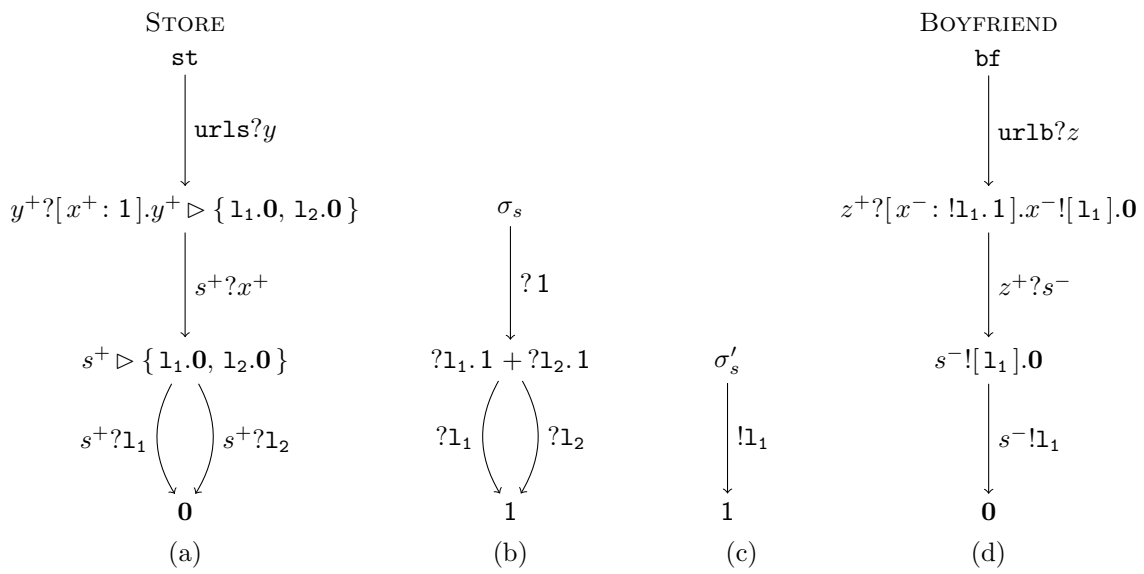


Figure 9.6: Observable behaviours of (1) the processes in Example 9.2.19, and of (2) the session contracts that type the session end-points used by those processes.

We assume two public names, `urls` and `urlb`, and we define three processes: a girlfriend `gf`, her boyfriend, `bf`, and a store, `store`.

$$\begin{array}{lcl} \texttt{gf} & = & (\nu b)\,((\nu s)\,(\texttt{urls}![\,s\,].s^-![\,\kappa\,].\texttt{urlb}![\,b^+\,].b^-![\,s^-\,].\mathbf{0})) \\ \texttt{bf} & = & \texttt{urlb}?[\,z^+\colon \sigma_b\,].z^+?[\,x\colon !\mathtt{l}_1.\,1\,].x^-![\,\mathtt{l}_1\,].\mathbf{0} \\ \texttt{st} & = & \texttt{urls}?[\,y^+\colon \sigma_s\,].y^+?[\,x\colon 1\,].y^+ \rhd \{\,\mathtt{l}_1.\mathbf{0}, \mathtt{l}_2.\mathbf{0}\,\} \end{array}$$

where

$$\begin{array}{lcl} \sigma_s & = & ?(1).(\sigma'_s + ?\mathtt{l}_2.\,1) \\ \sigma'_s & = & ?\mathtt{l}_1\,1 \\ \sigma_b & = & ?(!\mathtt{l}_1.\,1)\,1 \end{array}$$

In Figure 9.5, we depict how the state of the session end-points evolve during the execution of the composition $\texttt{gf} \mid\mid \texttt{bf} \mid\mid \texttt{st}$. There each process is decorated with the set of names that it owns.

The girlfriend wishes to give a present to her boyfriend, and the present has to be bought in a store that only the girlfriend has access to. In Figure 9.5 (a) represents the initial state, in which no session have been created. The girlfriend begins a session s with the store; the store gets the server end-point s^+, whereas the girlfriend gets the client end-point s^- (see (b) in Figure 9.5). The communication between the girlfriend and the boyfriend takes place on a different session, b, whose server end-point b^+ is used by the boyfriend (see (c)). After having sent on s a name κ that the boyfriend is not aware of (for instance some credentials), the girlfriend passes its endpoint, s^-, to the boyfriend (see (d)), that uses it to choose a gift.

We can infer the following typing judgement (we omit Z as there are no process variables)

$$\Gamma; \Delta \vdash (\nu b)\,((\nu s)\,(\texttt{gf} \mid\mid \texttt{st}) \mid\mid \texttt{bf})$$

where $\Delta = \{\texttt{urlb}\colon \sigma_b, \texttt{urls}\colon \sigma_s, \kappa\colon 1\}$.

In Figure 9.6 we depict the behaviour shown by the store and the boyfriend on the session s (respectively in column (a) and (d)). The behaviour of σ_s is depicted as well, in column (b), while column (c) shows the behaviour of σ'_s. Observe how the behaviour of σ_s describes exactly how

the composition `gf` || `bf` || `st` acts upon s^+. In the LTS of σ_s, the first transition is due to the communication of credentials κ between `st` and `gf`, while the second transition is due to the choices that `st` offers to `bf`.

Since s^- is communicated to `bf` only after a part of the interaction on the session has been realised, the session contract σ'_s in `bf` describes only the remaining part of the interaction pattern that is to be performed. Indeed, `bf` performs this part of the interaction with the the store, `st`, showing exactly the behaviour described by σ'_s. □

In this chapter we have sketched the research line that we are following at the moment. In particular, in Conjecture 9.2.18 we have stated what may be the chief advantage brought by session contracts into type systems for the pi-calculus.

Chapter 10

Literature Review

Chapter 2

The *Calculus of Communicating Systems* is a well established formalism to reason on the interactions performed by concurrent software. Well-known books on CCS are [Milner, 1989, 1999].

Milner presented the CCS in 1980, as a result of a research line started with the use of "processes" in [Milner, 1975], and pursed in [Milne and Milner, 1979] by introducing "communicating processes".

In the language of CCS internal computations are represented by the action τ. Consider the processes $\tau.b$ and $a + \tau.b$. The process $\tau.b$ performs some internal computation (for instance it chooses a branch of an IF statement) and then becomes ready to interact on b. The process $a + \tau.b$ either engages in the interaction a, or it performs some computation, and then becomes ready to perform b. The presence of the τ action creates some issues, as it makes relations such as the MUST pre-order not to be pre-congruences:

$$b \quad \sqsubseteq_{\text{MUST}} \quad \tau.b \qquad\qquad a + b \quad \not\sqsubseteq \text{MUST} \quad a + \tau.b$$

The inequality above is from [De Nicola and Hennessy, 1984, Page 92].

In view of the issue due to the action τ in the syntax, in 1987 De Nicola and Hennessy presented the CCS *without* τ*'s* ($\text{CCS}_{\text{w}\tau}$). The syntax of $\text{CCS}_{\text{w}\tau}$ has the sum $\oplus$ to express internal computation, and does not allow us to write τ, thereby solving the problem shown above.

Session contracts emerged recently within the field of "contracts for web services" (see next paragraph). Laneve and Padovani in 2008 have been the first to attempt to model with a compliance pre-order the sub-typing à la Gay and Hole. They focused on a sublanguage of first-order session types without input/output constructs. Barbanera and de'Liguoro later on (2010) exhibited a sound model of the sub-typing à la Gay and Hole tailored to the whole language of first-order session types. Their model uses session *behaviours*, which are too general to be interpreted into first-order session types. To prove also the completeness of the model suggested by Barbanera and de'Liguoro, in [Bernardi and Hennessy, 2012] we restricted the language of session behaviours, introducing session contracts, and we proved that the model is indeed fully abstract.

Chapter 3

The theories of compliance that appeared in the literature use versions of CCS as object language, and the terms of the chosen language are usually referred to as "contracts", or "contracts for web services". Hereafter we will use the word "contract" with this meaning.

The first compliance relation appeared in the literature is given in [Carpineti et al., 2006, Definition 4], which improves on the thesis [Carpineti, 2007].

The approach of Carpineti et al. to define the compliance is the converse of ours (and of subsequent

works). We first introduce the compliance, $\dashv$, and then define the refinements generated from it; whereas Carpineti et al. first define a *subcontract* relation (Definition 2), denoted $\preceq$, then, by means of a syntactic notion of dual contract, define the compliance: ρ complies with σ if only if $\overline{\rho} \preceq \sigma$. This style of definition has been dropped in the subsequent works.

The original idea of compliance has been reworked in [Laneve and Padovani, 2007], which presents a *behavioural compliance*. In that paper a testing-like approach has been taken, thereby showing some connections between the testing theory and the compliance theory. Laneve and Padovani first define the LTS

$$\langle\, \mathsf{CCS}^{\mathsf{rec,fb}}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$$

where $\mathsf{CCS}^{\mathsf{rec,fb}}_{\mathrm{w}\tau}$ is essentially a finite branching version of $\mathsf{CCS}_{\mathrm{w}\tau}$ with recursion, and the transitions are given by standard inference rules.

Then the authors define the language of constrained contracts (see Section 3.4) and the behavioural compliance, which is defined in terms of computations performed by clients and servers, deadlocks, and the action $\checkmark$ (see Definition 1 in that paper).

The compliance relation has been further studied, and the most recent accounts of it are given in [Castagna et al., 2009; Padovani, 2010]. Also these papers follow the testing-like approach.

Thus far, the only presentation of the compliance that treats divergence explicitly is [Laneve and Padovani, 2007], while Castagna et al. and Padovani focus on LTSs of convergent terms; respectively the LTS of co-inductively generated regular trees

$$\langle\, \mathsf{CCS}^{\mathsf{coind},\Downarrow}_{\mathrm{w}\tau}\, \mathsf{Act}_{\checkmark},\, \mapsto \rangle$$

and the LTS of recursive terms of $\mathsf{CCS}_{\mathrm{w}\tau}$,

$$\langle\, \mathsf{CCS}^{\mathsf{rec,fb},\Downarrow}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$$

At present two styles are used to define the (strong) compliance, Padovani uses computations (see [Padovani, 2010, Definition 2.1]), whereas [Castagna et al., 2009] prefers a co-inductive definition (see Definition 2.4 there).

The only comparison between the various compliances appeared in the literature is [Bugliesi et al., 2010].

Chapter 4

Testing theory was introduced by [De Nicola and Hennessy, 1984], and the standard reference on the topic is the "green book" [Hennessy, 1985].

In [De Nicola and Hennessy, 1984] a semantic theory for CCS is developed, and testing pre-orders are used to state when two processes are equivalent with respect to a set of tests. That paper contains the axiomatisation of testing pre-orders on finite terms (see Theorem 4.3.8 there), and define a denotational model of these pre-orders, which is based on representation trees (Theorem 5.2.10 there). The axiomatisation of the MUST pre-order sheds light also on other equivalences; one example is the failure equivalence of [Brookes et al., 1984]. This equivalence provides a denotational model for the language CSP, and its axiomatisation has been discussed by De Nicola first in 1983, and then in an extended paper appeared in 1985. An account of testing theory, failure equivalence, and other equivalences for concurrent languages can be found in [Sangiorgi, 2012, Chapter 5].

In [Cleaveland and Hennessy, 1993] a connection is established between the MUST testing equivalence $\eqsim_{\mathrm{MUST}}$ and a certain kind of bisimulations, the Π-bisimulations; this connection is exploited to show a decision procedure for testing on finite-state processes.

Recently the logical characterisation of the MUST pre-order has been shown by [Cerone and Hen-

nessy, 2010]. In that paper the authors isolate the fragment of recursive Hennessy-Milner logic which expresses exactly the properties that can distinguish two processes with respect to the equivalence $\eqsim_{\text{MUST}}$: $p \not\eqsim_{\text{MUST}} q$ if and only if there exists a formula ϕ such that p satisfies ϕ while q does not satisfy it.

Chapter 5

The overall aim of the research on the compliance theory is practical, and largely motivated by the adoption of web-services.

[Carpineti et al., 2006] is the first paper that shows, by means of examples, that it is possible to encode fragments of WSDL and WSCL in the proposed language of contracts.

In [Laneve and Padovani, 2007] a *subcontract* relation is defined as we define our server pre-orders, but using the behavioural compliance $\dashv^{\mathsf{bhv}}$ and requiring also a condition on the interfaces of the constrained contracts. The authors focus on a set of well-behaved contracts and show that they have duals, in the sense that if $\text{I}[\rho]$ is well-behaved then there exists a $\text{I}[\sigma]$ such that $\text{I}[\rho] \dashv^{\mathsf{bhv}} \text{I}[\sigma]$ (see Theorem 3 of that paper).

In view of the limitations of the subcontract relation, for instance its lack of "width extension", Castagna et al. and Padovani have introduced auxiliary tools in the theory: filters and orchestrators.

[Castagna et al., 2009] follows the research line opened in [Carpineti et al., 2006], extending the results. The *strong subcontract*, $\sqsubseteq$, (see Definition 2.7 there) is the server refinement of the theory, and an alternative characterisation is proven in Theorem 2.9. To overcome the limitations of the strong subcontract, a weak subcontract is defined, $\preceq$. The import of the weak relation is that if $\sigma_1 \preceq \sigma_2$ then it is possible to "filter" the behaviour of σ_2, so that $\sigma_2 \sqsubseteq f(\sigma_2)$. Filters are essentially coercions on the behaviour of contracts, and are meant to hide the interactions that would disrupt the correctness of the overall system. An effective deduction system for $\preceq$ is provided (see Proposition 3.25). The authors show also how to encode WS-BPEL activities [OASIS, 2007] into contracts, and discuss the implementation details of filters.

Padovani's approach is similar to the one of [Castagna et al., 2009]. Also in [Padovani, 2010] a weak refinement for servers is introduced, $\preceq$, such that if $f : \sigma_1 \preceq \sigma_2$ then $\sigma_1 \sqsubseteq f(\sigma_2)$. The filters devised by Padovani, and referred to as *orchestrators*, are more sophisticated than the ones of [Castagna et al., 2009], in that they acts as buffers and mediate the (possibly asynchronous) actions of clients and servers.

Fair theories Other approaches have been taken to investigate the notions of compliance; for instance [Bravetti and Zavattaro, 2009] are theories inspired by the fair testing of [Rensink and Vogler, 2007]. In this case the theories pertain only refinements for peers, and do not deal with the refinements for servers and clients. Filters have been adapted to the fair framework in [Bernardi et al., 2008].

Chapter 6

To the best of our knowledge, the only papers in the literature that discuss models of the sub-typing on first-order session types are [Barbanera and de'Liguoro, 2010; Laneve and Padovani, 2008] and our [Bernardi and Hennessy, 2012].

A detailed discussion of first two papers is in Section 6.6. Our paper contains the theory of compliance for session contracts that we have described in Section 6.5 and Section 6.5. In the paper the model of $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$ is defined directly in terms of the compliance pre-orders, as we had not yet investigated the must pre-orders on first-order session contracts.

Chapter 7

Sessions and session types have gained much consensus, and fostered so much research as to nearly overwhelm the novice.

First, we focus our discussion on the feature of session types that pertains our work most, namely the sub-typing relation; then we comment on other works and give an exhaustive series of pointers to the relevant literature.

[Dezani-Ciancaglini and de'Liguoro, 2009] overviews the state of the art, and the reader interested in technical details may find them in that paper.

The advent of computer networks, and the Internet in particular, has called for the development of means to help programmers check patterns of communication that software perform.

The system of *types for interactions* put forth by [Honda, 1993] has laid the ground for the subsequent introduction of session types, which are a restricted version of the interaction types; for instance the types $\downarrow \texttt{nat}; \texttt{1} \,\&\, \uparrow \texttt{nat}; \texttt{1}$ is an interaction type which is not a session type, as it is a branch type that contains an output ($\uparrow \texttt{nat}$) and an input ($\downarrow \texttt{nat}$).

The idea of *session* as pattern of information flowing between *two* programs, via a connection private to them, has been proposed for the first time by [Takeuchi et al., 1994]. There, a concurrent language with constructs to organise sessions is presented, along with a type discipline based on session types (see Definition 5.1). Intuitively, session types are meant to capture precisely the information flow that takes place via a given connection. [Honda et al., 1998] elaborates further on these ideas, comparing the need for structured computing with the urging need of primitives for structured communications. The paper by Honda et al. contains various examples which illustrate the advantages of sessions and session types; that paper is also the first account of recursive session types.

By and large, the paper [Pierce and Sangiorgi, 1996] on (sub)typing for mobile processes has laid the ground for the theory of sub-typing on session types; this theory has been developed by Gay and Hole.

The first account of sub-typing for session types was given in 1999. The proposed programming language is a dialect of the pi-calculus; the sub-typing, $\leq$, acts on non-session types as proposed by Pierce and Sangiorgi, whereas on session types acts as we described in Section 2.1.1.

The sub-typing is given by inference rules to be interpreted in an algorithmic fashion, as the algorithmic subsort relation of Pierce and Sangiorgi. To the best of our knowledge, [Gay and Hole, 1999] present the fist specification of a standard protocol, the `pop3`, by means of session types.

The next development of the sub-typing is in [Gay and Hole, 2005]. The new relation, denoted $\leq_c$, is defined co-inductively (see Definition 4 there), and is essentially the one we have presented in Definition 7.1.5. The differences between Definition 7.1.5 and the original definition of Gay and Hole are two,

- the co-inductive sub-typing of Definition 7.1.5 allows only one type in the input/output fields of types, whereas the original definition allows n types
- the original definition treats also standard channel types (i.e. non-session types).

In [Gay and Hole, 2005] the algorithmic sub-typing is still present, and is used to prove that the relation $\leq_c$ is decidable (see Theorem 4 in that paper). In turn this result is used to define a type checking algorithm for the proposed programming language.

To add more flexibility to the sub-typing, in [Gay, 2008] it shown a version of finite session types with bounded polymorphism, along with a sub-typing relation defined inductively on these terms. The sub-typing is proven decidable (Theorem 3 there), and a type checking algorithm is given.

The most recent account of sub-typing on (session) types is [Gay and Vasconcelos, 2010]. There the object language is multi-threaded, functional, and the communication, which takes places on buffered channels, is asynchronous. Session types are used to (a) ensure that threads do not become blocked,

and (b) prove statically the bounds on the size of the buffers required by the communication channels; this result depends on the sub-typing. The authors define the sub-typing by mixing the sub-typing à la Gay and Hole on session types with the sub-typing on function types and pairs as per [Pierce, 2002]; there is also an additional rule for linear function,

$$T \to U <: T \multimap U \tag{10.1}$$

The rule above ensures that a function that can be used exactly once can be replaced by a function that can be used an unlimited amount of times. The definition of $<:$ (Definition 1) adheres to the style of [Vasconcelos, 2009a], and shows the set-theoretical construction of the the operator F whose greatest fixed point is the sub-typing. We followed a similar style, but we defined our functional $\mathcal{F}^{\preccurlyeq_{\mathsf{sbt}}}$ by means of inference rules.

The various versions of the sub-typing à la Gay and Hole are defined in syntactical terms.

Another approach has been taken in [Giachino, 2009, Chapter 3], and the paper [Castagna et al., 2009]. Giachino puts forth a theory of types and *session descriptors* for a dialect of the pi-calculus named PiST. The main result is that the well-typed compositions of processes that (a) are closed and (b) contain no private channels, enjoy the progress property (Theorem 3.2.22 in the PhD thesis of Giachino).

As for syntax, session descriptors are strikingly similar to $\mathsf{CCS}_{\mathrm{w}\tau}$, but types are generated co-inductively in the style of [Castagna et al., 2009]. Moreover, the operational semantics of session descriptors differ from the one of $\mathsf{CCS}_{\mathrm{w}\tau}$. The sub-typing on session descriptors is defined *semantically*; the construction relies on the techniques of Castagna and Frisch [2005], and starts from a sub-typing on base reminiscent of our $\preccurlyeq_{\mathsf{b}}$.

The sub-typing is but one of the many areas in which session types have been applied. The oncoming paragraphs are roughly organised by "theme"; in each paragraph we give the pointers to the relevant literature on session types with respect to the theme.

Programming paradigms The research community has invested much effort in adapting session types to suit different kinds of programming languages; for example,

- process calculi (pi-calculus, CaSPiS) [Dezani-Ciancaglini and de'Liguoro, 2009; Gay and Hole, 1999, 2005; Mezzina, 2008]
- ambient calculi [Garralda et al., 2006]
- functional languages [Vasconcelos, 2009b; Vasconcelos et al., 2006] and the most recent [Gay and Vasconcelos, 2010]
- object oriented languages [Dezani-Ciancaglini et al., 2009; Gay et al., 2012; Giachino, 2009]

Implementations Implementations of session types also exist,

- in Haskell [Imai et al., 2010; Neubauer and Thiemann, 2004; Pucella and Tov, 2008; Sackman and Eisenbach, 2008],
- in Sing#, which is a variant of C#, by [Fähndrich et al., 2006]
- in Java [Gay et al., 2012, 2010; Hu et al., 2008]
- [Honda et al., 2012] suggests to use the multiparty asynchronous session types presented in [Honda et al., 2008] to verify MPI programs.

Logic Linear logic [Girard, 1987] is a useful tool in reasoning on resource ownership, replication, and phenomenon as interference. Connection between linear logic and session types have been shown in [Caires and Pfenning, 2010], and simplified in 2012 by Wadler.

He defines two calculi, respectively called CP and GV. CP is a version of the pi-calculus tailored to be typed by propositions of classical linear logic, while GV is a slightly amended version of the language of [Gay and Vasconcelos, 2010]. Wadler defines an interpretation of the calculus GV into the calculus CF (see Figure 6 and 7 in his paper), that preserves types and shows a relation reminiscent of the Curry-Howard isomorphism, propositions are (session) types, proofs are processes, and *communication* is the cut elimination.

Deadlock freedom The standard type systems based on session types, as the one used by [Gay and Hole, 2005], are not strong enough to prove that well-typed processes progress. The archetypal example is the following composition,

$$x!(3).y!(\texttt{true}).\mathbf{0} \mid\mid y?(z\colon \texttt{Bool}).x?(v\colon \texttt{Int}).\mathbf{0}$$

The composition above is well types, as the two processes run in parallel use the channel x and y in complementary ways, and the type safety indeed works: if a communication takes places, then there is no mismatch-between the type of the sent data and the data expected by the input operations. Nevertheless, the processes above use x and y in opposite order, so no interaction can take place; the composition is stuck.

This issue has been tackled by [Giachino, 2009] and by [Dezani-Ciancaglini et al., 2007]. The result obtained by Dezani-Ciancaglini et al. is similar to the progress theorem of Giachino, which we have already described.

Higher-order languages In [Mostrous and Yoshida, 2007] two versions of session types for HOπ-calculus have been presented. The first type system, which we refer to as "simple", relies on a combination of session types and types of the simply typed λ-calculus, with the additional rule in Eq. (10.1). The type safety result (Theorem 3.4) ensures that well-typed processes with balanced sessions do not reduce into errors. The second type system put forth by Mostrous and Yoshida is based on fine grained session types, in turn inspired to the works [Yoshida, 2004; Yoshida and Hennessy, 2002], and [Hennessy et al., 2005]. The fine-grained type system allows to type more processes than the simple type system, and still guarantees that well typed programs do not reduce to errors (Theorem 4.2).

Multi-party session types Most of the research on sessions and their types pertains to *binary* sessions, that is communication patterns realised by parties that communicate via *two* endpoints of a connection. The consequence is that binary sessions are not expressive enough to model interactions that do not follow a strict peer to peer logic. For instance, multicasting allows messages to be sent to a finite number of parties; also, more than two parties may (in some sense) combine their behaviours to reach a common end.

These facts have called for an extension of session types expressive enough to specify and statically check the behaviour of n-parties; that is multi-party composition of processes.

The *global types* of [Honda et al., 2008] are such an extension. Roughly speaking, a global type describes the overall communications that n-parties should adhere to. The local type of each participant is obtain as a projection of the global type, and the code of the participant is checked against the local type. The type system of Honda et al. is powerful enough to ensure communication safety and progress (Theorem 5.5 and 5.12 there).

A sub-typing for multi-party session types has been studied by [Padovani, 2011]. The *fair sub-typing* defined there follows a testing approach inspired to the fair testing, and it turns out to differ from the sub-typing à la Gay and Hole. The fair testing (a) does not allow refinements such as $a \oplus b \leq a$, for they may hinder the possibility of reaching success; and (b) allows the refinement $a + b \leq a$ if b is not a usable action. This is similar to our restricted MUST client pre-order (see point (iii) of Lemma 6.2.9).

Chapter 8

To the best of our knowledge, there exist two encodings of session types and the sub-typing à la Gay and Hole into other domains.

[Gay et al., 2008] show an encoding of session types in the generic type system (GTS) of [Igarashi and Kobayashi, 2004]. The correspondences obtained are on process reduction, typing derivations (see Theorem 1 and 3), and type safety. By and large, the outcome of the study of Gay et al. is that since reasoning techniques for session types are *conceptually fairly straightforward*, the effort required to use the GTS in practice seems not to pay off.

The second interpretation has been put forth by [Dardha et al., 2012]. The authors are concerned with the expressive power of session types. The result of their investigation is that session types and their features can be recovered by using the standard types of pi-calculus [Sangiorgi and Walker, 2001, see Part III]. Moreover, results such as preservation and type safety for session types become consequences of the same results in the standard theory of types.

Chapter 9

The pi-calculus is the evolution of the CCS that can express the notion of mobility, and was presented first in [Milner et al., 1992]. The standard books on the pi-calculus are [Milner, 1999], and [Sangiorgi and Walker, 2001]; while Milner's work is an introductory text, the book by Sangiorgi and Walker can be considered as the encyclopedia of the pi-calculus.

The pi-calculus has two remarkable features. First, many notions such as labels, channels, variables, pointers, and so forth are replaced by *names*. Processes operates on names, and mobility is modelled by name passing. This leads to a great simplification in the formalism, while retaining the expressive power of usual programming languages. The second feature, shown by [Sangiorgi, 1993], is that even though formally only names can be passed around by processes, this is enough to represent also the mobility of processes (i.e. programs) themselves.

Chapter 11

Conclusion

Truly there is no such thing as finality.

— Bram Stocker, *Dracula*

11.1 Summary

In this thesis we have taken a foundational approach, and proposed two ways to formalise the correctness of a software system, namely the MUST testing and the compliance relation. This has allowed us to propose a series of pre-orders that let us understand and answer rigorously questions such as

Q) if $r \parallel p$ is a correct system, what relation between p and q guarantees that $r \parallel q$ is a correct system as well?

In particular, by proving the behavioural characterisations of the pre-orders generated by MUST and $\dashv$, we have laid bare a series of principles that let us replace a given piece of software, with another piece of software, without hampering the overall correctness of the system.

We started our investigation in Chapter 4, studying an extension of the well-known testing theory on processes and exhibiting the characterisation of the MUST pre-orders for servers, clients, and peers.

In Chapter 5 we moved to the compliance theory, and the compliance pre-orders for servers, clients and peers. The results of Chapter 4 proved to be useful touchstones to characterise the compliance pre-orders. Indeed, common ideas lay at the heart of the behavioural characterisation of the compliance and the MUST pre-orders, for instance traces, convergence, usability, and acceptance sets, to name a few.

The MUST testing and the compliance relation, when used in LTSs as general as the one of processes, generate pre-orders that do not allow width extension: $a \not\sqsubseteq a + b$.

Because of this reason in Chapter 6 we focused our attention on the LTS of session contracts, and we have studied the MUST pre-orders and the compliance pre-orders that emerge in the restricted LTS of session contracts. It turned out that the pre-orders of both families, when combined, provide a fully abstract model of the sub-typing on first-order session types.

Motivated by this result, in the second part of the thesis we have enriched the language of session contracts with higher-order terms, thereby providing a fully abstract model of the well-known sub-typing à la Gay and Hole on session types.

In this thesis we began our exposition from processes and MUST, and we concluded it discussing session types and sub-typing, passing through the theory of compliance. Step by step, we have motivated the oncoming ideas, and we have tried to show the connections between the theories, so as to unravel them in an organic and harmonious way. The reader will decide the extent of our success.

11.2 Open questions

Death: Don't you ever stop asking?
Antonius Block: No. I never stop.
— Ingmar Bergman, *The Seventh Seal*

This section contains a series of problems which arise from our results. The order in which we present the questions follows the structure of the chapters.

Q1) A relation on basic parallel processes (BPP) called type compliance and denoted $\propto$ is presented by [Acciai and Boreale, 2008]. BPP is not a subset of our language for processes, because BPP has replication, whereas $\mathsf{CCS}_{\mathrm{w}\tau}$ has not. Does an encoding of BPP into $\mathsf{CCS}_{\mathrm{w}\tau}$ which preserves the behaviour of terms (possibly up-to weak bisimulation) exist? If such an encoding exists then it is conceivable that our compliance relation, $\dashv$, coincides (possibly up-to weak bisimulation) with $\propto$. This would prove that the relation $\sqsubseteq_{\mathrm{SVR}}$ is the sub-typing, thereby answering a question posed in the last section of [Acciai and Boreale, 2008].

Q2) In Section 4.2 we introduced the pre-order for clients $\sqsubseteq_{\mathrm{CLT}}$. The obvious question to be answered about $\sqsubseteq_{\mathrm{CLT}}$ is whether that relation is decidable or not. Given the non-trivial role played by the usability in the characterisation of $\sqsubseteq_{\mathrm{CLT}}$, a problem related to the decidability of $\sqsubseteq_{\mathrm{CLT}}$ is the decidability of $\mathcal{U}^{\mathrm{MUST}}_{\mathrm{CLT}}$.

Q3) The MUST peer pre-order $\sqsubseteq_{\mathrm{P2P}}$ that we defined in Section 4.3 is given by the relation $\mathrm{MUST}^{\mathrm{P2P}}$. The relation $\mathrm{MUST}^{\mathrm{P2P}}$ does not require the peers to report success at the same time, so one may introduce a synchronous version of $\mathrm{MUST}^{\mathrm{P2P}}$, say $\mathrm{MUST}^{\mathsf{snc}}$, that require this. That is r $\mathrm{MUST}^{\mathsf{snc}}$ p if and only if all the maximal computations of $r \parallel p$ contain a state $r' \parallel p'$ in which $r' \xrightarrow{\checkmark}$ and $p' \xrightarrow{\checkmark}$.

In the obvious way we obtain the *synchronous* MUST peer pre-order $\sqsubseteq_{\mathsf{snc}}$. Plainly $\sqsubseteq_{\mathrm{P2P}} \not\subseteq \sqsubseteq_{\mathsf{snc}}$, for instance $\alpha.1 \sqsubseteq_{\mathrm{P2P}} 1 + \alpha.0$, whereas $\alpha.1 \not\sqsubseteq_{\mathsf{snc}} 1 + \alpha.0$; the peer $\overline{\alpha}.1$ lets us prove the last inequality.

What is the behavioural characterisation of the the synchronous MUST peer pre-order?

Q4) A second pre-order for clients that we introduced is $\sqsubseteq_{\mathrm{CLT}}$ (see Section 5.2). As for $\sqsubseteq_{\mathrm{CLT}}$, the open question regards the decidability of $\sqsubseteq_{\mathrm{CLT}}$. In turn this leads to the question is $\mathcal{U}^{\dashv}_{\mathrm{CLT}}$ decidable? A starting point to answer the question may be Section 4 of [Padovani, 2010], in which a co-inductive characterisation of viable contracts is proven.

Q5) Some peer pre-orders studied in the literature, see for instance [Bravetti and Zavattaro, 2009; Padovani, 2011], are formulated in terms of multi-party systems; that is systems with any finite number of distinguished participants,

$$p_1 \parallel p_2 \parallel p_3 \parallel \ldots \parallel p_n$$

Our pre-orders for peers (see Definition 4.3.1 and Definition 5.3.1), on the contrary, take into the account only two participants. Let us denote with $\sqsubseteq^n_{\mathrm{P2P}}$ the pre-order defined in terms of n participants being satisfied, with $n > 2$. It is easy to prove that if $p \sqsubseteq^n_{\mathrm{P2P}} q$ then $p \sqsubseteq_{\mathrm{P2P}} q$. The open question is the converse, is it true that if $p \sqsubseteq_{\mathrm{P2P}} q$ then $p \sqsubseteq^n_{\mathrm{P2P}} q$ for every $n > 2$? If it is not true, what is, then the characterisation of $\sqsubseteq^n_{\mathrm{P2P}}$?

The same question can be asked also about $\sqsubseteq_{\mathrm{P2P}}$.

Q6) In Chapter 8 we have seen that there exists a straightforward way to map session types into contracts, namely the function $\mathcal{M}$, which preserves the sub-typing relation on types; the image

of $\preccurlyeq_{\mathsf{sbt}}$ through $\mathcal{M}$ being the pre-order $\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$. We have not addressed the converse question; does a mapping from contracts to session types exist, which maps any one of the behavioural pre-orders we studied, into the sub-typing relation?

Q7) We have seen that in general the MUST server pre-order and the compliance server pre-order are not comparable (see Eq. (5.3)). Then we have shown that in the LTSs $\langle\, \mathsf{CCS}^{\Downarrow}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$, $\langle\, \mathsf{CCS}^{\mathsf{fb},\Downarrow}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$, and $\langle\, \mathsf{SC}_{\mathrm{fo}},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$ the server pre-orders are comparable, and in the second and third LTS they coincide (Theorem 5.1.20, Corollary 6.4.8),

$$\sqsubseteq^{\Downarrow,\mathsf{fb}}_{\approx\text{SVR}} = \sqsubseteq^{\Downarrow,\mathsf{fb}}_{\text{SVR}}, \qquad \sqsubseteq^{\mathsf{fo}}_{\approx\text{SVR}} = \sqsubseteq^{\mathsf{fo}}_{\text{SVR}}$$

A question that we have not addressed is whether the conditions of being convergent and finite branching are sufficient for the server pre-orders to coincide.

More formally, if S is some set of states, is it true that if the LTS $\langle\, S,\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$ is convergent and finite-branching then $\sqsubseteq^{S}_{\approx\text{SVR}} = \sqsubseteq^{S}_{\text{SVR}}$?

Q8) We have seen that $\sqsubseteq_{\approx\text{CLT}} \neq \sqsubseteq_{\text{CLT}}$, and that $\sqsubseteq^{\mathsf{fo}}_{\approx\text{CLT}} \neq \sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$ (Eq. (5.5), Example 6.2.8).

We conjecture that in the LTS of finite (i.e. non recursive) session contacts, $\langle\, \mathsf{SC}^{fin},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$, the equality $\sqsubseteq^{\mathsf{SC}^{fin}}_{\text{CLT}} = \sqsubseteq^{\mathsf{SC}^{fin}}_{\approx\text{CLT}}$ holds true.

What conditions on the LTS are necessary to prove the equality between the client pre-orders? Is there a non-trivial LTS in which the refinements for clients coincide?

Q9) In [Cerone and Hennessy, 2010] a logical characterisation of the standard MUST pre-order has been proven. This lets us prove that $p \not\approx_{\text{MUST}} q$ if there is some formula ϕ that is satisfied by p and not by q.

What is the logical characterisation of the MUST client pre-order? What are the logical characterisations of the compliance pre-orders?

Q10) In the opening of Chapter 6 we pointed out that the pre-orders on the general LTS

$$\langle\, \mathsf{CCS}_{\mathrm{w}\tau},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$$

do not allow width extension, that is refinements such as the following $\alpha \preceq \alpha + \beta$. In the LTS of session contract $\langle\, \mathsf{SC}_{\mathrm{fo}},\, \mathsf{Act}_{\tau\checkmark},\, \longrightarrow \rangle$, on the contrary, the server and the client pre-orders allow width extension.

Which restriction on an LTS are necessary so that the MUST and the compliance pre-orders allow width extension?

Q11) In Section 6.2 we put forth a characterisation of the pre-order $\sqsubseteq^{\mathsf{fo}}_{\approx\text{CLT}}$ which is syntax-oriented (Proposition 6.2.12). The characterisation, though, is not completely syntactical, because it involves the set of usable clients $\mathcal{U}^{\mathsf{SC}_{\mathrm{fo}}}_{\text{CLT}}$, which we have not characterised syntactically.

What is the syntactic characterisation of the set $\mathcal{U}^{\mathsf{SC}_{\mathrm{fo}}}_{\text{CLT}}$?

Q12) In Section 6.3 we have shown a fully abstract model of the relation $\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}$. Can the interpretation of session types into session contracts, and the language of session contracts be adapted so as to provide a model also of the sub-typing on standard channel types?

Q13) The dependent compliance relation lets us exhibit a model of the sub-typing $\preccurlyeq_{\mathsf{sbt}}$ on higher-order session types (see Theorem 8.4.9).

Does the dependent MUST testing provide the same model of $\preccurlyeq_{\mathsf{sbt}}$ given by the dependent compliance?

Q14) What is the relation between the fair sub-typing of [Padovani, 2011] and the MUST pre-orders on session contracts?

Q15) To what degree can the model exhibited in Theorem 8.4.9 be extended to the polymorphic types of [Gay, 2008]?

Q16) What are the implication of the interpretation of $\preccurlyeq_{\mathsf{sbt}}$ given in [Dardha et al., 2012]? To What degree $\mathsf{CCS}_{\mathrm{w}\tau}$can be used as a model for the sub-typing for the types of the pi-calculus?

Q17) The results of Chapter 8 rely heavily on the fixed points of particular functionals. For instance the model of $\preccurlyeq_{\mathsf{sbt}}$ is provided by $\nu X.\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}(X) \cap \mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{SVR}}}(X)$. Have the functionals $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}$ and $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{SVR}}}$ a unique fixed point?

Q18) Can we obtain the session contract pre-order by applying the functionals $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}$ and $\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{SVR}}}$ one after the other? Formally, is the following equality true?

$$\nu X.\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{CLT}}}(\mathcal{F}^{\sqsubseteq^{\mathsf{ho}}_{=\mathrm{SVR}}}(X)) = \sqsubseteq^{\mathsf{ho}}_{=\mathrm{P2P}}$$

Appendix A

A complete lattice of pre-orders on higher-order session contracts

Many of the relations used throughout the thesis are defined as fixed point of functions (or rule functionals). To prove the existence of these fixed points we rely on the Knaster-Tarski theorem, which we can apply only to endofunctions on complete lattices. This raises the problem of finding a suitable complete lattice each time we need one. In particular, the functions $\mathcal{F}_{\text{CLT}}$ (Lemma C.0.34) and $\mathcal{F}_{\text{SRV}}$ (Lemma 8.2.8) are monotone, but their domain is not closed with respect to set inclusion, so we have to find another operation to obtain the least upper bound of $\textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$.

Example A.0.1. In this example we show that if we order pre-orders by using the set inclusion $\subseteq$, the the operation $\cup$ on pre-orders does not give the LUB of its two arguments. This is a consequence of a more general fact that pertains transitive relations.

Let $\mathcal{T}$ be the set of transitive relations on session contracts. A natural way to order the elements of $\mathcal{T}$ is the set inclusion. The operations that give the LUB, and the GLB *on sets* are $\cup$ and $\cap$.

The structure $\langle \mathcal{T}, \cup \rangle$ is *not* a magma. See also Figure A.1. Let

$$T_1 = \{(?\mathtt{1}_1.\,\mathtt{1},\, \mathtt{0})\}$$
$$T_2 = \{(\mathtt{0},\, !\mathtt{1}.\,\mathtt{1})\}$$

Both relations are trivially transitive, thus they are in $\mathcal{T}$. Consider the naive candidate as LUB of T_1 and T_2, that is $T_1 \cup T_2$. We know by construction that

$$(\mathtt{1},\, \mathtt{0}) \in \mathcal{T}_1 \cup \mathcal{T}_2, \quad (\mathtt{0},\, !\mathtt{1}.\,\mathtt{1}) \in \mathcal{T}_1 \cup \mathcal{T}_2$$

Note, though, that $(?\mathtt{1}_1.\,\mathtt{1},\, !\mathtt{1}.\,\mathtt{1}) \notin \mathcal{T}_1 \cup \mathcal{T}_2$, so $T_1 \cup T_2$ is not an element of $\mathcal{T}$. This means that if we order *transitive* relations according to $\subseteq$, the relation $T_1 \cup T_2$ can not be a LUB of T_1 and T_2. □

The example above shows that the problem lies in the naive use of the operation $\cup$ to (try to) construct the LUBs of objects. To provide a (complete) lattice on transitive relations, we have to use operations that preserve the structure of "being transitive", and the operation $\cup$ does not.

The right operations are the transitive closure of the union of sets, and the intersection on sets; we denote them as follows,

$$\bigsqcup X = [\bigcup\{\, \mathcal{O} \mid \mathcal{O} \in X \,\}]^+$$
$$\bigsqcap X = \bigcap\{\, \mathcal{O} \mid \mathcal{O} \in X \,\}$$

Hereafter we will use the infix notation when using $\sqcup$ and $\sqcap$ as binary operations.

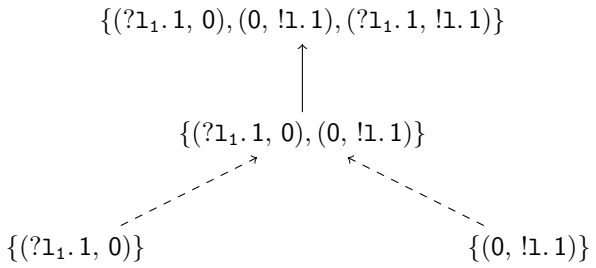


Figure A.1: The result of the operations $\cup$ (dashed arrows) and $\sqcup$ (dashed and solid arrows).

Example A.0.2. Consider again the relations T_1 and T_2 of Example A.0.1. The definition of $\sqcup$ implies that

$$T_1 \sqcup T_2 = \{(?\mathtt{1}_1.\,1,\, 0), (0,\, !\mathtt{1}.\,1), (?\mathtt{1}_1.\,1,\, !\mathtt{1}.\,1)\}$$

The relation $T_1 \sqcup T_2$ is the smallest transitive relation that contains T_1 and T_2. □

To prove the result of this appendix, we need an auxiliary lemma.

Lemma A.0.3. Let $Y \subseteq \textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$, and $\mathcal{O} \in Y$. If for every $\mathcal{P} \in Y$, $\mathcal{P} \subseteq \mathcal{O}$, then $\bigsqcup Y \subseteq \mathcal{O}$.

Proof. Let $\mathcal{B} = \bigcup\{\, \mathcal{P} \mid \mathcal{P} \in Y \,\}$. Fix a pair $(\sigma_1, \sigma_3) \in \bigsqcup Y$. Note that by definition of $\bigsqcup$, $(\sigma_1, \sigma_3) \in \bigsqcup Y$ if and only if $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$.

We prove that if $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$ then $\sigma_1 \mathcal{O} \sigma_3$; the argument is by induction on the derivation of $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$.

Base case In this case the derivation that $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$ is due to the axiom in Figure 7.10,

$$\frac{}{\sigma_1\ [\mathcal{B}]^+\ \sigma_3}\ (\sigma_1, \sigma_3) \in \mathcal{B};\ [\textsc{trc-a}]$$

The side conditions and the definition of $\mathcal{B}$ ensure that there exists a $\mathcal{P} \in Y$ such that $\sigma_1 \mathcal{P} \sigma_3$. The hypothesis that for every $\mathcal{P} \in Y$ the inclusion $\mathcal{P} \subseteq \mathcal{O}$ holds true, implies that $\sigma_1 \mathcal{O} \sigma_3$.

Inductive case In the inductive case the last rule in the derivation of $\sigma_1\ [\mathcal{B}]^+\ \sigma_3$ has to be [trc-r], and the derivation has the following form,

$$\frac{\dfrac{\vdots}{\sigma_1\ [\mathcal{B}]^+\ \sigma_2} \qquad \dfrac{\vdots}{\sigma_2\ [\mathcal{B}]^+\ \sigma_3}}{\sigma_1\ [\mathcal{B}]^+\ \sigma_3}\ [\textsc{trc-r}]$$

The inductive hypothesis ensures that $\sigma_1 \mathcal{O} \sigma_2$ and $\sigma_2 \mathcal{O} \sigma_3$. The transitivity of $\mathcal{O}$ implies that $\sigma_1 \mathcal{O} \sigma_3$. □

Lemma A.0.4. The pre-order $\langle\, \textsc{Pre}(\mathsf{SC}^2_{\text{HO}}), \sqsubseteq \,\rangle$ is a complete lattice.

Proof. We have to prove that every subset of $\textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$ has least upper bound and greatest lower bound. Let X be a subset of $\textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$, that is a set of pre-orders on higher-order session contracts.

We show that $\bigsqcap X$ is the greatest lower bound of X. We have to prove two things:

1) for every $\mathcal{O} \in X$, $\bigsqcap X \subseteq \mathcal{O}$

2) if $\mathcal{O}$ is a lower bound of X, then $\mathcal{O} \subseteq \bigsqcap X$

Let $\mathcal{O} \in X$; to prove 1) we have to show that $\bigcap X \subseteq \mathcal{O}$. This is obvious in view of the definition of set intersection. To prove 2) above let $\mathcal{O}$ be a lower bound of the set X; we have to show that $\mathcal{O} \subseteq \bigcap X$. As $\mathcal{O}$ is a lower bound, $\mathcal{O} \subseteq \mathcal{P}$ for every $\mathcal{P} \in X$. This and the definition of set intersection imply that $\mathcal{O} \subseteq \bigcap X$.

Now we show that $\bigsqcup X$ is the least upper bound of X:

3) for every $\mathcal{O} \in X$, $\mathcal{O} \subseteq \bigsqcup X$

3) if $\mathcal{O}$ is an upper bound of X, then $\bigsqcup X \subseteq \mathcal{O}$

Let $\mathcal{O} \in X$; 3) above follows from the following set inclusions

$$\mathcal{O} \subseteq \bigcup\{\, \mathcal{P} \mid \mathcal{P} \in X \,\} \subseteq [\, \bigcup\{\, \mathcal{P} \mid \mathcal{P} \in X \,\} \,]^{+} = \bigsqcup X$$

To prove 4), let $\mathcal{O} \in X$ and assume that $\mathcal{P} \subseteq \mathcal{O}$ for every $\mathcal{P} \in X$. Lemma A.0.3 implies that $\bigsqcup X \subseteq \mathcal{O}$. □

Appendix B

Necessary and sufficient conditions

In Section 8.1 we have proven that if we restrict our attention to the pre-orders on higher-order session contracts, then we can provide a characterisation of the dependent client pre-orders, and also show that the function $\lambda X.\, \sqsubseteq_{\text{CLT}}^{X}$ is monotone. We have proven that the fixed point $\nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X)$ is a transitive relation (Corollary 8.1.14).

In this appendix we explain why it is *necessary* to use pre-orders to achieve the results of Section 8.1.

The first example pertains the existence of dual session contracts; in the higher-order setting it is not an obvious fact.

Example B.0.5. [Strong totality is necessary]
In Lemma 7.2.17 we have proven that if a relation $\mathcal{B}$ is strongly total, then $\textsc{Dual}(\mathcal{B})$ is total.

In this example we show that if a relation $\mathcal{B}$ is not strongly totalthen $\textsc{Dual}(\mathcal{B})$ is not a total $\mathcal{B}$-compliance relation. This is true because if $\mathcal{B}$ is not strongly total, then there exists a session contract ρ that does not comply with any other session contract. Let us see why.

Suppose that $\mathcal{B}$ be not total. It follows that there exists some $\hat{\rho}$ such that for every $\hat{\sigma}$, (a) $\hat{\rho} \not\mathcal{B}\ \hat{\sigma}$, or (b) $\hat{\sigma} \not\mathcal{B}\ \hat{\rho}$.

Without loss of generality we can assume (a), and let $\rho = !(\hat{\rho}).\,\mathtt{1}$. Thanks to (a), we can prove that for every σ, $\rho \parallel \sigma \overset{\tau}{\not\longrightarrow}$; we can prove also that $\rho \overset{\checkmark}{\not\longrightarrow}$, thus Definition 7.2.4 ensures that $\rho \not\dashv_{\mathcal{B}} \sigma$. □

The next examples show that the relation $\mathcal{B}$ has to be a pre-order if we want our syntactical characterisation of $\sqsubseteq_{\text{CLT}}^{\mathcal{B}}$ to be true.

Example B.0.6. [Reflexivity is necessary]
Let $\mathcal{B}$ be a binary relation on session contracts, that is not reflexive. In this example we show that Lemma 8.1.5 is not true for the pre-order $\sqsubseteq_{\text{CLT}}^{\mathcal{B}}$. In other terms, we prove that

$$\sqsubseteq_{\text{CLT}}^{\mathcal{B}} \not\subseteq \preceq_{\text{CLT}}^{\mathcal{B}}$$

We have to exhibit a pair in $\sqsubseteq_{\text{CLT}}^{\mathcal{B}}$ which does not satisfy any one of the points (i) ... (**??**) of Lemma 8.1.5.

There exists a session contract $\hat{\rho}$ such that $\hat{\rho} \not\mathcal{B}\ \hat{\rho}$. The argument depends on $\mathcal{B}$ being strongly total or not. We discuss first the latter case.

If $\mathcal{B}$ is not strongly total, then Example B.0.5 implies that there exists a $\rho = !(\hat{\rho}).\,\mathtt{1}$ such that

$$\{\, \sigma \mid \rho \dashv_{\mathcal{B}} \sigma \,\} = \emptyset$$

This let us prove that $\rho \sqsubseteq_{\text{CLT}}^{\mathcal{B}} ?\mathtt{1}.\,\mathtt{1}$. It is routine work to check that the pair

$$(!(\hat{\rho}).\,\mathtt{1},\ ?\mathtt{1}.\,\mathtt{1})$$

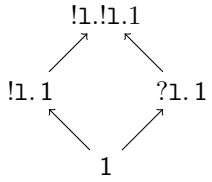
does not satisfy any of the points point (i) ... point (vii) of Lemma 8.1.5.

Now we define a relation $\mathcal{B}$ which is strongly total, but not reflexive; and then we exhibit a pair in $\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$, that does not satisfy any of the points in Lemma 8.1.5.

Let

$$\begin{aligned}\mathcal{B} = \mathcal{I}_{\mathsf{ho}} \setminus &\{(?\mathtt{1}.1,\ ?\mathtt{1}.1), (!\mathtt{1}.1,\ !\mathtt{1}.1)\}\\ \cup &\{(1\ !\mathtt{1}.1), (1,\ ?\mathtt{1}.1)\}\\ \cup &\{(?\mathtt{1}.1,\ !\mathtt{1}.!\mathtt{1}.1), (!\mathtt{1}.1,\ !\mathtt{1}.!\mathtt{1}.1)\}\end{aligned}$$

The relation $\mathcal{B}$ is not reflexive, as $!\mathtt{1}.1 \not\mathcal{B} !\mathtt{1}.1$; though, the relation is strongly total: for every $\rho \in \mathsf{SC}_{\text{HO}}$ we can show a σ such that $\rho\ \mathcal{B}\ \sigma$ or $\sigma\ \mathcal{B}\ \rho$. As to the pairs that we have added to $\mathcal{I}_{\mathsf{ho}}$ in the construction of $\mathcal{B}$, we depict them below, where the arrows connects the first elements to the second elements of the pairs.



The picture shows that if we think of $\mathcal{B}$ as a pre-order, then we can see that the terms $!\mathtt{1}.1$, and $?\mathtt{1}.1$ have a LUB and GLB, but they are *not* related by $\mathcal{B}$. This is the property of $\mathcal{B}$ that makes the example work.

Let $\rho_1 = ?(!\mathtt{1}.1).1$ and $\rho_2 = ?(?\mathtt{1}.1).1$. The bulk of the work is to prove that $\rho_1 \sqsubseteq^{\mathcal{B}}_{\text{CLT}} \rho_2$; to show this, it suffices to prove that the relation

$$\mathcal{R}' = \{\, (\rho_2,\ \sigma) \mid \rho_1 \dashv_{\mathcal{B}} \sigma \,\}$$

is a co-inductive $\mathcal{B}$-compliance relation. Lemma 3.3.10 ensures that we can prove a different result, that is that the relation

$$\mathcal{R} = \{\, (\rho_2,\ \sigma) \mid \rho_1 \dashv^{s}_{\mathcal{B}} \sigma \,\}$$

is a co-inductive $\mathcal{B}$-syntactic compliance relation. We are required to prove an implication: if $\rho_2\ \mathcal{R}\ \sigma$, then $(\rho_2,\ \sigma) \in \mathcal{F}^{\dashv^{s}}(\mathcal{R}, \mathcal{B})$. To prove the consequences, if $\rho'\ \mathcal{R}\ \sigma'$, then it is enough to a one step derivation of the form

$$\frac{\cdots}{\rho_2 \dashv^{s}_{\mathcal{B}} \sigma}$$

done by instantiating one of the rules in Figure 7.8.

Let $\rho\ \mathcal{R}\ \sigma$, either $\rho = 1$ or $\rho = \rho_2$. In the former case, we have the derivation

$$\frac{}{\rho \dashv^{s}_{\mathcal{B}} \sigma}\ [\text{A-GOAL}]$$

In the latter case, $\rho_1 \dashv^{s}_{\mathcal{B}} \sigma$. Given the shape of ρ_1, the fact that $\rho_1 \dashv^{s}_{\mathcal{B}} \sigma$ can be proven only by the derivation

$$\frac{1 \dashv^{s}_{\mathcal{B}} \sigma'}{\rho_1 \dashv^{s}_{\mathcal{B}} \sigma}\ \sigma = !(\hat{\sigma}).\sigma',\ ?(!\mathtt{1}.1) \bowtie_{\mathcal{B}} !(\hat{\sigma});\ [\text{R-ETA}]$$

From the definition of $\bowtie$, it follows that $\hat{\sigma}\ \mathcal{B} !\mathtt{1}.1$. The construction of $\mathcal{B}$ guarantees that $\hat{\sigma} = 1$. So far, we have proven that if $\rho_1 \dashv^{s}_{\mathcal{B}} \sigma$, then (a) $\sigma = !(1).\sigma'$ and (b) $1 \dashv^{s}_{\mathcal{B}} \sigma'$.

Since $1\ \mathcal{B} ?\mathtt{1}.1$, the definition of $\bowtie$ implies that $?(?\mathtt{1}.1) \bowtie_{\mathcal{B}} !(1)$; moreover, by construction, we have

$1 \mathcal{R} \sigma'$, thus we have the derivation

$$\frac{1 \dashv^s_{\mathcal{B}} \sigma'}{\rho_2 \dashv^s_{\mathcal{B}} \sigma} \; ?(?1.1) \bowtie_{\mathcal{B}} !(1); \; [\text{R-ETA}]$$

We have shown that every pair in $\mathcal{R}$, is also in $\mathcal{F}^{\dashv^s}(\mathcal{R}, \mathcal{B})$. Given the definition of $\mathcal{R}$, it follows that if $\rho_1 \dashv^s_{\mathcal{B}} \sigma$ then $\rho_2 \dashv^s_{\mathcal{B}} \sigma$. Thanks to Lemma 3.3.10, we have proven that $\rho_1 \sqsubseteq^{\mathcal{B}}_{\text{CLT}} \rho_2$.

To conclude the example, we have to show that the pair (ρ_1, ρ_2) does not satisfy any of the points (a) ... (g). Since $\rho_1 = ?(!1.1).1$ and $\rho_2 = ?(?1.1).1$, the only point that could be true is point (iii), but $!1.1 \not\mathcal{B} ?1.1$, thus point (iii) is false. □

Example B.0.7. [Transitivity is necessary]
In this example, we show that if $\mathcal{B}$ is a binary relation on session contracts, which is not transitive, then Lemma 8.1.10 is false, that is

$$\preceq^{\mathcal{B}}_{\text{CLT}} \not\subseteq \sqsubseteq^{\mathcal{B}}_{\text{CLT}}$$

To this aim, we define the following relation,

$$\mathcal{B} = \mathcal{I}_{\mathsf{ho}} \cup \{ (1, 0), (0, !1.1) \}$$

The relation $\mathcal{B}$ is not transitive, because $1 \mathcal{B} 0$ and $0 \mathcal{B} !1.1$, while $1 \not\mathcal{B} !1.1$.

Let $\rho_1 = !(0).1$ and let $\rho_2 = !(1).1$. We prove that $\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2$. It is enough to show a prefixed point of $\mathcal{F}^{\preceq_{\text{CLT}}}$, that contains the pair at hand. Let the candidate prefixed point be $\mathcal{R} = \{(\rho_1, \rho_2), (1, 1)\}$. We are required to prove that $\mathcal{R} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{R}, \mathcal{B})$; this amounts in showing that a) $(1, 1) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{R}, \mathcal{B})$, and b) $(\rho_1, \rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{R}, \mathcal{B})$. To prove both points we have to show an application of one of the rules in Figure 8.1, which has as consequence one of the two pairs at issue.

The proof of a) is the derivation

$$\frac{}{1 \preceq^{\mathcal{B}}_{\text{CLT}} 1} \; [\text{A-GOAL-C}]$$

The proof of b) is the derivation

$$\frac{1 \preceq^{\mathcal{B}}_{\text{CLT}} 1}{\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2} \; 1 \mathcal{B} 0; \; [\text{R-OUT-H}]$$

It follows that $\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2$.

Now we prove that $\rho_1 \not\preceq^{\mathcal{B}}_{\text{CLT}} \rho_2$. We have to exhibit a session contract σ, such that $\rho_1 \dashv_{\mathcal{B}} \sigma$, and $\rho_2 \not\dashv_{\mathcal{B}} \sigma$. Let $\sigma = ?(!1.1).1$; this is our candidate σ. To see why $\rho_1 \dashv_{\mathcal{B}} \sigma$, note that the relation $\mathcal{R} = \{ (\rho_1, \sigma), (1, 1) \}$ is contained in a co-inductive $\mathcal{B}$-compliance relation. To conclude the example, we have to prove that $\rho_2 \not\dashv_{\mathcal{B}} \sigma$. The witness that $\mathcal{B}$ is not transitive is the fact that $!(1) \not\bowtie_{\mathcal{B}} ?(!1.1)$. This implies that $\rho_2 \,||\, \sigma \overset{\tau}{\not\longrightarrow}$. Since $\rho_2 \overset{\checkmark}{\longrightarrow}_{\mathcal{B}}$, it follows that $\rho_2 \not\dashv_{\mathcal{B}} \sigma$.

In this example, using the fact that $\mathcal{B}$ is not transitive, we have exhibited two session contracts ρ_1, ρ_2, such that $\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_2$, and that $\rho_1 \not\preceq^{\mathcal{B}}_{\text{CLT}} \rho_2$. □

To prove that $\nu X.\mathcal{F}^{\preceq_{\text{CLT}}}(X, X)$ we used Proposition **??**. That proposition essentially states when the image via $\mathcal{F}^{\preceq_{\text{CLT}}}$ of two relations is a transitive relation. This is the case is *both* the parameters of $\mathcal{F}^{\preceq_{\text{CLT}}}$ are transitive. We prove this in the next two examples.

Example B.0.8. In this example we show that if $\mathcal{T}$ is a transitive relation on session contracts, and $\mathcal{B}$ is not, then $\mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B}, \mathcal{T})$ need not be transitive.

Let $\mathcal{T} = \{(1, 1)\}$, and let $\mathcal{B} = \{(1, 0), (0, !1.1)\}$ We exhibit three session contracts ρ_1, ρ_2, and ρ_3 such that

$$\{(\rho_1, \rho_2), (\rho_2, \rho_3)\} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B}, \mathcal{T}), \quad (\rho_1, \rho_3) \notin \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B}, \mathcal{T})$$

This proves that $\mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{T})$ is not transitive. Let

$$\rho_1 = !(1).\,1$$
$$\rho_2 = !(1).\,0$$
$$\rho_3 = !(1).!1.\,1$$

The proof that $(\rho_1,\,\rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{T})$ is the derivation

$$\frac{1 \preceq^{\mathcal{B}}_{\text{CLT}} 0}{!(1).\,1 \preceq^{\mathcal{B}}_{\text{CLT}} !(1).\,0}\ 1\ \mathcal{T}\ 1;\ [\text{R-OUT-H}]$$

The proof that $(\rho_2,\,\rho_3) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{T})$ is the derivation

$$\frac{0 \preceq^{\mathcal{B}}_{\text{CLT}} !1.\,1}{!(1).\,0 \preceq^{\mathcal{B}}_{\text{CLT}} !(1).!1.\,1}\ 1\ \mathcal{T}\ 1;\ [\text{R-OUT-H}]$$

To prove that $(\rho_1,\,\rho_3) \notin \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{T})$ we have to show that no inference rule let us derive

$$\frac{\ldots}{\rho_1 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_3}$$

Because of the form of ρ_1 (or, equivalently, ρ_3), the only inference rule that can be applied to derive $(\rho_1,\,\rho_3)$ is [R-OUT-H]; we see easily, though, that the premises of the rule are not satisfied. Let us consider the derivation

$$\frac{1 \preceq^{\mathcal{B}}_{\text{CLT}} !1.\,1}{!(1).\,1 \preceq^{\mathcal{B}}_{\text{CLT}} !(1).!1.\,1}\ 1\ \mathcal{T}\ 1;\ [\text{R-OUT-H}]$$

The premises require the pair $(1,\,!1.\,1)$ to be in the relation $\mathcal{B}$; this is not the case. Note, moreover, that $1\ \mathcal{B}!1.\,1$ is the witness of the fact that $\mathcal{B}$ is not transitive. This means that we have shown that $\mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{T})$ is not transitive, because $\mathcal{B}$ is not. □

Example B.0.9. In this example we show that if $\mathcal{T}$ is a transitive relation on session contracts, and $\mathcal{B}$ is not, then $\mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{T},\mathcal{B})$ need not be transitive. Let the relations $\mathcal{B}$ and $\mathcal{T}$ be defined as in the previous example, $\mathcal{T}= \{(1,\,1)\}$, and let $\mathcal{B}= \{(1,\,0),(0,\,!1.\,1)\}$; and let

$$\rho_1 = ?(1).\,1$$
$$\rho_2 = ?(0).\,1$$
$$\rho_3 = ?(!1.1).\,1$$

We prove that $\mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{T},\mathcal{B})$ is not transitive, by showing the following

$$\{(\rho_1,\,\rho_2),(\rho_2,\,\rho_3)\} \subseteq \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{T}),\quad (\rho_1,\,\rho_3) \notin \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{B},\mathcal{T})$$

The proof that $(\rho_1,\,\rho_2) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{T},\mathcal{B})$ is the ensuing derivation

$$\frac{1 \preceq^{\mathcal{B}}_{\text{CLT}} 1}{?(1).\,1 \preceq^{\mathcal{B}}_{\text{CLT}} ?(0).\,1}\ 1\ \mathcal{B}\ 0;\ [\text{R-IN-H}]$$

The proof that $(\rho_2 \preceq^{\mathcal{B}}_{\text{CLT}} \rho_3) \in \mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{T},\mathcal{B})$ is the ensuing derivation

$$\frac{1 \preceq^{\mathcal{B}}_{\text{CLT}} 1}{?(0).\,1 \preceq^{\mathcal{B}}_{\text{CLT}} ?(!1.1).\,1}\ 0\ \mathcal{B}!1.1;\ [\text{R-IN-H}]$$

Now we show that it is not possible to derive the pair $(\rho_1,\,\rho_3)$. Given the form of ρ_3, the only rule

that could let us derive the pair at issue is [R-In-H]; the side conditions of the rule, though are not satisfied:

$$\frac{1 \preceq^{\mathcal{B}}_{\text{CLT}} 1}{?(1).\,1 \preceq^{\mathcal{B}}_{\text{CLT}} ?(!1.1).\,1}\ 1\ \mathcal{B}!1.1;\ [\text{R-In-H}]$$

Plainly, we have $1\ \not\mathcal{B}!1.\,1$.

In this example, we have used the witness that $\mathcal{B}$ is not transitive, to prove that neither $\mathcal{F}^{\preceq_{\text{CLT}}}(\mathcal{T}, \mathcal{B})$ is transitive. □

Now we turn our attention the monotonicity of the function $\lambda X.\ \sqsubseteq^{\mathcal{X}}_{\text{CLT}}$. In the next two examples we show that if we let X range over relations that are not transitive, then

1. the function $\lambda X.\ \sqsubseteq^{\mathcal{X}}_{\text{CLT}}$ is **not** monotone
2. Lemma 8.1.15 is false

Example B.0.10. [Strong totality is necessary] Let $\mathcal{R}$ be a relation which is not strongly total. There exists a relation $\mathcal{R}'$ such that $\mathcal{R} \subseteq \mathcal{R}'$, and $\sqsubseteq^{\mathcal{R}}_{\text{CLT}} \not\subseteq \sqsubseteq^{\mathcal{R}'}_{\text{CLT}}$.

We can assume without loss of generality that there exists a session contract $\hat{\rho}$ such that $\hat{\rho}\ \not\mathcal{R}\ \hat{\sigma}$ for every $\hat{\sigma} \in \mathsf{SC}_{\text{HO}}$; that is $\mathcal{R}$ is not total. If $\mathcal{R}^{-1}$ is not total the argument is similar to one we use now.

Let $\rho_1 = !(\hat{\rho}).\,1$; thanks to the assumption on $\hat{\rho}$, we can prove that for every stable σ we have $\rho_1 \mid\mid \sigma \not\xrightarrow{\tau}_{\mathcal{R}}$; since $\rho \xrightarrow{\checkmark}_{\mathcal{R}}$ it follows that $\hat{\rho} \not\dashv_{\mathcal{R}} \sigma$. We can use this to prove that $\hat{\rho} \not\dashv_{\mathcal{R}} \sigma$ for every $\sigma \in \mathsf{SC}_{\text{HO}}$. Definition 8.1.1 ensures that

$$\{\, \sigma \in \mathsf{SC}_{\text{HO}} \mid \rho \dashv_{\mathcal{R}} \sigma \,\} = \emptyset$$

so we trivially have $\rho \sqsubseteq^{\mathcal{R}}_{\text{CLT}} \rho_2$ for every $\rho_2 \in \mathsf{SC}_{\text{HO}}$.

Let us fix $\rho_2 = !1.\,1$, and let $\mathcal{R}' = \mathcal{R} \cup \{(\hat{\rho}, \hat{\rho})\}$. We have $!(\hat{\rho}) \bowtie_{\mathcal{R}'} ?(\hat{\rho})$, and we can prove that the relation

$$\{(!(\hat{\rho}).\,1,\ ?(\hat{\rho}).\,1), (1,\ 1)\}$$

is a co-inductive compliance relation, hence $\rho \dashv_{\mathcal{R}'} ?(\hat{\rho}).\,1$.

To prove that $\rho \not\sqsubseteq^{\mathcal{R}'}_{\text{CLT}} \rho_2$, it suffices to observe that $\rho_2 \not\dashv_{\mathcal{R}'} ?(\hat{\rho}).\,1$. The observation follows from $\rho_2 \mid\mid \sigma \not\xrightarrow{\tau}$, $\rho_2 \not\xrightarrow{\tau}_{\mathcal{R}'}$, and condition (b) of Definition 7.2.4.

We have exhibit a relation $\mathcal{R}'$ such that $\mathcal{R} \subseteq \mathcal{R}'$ and that $\sqsubseteq^{\mathcal{R}}_{\text{CLT}} \not\subseteq \sqsubseteq^{\mathcal{R}'}_{\text{CLT}}$, so the lemma is proven. □

Example B.0.11. In this example we show that for Lemma 8.1.15 to be true, it is necessary that the relation $\mathcal{T}$ in its hypothesis be transitive, as long as we assume $\mathcal{R}$ reflexive.

First, we define two suitable relations.

$$\begin{aligned} \mathcal{R} &= \mathcal{I}_{\mathsf{ho}} \cup \{(!1.\,1,\ 1)\} \\ \mathcal{R}' &= \mathcal{R} \cup \{(0,\ !1.\,1)\} \end{aligned}$$

The relation $\mathcal{R}$ is reflexive, because it contains the identity relation; and the relation $\mathcal{R}'$ is not transitive, because

$$0\ \mathcal{R}!1.\,1, \qquad !1.\,1\ \mathcal{R}\ 1, \qquad 0\ \not\mathcal{R}'\ 1$$

By construction, we also know that $\mathcal{R} \subseteq \mathcal{R}'$.

(a) We prove that $?(!1.\,1).\,1 \sqsubseteq^{\mathcal{R}}_{\text{CLT}} ?(1).\,1$.

Suppose that $?(!1.\,1).\,1 \dashv_{\mathcal{R}} \sigma$; it follows that σ has to interact with $?(!1.\,1).\,1$, and this can happen only via the action $!(!1.\,1)$. From the definition of $\bowtie_{\mathcal{R}}$ it follows that σ can interact also with $?(1)$, and so $?(1).\,1 \dashv_{\mathcal{R}} \sigma$.

(b) Now we prove that $?(!\mathtt{1}.\,\mathtt{1}).\,\mathtt{1} \not\sqsubseteq^{\mathcal{R}'}_{\mathrm{CLT}} ?(\mathtt{1}).\,\mathtt{1}$.

We have to show a σ such that $?(!\mathtt{1}.\,\mathtt{1}).\,\mathtt{1} \dashv_{\mathcal{R}'} \sigma$, and $?(\mathtt{1}).\,\mathtt{1} \not\dashv_{\mathcal{R}'} \sigma$. Let $\sigma = !(\mathtt{0}).\,\mathtt{1}$. The definitions of $\bowtie$ and of $\mathcal{R}'$ ensure that $?(!\mathtt{1}.\,\mathtt{1}) \bowtie_{\mathcal{R}} !(\mathtt{0})$, thus we can prove that

$$\mathcal{S} = \{(?(!\mathtt{1}.\,\mathtt{1}).\,\mathtt{1},\, \sigma),\, (\mathtt{1},\, \mathtt{1})\}$$

is a co-inductive compliance; it follows that $?(!\mathtt{1}.\,\mathtt{1}).\,\mathtt{1} \dashv_{\mathcal{R}'} \sigma$.

Consider now the composition $?(\mathtt{1}).\,\mathtt{1} \parallel \sigma$; the witness that $\mathcal{R}'$ is not transitive is the fact that $\mathtt{0}\ \mathcal{R}'\ \mathtt{1}$, and this implies that $?(\mathtt{1}) \not\bowtie_{\mathcal{R}'} !(\mathtt{0})$. It follows that $?(\mathtt{1}).\,\mathtt{1} \parallel \sigma \overset{\tau}{\not\longrightarrow}$; since $?(\mathtt{1}).\,\mathtt{1} \overset{\checkmark}{\not\longrightarrow}_{\mathcal{R}'}$ Definition 7.2.4 ensures that $?(\mathtt{1}).\,\mathtt{1} \not\dashv_{\mathcal{R}'} \sigma$.

The argument we have unravelled shows that $\sqsubseteq^{\mathcal{R}}_{\mathrm{CLT}} \not\subseteq \sqsubseteq^{\mathcal{R}'}_{\mathrm{CLT}}$. □

In Lemma B.0.15 we prove that the function $\lambda X.\ \sqsubseteq^{X}_{\mathrm{CLT}}$ is monotone not only if X ranges over pre-orders, but also if X ranges over another kind of relations. We define these relations now.

Definition B.0.12. [Transitive identity]
We say that a relation $\mathcal{R} \subset A \times A$ is a *transitive identity*, if and only if for every $a\ \mathcal{R}\ b$ and $b\ \mathcal{R}\ c$ imply $a = c$; that is $[\mathcal{R}]^{+} \subseteq \mathcal{I}_A$. □

Notice that to be transitive identity is a property independent from the property of being a reflexive relation. We show this in the next example.

Example B.0.13. In this example we prove that the properties of being reflexive, and of being a transitive identity, are independent; that is

a) to be reflexive does not imply to be a transitive identity

b) to be a transitive identity does not imply to be reflexive

Let $\mathcal{R}$ be the relation

$$\mathcal{I}_{\mathsf{ho}} \cup \{(\mathtt{0},\, \mathtt{1})\}$$

As $\mathcal{I}_{\mathsf{ho}} \subset \mathcal{R}$, the relation $\mathcal{R}$ is reflexive. The proof that $\mathcal{R}$ is not a transitive identity is that $\{\,(\mathtt{0},\, \mathtt{1}), (\mathtt{1},\, \mathtt{1})\,\} \subset \mathcal{R}$ and $\mathtt{0} \neq \mathtt{1}$.

Let $\mathcal{S}$ be the relation

$$\mathcal{I}_{\mathsf{ho}} \setminus \{\,(\mathtt{1},\, \mathtt{1}),\, (!\mathtt{1}.\,\mathtt{1}, !\mathtt{1}.\,\mathtt{1})\,\} \cup \{\,(\mathtt{1},\, !\mathtt{1}.\,\mathtt{1}),\, (!\mathtt{1}.\,\mathtt{1},\, \mathtt{1})\,\}$$

The relation $\mathcal{S}$ is not reflexive, because $\mathtt{1}\ \not\mathcal{S}\ \mathtt{1}$. We prove that $\mathcal{S}$ is a transitive identity. Let $\sigma\ \mathcal{S}\ \sigma'$ and $\sigma'\ \mathcal{S}\ \sigma''$; we have to prove that $\sigma = \sigma''$. If $\sigma = \mathtt{1}$, then by construction it must be $\sigma' = !\mathtt{1}.\,\mathtt{1}$; hence, again by construction, it must be $\sigma'' = \mathtt{1}$; we have thus $\sigma = \mathtt{1} = \sigma''$. A similar argument can be used if $\sigma = !\mathtt{1}.\,\mathtt{1}$.

If $\sigma \notin \{\mathtt{1}, !\mathtt{1}.\,\mathtt{1}\}$, then $\sigma\ \mathcal{I}_{\mathsf{ho}}\ \sigma'$, thus $\sigma = \sigma'$. It follows that $\sigma' \notin \{\mathtt{1}, !\mathtt{1}.\,\mathtt{1}\}$, and so $\sigma'\ \mathcal{I}_{\mathsf{ho}}\ \sigma''$. It follows $\sigma = \sigma' = \sigma''$. □

The reflexive relations and the transitive identities are not disjoint; there is *one* relation which is reflexive and a transitive identity.

Proposition B.0.14. Let $\mathcal{R}$ be a reflexive transitive identity on the set A. The relation $\mathcal{R}$ is the identity.

Proof. We have to prove that $\mathcal{R} = \mathcal{I}_A$. The set inclusion $\mathcal{I}_A \subset \mathcal{R}$ is true because $\mathcal{R}$ is reflexive, so we have to check only why $\mathcal{R} \subseteq \mathcal{I}_A$. Let $a\ \mathcal{R}\ b$; we have to show that $a\ \mathcal{I}_A\ b$. As $\mathcal{R}$ is reflexive we know that $a\ \mathcal{R}\ a$; it follows that $a\ [\mathcal{R}]^{+}\ b$. The hypothesis of $\mathcal{R}$ being a transitive identity ensures that $a\ \mathcal{I}_A\ b$. □

The next lemma implies that to let X range over pre-orders is a condition *sufficient* for $\lambda X.\ \sqsubseteq_{\text{CLT}}^{X}$ to be monotone, but it not necessary.

Lemma B.0.15. Let $\mathcal{R}$ be a strongly total transitive identity on session contracts. If $\mathcal{R} \subseteq \mathcal{R}'$ then $\sqsubseteq_{\text{CLT}}^{\mathcal{R}} \subseteq \sqsubseteq_{\text{CLT}}^{\mathcal{R}'}$.

Proof. We have to prove that for every $\rho_1, \rho_2 \in \mathsf{SC}_{\text{HO}}$, if $\rho_1 \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \rho_2$ then $\rho_1 \dashv_{\mathcal{R}'} \sigma$ implies $\rho_2 \dashv_{\mathcal{R}'} \sigma$.

Thanks to Lemma 7.2.12, the latter implication is equivalent to if $\rho_1 \dashv^{s}_{\mathcal{R}'} \sigma$ then $\rho_2 \dashv^{s}_{\mathcal{R}'} \sigma$. Lemma 7.2.7 ensures that it is enough to prove that if $\rho_1 \dashv^{s}_{\mathcal{R}'} \sigma$ then $\rho_2 \dashv^{s}_{\mathcal{R}'} \sigma$. We prove this fact. Let

$$\mathcal{S} = \{\, (\rho_1,\, \sigma) \mid \rho_1 \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \rho_2,\, \rho_1 \dashv^{s}_{\mathcal{R}'} \sigma \,\}$$

Plainly, $\rho_2\ \mathcal{S}\ \sigma$; so to prove $\rho_2 \dashv^{s}_{\mathcal{R}'} \sigma$ it suffices to show that $\mathcal{S}$ is a co-inductive $\mathcal{R}'$-syntactic compliance relation.

The argument is by case analysis on the depth of ρ_2 and σ, and then on the form of $\text{UNF}(\rho_1)$. Suppose that $\mathit{depth}(\rho_2) + \mathit{depth}(\sigma) > 0$. If $\rho_2 = \mu x.\, \rho_2$, then note that $\mathit{depth}(\rho_2) > 0$. Corollary 8.1.4 and Lemma 7.2.7 ensure the following facts,

$$\text{UNF}(\rho_1) \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \text{UNF}(\rho_2), \qquad \text{UNF}(\rho_1) \dashv^{s}_{\mathcal{R}'} \text{UNF}(\sigma)$$

The definition of $\mathcal{S}$ ensures that $\text{UNF}(\rho_2)\ \mathcal{S}\ \text{UNF}(\sigma)$. It follows that we can apply [R-Unfold],

$$\frac{\text{UNF}(\rho_2) \dashv^{s}_{\mathcal{R}'} \text{UNF}(\sigma)}{\rho_2 \dashv^{s}_{\mathcal{R}'} \sigma}\ \mathit{depth}(\rho_2) + \mathit{depth}(\sigma) > 0;\ [\text{R-Unfold}]$$

Now suppose that $\mathit{depth}(\rho_2) + \mathit{depth}(\sigma) = 0$. Now we reason by case analysis on $\text{UNF}(\rho_1)$; all the cases are straightforward, except the higher-order ones. For instance $\rho_1 = ?(\hat{\rho}_1).\rho_1'$. We discuss this case. If $\text{UNF}(\rho_1) = ?(\hat{\rho}_1).\rho_1'$, we show that we can apply rule [R-Eta] to prove $\rho_2 \dashv^{s}_{\mathcal{R}'} \sigma$.

We have to prove four things: 1) $\rho_2 = ?(\hat{\rho}_2).\rho_2'$, 2) $\sigma = \theta.\sigma'$, 3) $?(\hat{\rho}_2) \bowtie_{\mathcal{R}'} \theta$, 4) $\rho_2'\ \mathcal{R}'\ \sigma'$.

We know that $\text{UNF}(\rho_1) \dashv^{s}_{\mathcal{R}'} \sigma$ and that $\text{UNF}(\rho_1) \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \rho_2$.

1) As by hypothesis $\mathcal{R}$ is strongly total, we know that $\hat{\rho}\ \mathcal{R}\ \hat{\rho}_1$ for some $\hat{\rho} \in \mathsf{SC}_{\text{HO}}$, so the definition of $\bowtie$ ensures that $?(\hat{\rho}_1) \bowtie_{\mathcal{R}} !(\hat{\rho})$.

   Let $\rho_1' \dashv^{s}_{\mathcal{R}} \sigma''$; Lemma 7.2.17 and the totality of $\mathcal{R}$ ensure that one such σ'' exists (ie. $\text{Dual}(\rho_1', \mathcal{R}) \neq \emptyset$). We can prove that $\text{UNF}(\rho_1) \dashv^{s}_{\mathcal{R}} !(\hat{\rho}).\sigma''$.

   From $\rho_1 \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \rho_2$, it follows that $\rho_2 \dashv^{s}_{\mathcal{R}} !(\hat{\rho}).\sigma''$, thus $\rho_2 = ?(\hat{\rho}_2).\rho_2'$, $\hat{\rho}\ \mathcal{R}\ \hat{\rho}_2$, and $\rho_1' \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \rho_2'$. We have proven 1). By hypothesis $\mathcal{R}$ is a transitive identity, thus from $\hat{\rho}_1\ \mathcal{R}\ \hat{\rho}$ and $\hat{\rho}\ \mathcal{R}\ \hat{\rho}_2$, it follows that

$$\hat{\rho}_1 = \hat{\rho}_2 \tag{B.1}$$

2) The fact that $\text{UNF}(\rho_1) \dashv^{s}_{\mathcal{R}'} \sigma$ implies that $\sigma = !(\hat{\sigma}).\sigma'$, $\hat{\sigma}\ \mathcal{R}'\ \hat{\rho}_1$, and $\rho_1' \dashv^{s}_{\mathcal{R}'} \sigma'$. We have proven 2), with $\theta = !(\hat{\sigma})$.

3) We prove $?(\hat{\rho}_2) \bowtie_{\mathcal{R}'} !(\hat{\sigma})$. It suffices to show that $\hat{\sigma}\ \mathcal{R}'\ \hat{\rho}_2$. We know that $\hat{\sigma}\ \mathcal{R}'\ \hat{\rho}_1$, thus the equality in B.1 ensures that $\hat{\sigma}\ \mathcal{R}'\ \hat{\rho}_2$.

4) To prove 4) we have to exhibit a ρ'' such that $\rho'' \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \rho_2'$ and $\rho'' \dashv^{s}_{\mathcal{R}'} \sigma'$. The ρ'' we are after is ρ_1', for we have already proven that

$$\rho_1' \sqsubseteq_{\text{CLT}}^{\mathcal{R}} \rho_2' \text{ and } \rho_1' \dashv^{s}_{\mathcal{R}'} \sigma'$$

□

We are now aware of two sets of conditions which are sufficient to prove that $\lambda X.\ \sqsubseteq^X_{\mathrm{CLT}}$ is monotone. These conditions essentially are that X range over pre-orders, or over transitive identities.

We prove that these conditions on X are necessary for $\lambda X.\ \sqsubseteq^X_{\mathrm{CLT}}$ to be monotone.

Example B.0.16. [Reflexive relations or transitive identities are necessary]
In this example we show that for the function $\lambda X.\ \sqsubseteq^X_{\mathrm{CLT}}$ to be monotone, it is necessary that $\mathcal{R}$ be a reflexive relation or a transitive identity.

We define an $\mathcal{R}$ which is nor reflexive, neither a transitive identity; and a transitive $\mathcal{R}'$ such that $\mathcal{R} \subseteq \mathcal{R}'$, and we prove that $\sqsubseteq^{\mathcal{R}}_{\mathrm{CLT}} \not\subseteq \sqsubseteq^{\mathcal{R}'}_{\mathrm{CLT}}$.

Consider the following relations.

$$\begin{array}{lcl} \mathcal{R} & = & \mathcal{I} \cup \{(1,\ !\mathtt{1}.\,1), (1, !\mathtt{1}.!\mathtt{1}.\,1), (!\mathtt{1}.\,1,\ ?\mathtt{1}.\,1)\} \setminus \{(!\mathtt{1}.\,1,\ !\mathtt{1}.\,1)\} \\ \mathcal{R}' & = & \mathcal{R}' \cup \{(!(1).\,1,\ !\mathtt{1}.\,1),\ (!(1).\,1,\ ?\mathtt{1}.\,1)\} \end{array}$$

The relation $\mathcal{R}$ is not reflexive, because $!\mathtt{1}.\,1 \not\mathcal{R} !\mathtt{1}.\,1$, and it is not a transitive identity, because $\{(1,\,1), (1, !\mathtt{1}.!\mathtt{1}.\,1)\} \subset \mathcal{R}$ and $1 \neq !\mathtt{1}.!\mathtt{1}.\,1$. We can prove that the relation $\mathcal{R}$ is strongly total; the pair $(!\mathtt{1}.\,1,\ ?\mathtt{1}.\,1)$ is in it exactly to this aim. Plainly, $\mathcal{R} \subseteq \mathcal{R}'$, moreover the relation $\mathcal{R}'$ is transitive.

We prove that

$$?(!\mathtt{1}.\,1).\,1 \sqsubseteq^{\mathcal{R}}_{\mathrm{CLT}} ?(!\mathtt{1}.!\mathtt{1}.\,1).\,1$$

Let $?(!\mathtt{1}.\,1).\,1 \dashv_{\mathcal{R}} \sigma$; since $?(!\mathtt{1}.\,1).\,1 \not\xrightarrow{\checkmark}_{\mathcal{R}}$, it follows that σ must interact with $?(!\mathtt{1}.\,1).\,1$; we can prove that σ performs the action $!(1)$. As $?(!\mathtt{1}.!\mathtt{1}.\,1) \bowtie_{\mathcal{R}} !(1)$, it follows that $?(!\mathtt{1}.!\mathtt{1}.\,1).\,1 \dashv_{\mathcal{R}} \sigma$.

Now we prove that

$$?(!\mathtt{1}.\,1).\,1 \not\sqsubseteq^{\mathcal{R}'}_{\mathrm{CLT}} ?(!\mathtt{1}.!\mathtt{1}.\,1).\,1$$

On the one hand, we can prove that $?(!\mathtt{1}.\,1) \bowtie_{\mathcal{R}'} !(!(1).\,1)$; this is enough to show that the following set

$$\{(?(!\mathtt{1}.\,1).\,1,\ !(!(1).\,1).\,1)), (1,\,1)\}$$

is a co-inductive compliance relation (with respect to $\mathcal{R}'$). It follows that

$$?(!\mathtt{1}.\,1).\,1 \dashv_{\mathcal{R}'} !(!(1).\,1).\,1$$

On the other hand, we have $?(!\mathtt{1}.!\mathtt{1}.\,1) \not\bowtie_{\mathcal{R}'} !(!(1).\,1)$, thus $?(!\mathtt{1}.!\mathtt{1}.\,1).\,1 \,||\, !(!(1).\,1).\,1 \not\xrightarrow{\tau}$; moreover, $?(!\mathtt{1}.!\mathtt{1}.\,1).\,1 \not\xrightarrow{\checkmark}_{\mathcal{R}'}$, thus

$$?(!\mathtt{1}.!\mathtt{1}.\,1).\,1 \not\dashv_{\mathcal{R}'} !(!(1).\,1).\,1$$

As $\mathcal{R} \subseteq \mathcal{R}'$ and $\sqsubseteq^{\mathcal{R}}_{\mathrm{CLT}} \not\subseteq \sqsubseteq^{\mathcal{R}'}_{\mathrm{CLT}}$, we have proven that $\mathcal{R}$ has to be reflexive or a transitive identity, in order for $\lambda X.\ \sqsubseteq^X_{\mathrm{CLT}}$ to be monotone. □

Appendix C

Monotone functionals

Throughout the this thesis we have used (co)inductive definitions; that is we have used least and greatest fixed points of suitable rule functionals. To ensure that these fixed points exist, it is necessary that the functionals be monotone. The statements that guarantee the monotonicity of the functionals we defined are collected in this appendix. Their proofs are obvious, and we omit them.

Lemma C.0.17. The rule functional $\mathcal{F}^{\preccurlyeq^{\mathsf{fo}}_{\mathsf{sbt}}}$ is monotone.

Lemma C.0.18. The functional $\mathcal{F}^{\dashv}$ is monotone.

Lemma C.0.19. The functional $\mathcal{F}^{\dashv_{\text{P2P}}}$ is monotone.

Lemma C.0.20. The rule functional $\mathcal{F}^{\dashv^s}_{\text{MUST}^{\text{S}}}$ is monotone.

Lemma C.0.21. The rule functional $\mathcal{F}_{\Downarrow}$ is monotone.

Lemma C.0.22. The rule functional $\mathcal{F}_{\Longrightarrow\!\!\!\!\!\!\diagup}$ is monotone.

Lemma C.0.23. The rule functional $\mathcal{F}_{\Downarrow_{\text{CLT}}}$ is monotone.

Lemma C.0.24. The rule functional $\mathcal{F}_{\downarrow\downarrow}$ is monotone.

Lemma C.0.25. The rule functional $\mathcal{F}_{\mathsf{usbl}}$ is monotone.

Lemma C.0.26. The rule functional $\mathcal{F}^{\precsim^{syn}_{\text{SVR}}}$ is monotone.

Lemma C.0.27. The rule functional $\mathcal{F}^{\precsim^{syn}_{\text{CLT}}}$ is monotone.

Lemma C.0.28. The rule functional $\mathcal{F}_{\text{DUAL}}$ is monotone.

Lemma C.0.29. The rule functional $\mathcal{F}^{\preceq^{syn}_{\text{CLT}}}$ is monotone.

Lemma C.0.30. The rule functional $\mathcal{F}^{\preccurlyeq^{\mathsf{sbt}}}$ is monotone.

Lemma C.0.31. The rule functional $\mathcal{F}^{\dashv}$ is monotone in its first variable.

Lemma C.0.32. The rule functional $\mathcal{F}^{\dashv^s}$ is monotone in its first variable.

Lemma C.0.33. The rule functional $\mathcal{F}_{\text{DUAL}}$ is monotone in its first variable.

Lemma C.0.34. The rule functional $\mathcal{F}^{\sqsubset^{\mathsf{ho}}_{\text{CLT}}}$ is monotone.

Lemma C.0.35. The rule functional $\mathcal{F}^{\precsim_{\text{CLT}}}$ is monotone in its first variable.

Lemma C.0.36. The rule functional $\mathcal{F}^{\precsim_{\text{SVR}}}$ is monotone in its first variable.

Proposition C.0.37. The rule functional $\mathcal{F}_{+}$ is monotone in its first argument.

Lemma C.0.38. The rule functional $\mathcal{F}_{\vdash}$ is monotone.

References

Acciai, L. and M. Boreale (2008). A type system for client progress in a service-oriented calculus. See Degano et al. [2008], pp. 642–658.

Aceto, L., A. Ingólfsdóttir, K. G. Larsen, and J. Srba (2007). *Reactive Systems: Modelling, Specification and Verification*. New York, NY, USA: Cambridge University Press.

Arnold, A. and D. Niwiński (2001). *Rudiments of μ-calculus*. Studies in Logic and the Foundations of Mathematics. Elsevier.

Barbanera, F. and U. de'Liguoro (2010). Two notions of sub-behaviour for session-based client/server systems. In T. Kutsia, W. Schreiner, and M. Fernández (Eds.), *PPDP*, pp. 155–164. ACM.

Barwise, J. and L. Moss (1996, August). *Vicious Circles*. (Center for the Study of Language and Information.

Bernardi, G., M. Bugliesi, D. Macedonio, and S. Rossi (2008). A theory of adaptable contract-based service composition. In V. Negru, T. Jebelean, D. Petcu, and D. Zaharie (Eds.), *SYNASC*, pp. 327–334. IEEE Computer Society.

Bernardi, G. and M. Hennessy. Modelling session types using contracts. *Mathematical Structures in Computer Science*.

Bernardi, G. and M. Hennessy (2011, Aug). Modelling session types using contracts. Technical Report TCD-CS-2011-7, University of Dublin, Trinity College.

Bernardi, G. and M. Hennessy (2012). Modelling session types using contracts. In S. Ossowski and P. Lecca (Eds.), *SAC*, pp. 1941–1946. ACM.

Bernardo, M., L. Padovani, and G. Zavattaro (Eds.) (2009). *Formal Methods for Web Services, 9th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2009, Bertinoro, Italy, June 1-6, 2009, Advanced Lectures*, Volume 5569 of *Lecture Notes in Computer Science*. Springer.

Best, E. (Ed.) (1993). *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, Volume 715 of *Lecture Notes in Computer Science*. Springer.

Brandt, M. and F. Henglein (1998). Coinductive axiomatization of recursive type equality and subtyping. *Fundam. Inform. 33*(4), 309–338.

Bravetti, M. and G. Zavattaro (2009). Contract-based discovery and composition of web services. See Bernardo et al. [2009], pp. 261–295.

Brookes, S. D., C. A. R. Hoare, and A. W. Roscoe (1984). A theory of communicating sequential processes. *J. ACM 31*(3), 560–599.

Bugliesi, M., D. Macedonio, L. Pino, and S. Rossi (2010). Compliance preorders for web services. In C. Laneve and J. Su (Eds.), *Web Services and Formal Methods*, Volume 6194 of *Lecture Notes in Computer Science*, pp. 76–91. Springer Berlin / Heidelberg.

Caires, L. and F. Pfenning (2010). Session types as intuitionistic linear propositions. In P. Gastin and F. Laroussinie (Eds.), *CONCUR*, Volume 6269 of *Lecture Notes in Computer Science*, pp. 222–236. Springer.

Carpineti, S. (2007, March). *Data and Behavioural Contracts for Web Services*. Ph. D. thesis, Università di Bologna – Università di Padova.

Carpineti, S., G. Castagna, C. Laneve, and L. Padovani (2006). A formal account of contracts for web services. In M. Bravetti, M. Núñez, and G. Zavattaro (Eds.), *WS-FM*, Volume 4184 of *Lecture Notes in Computer Science*, pp. 148–162. Springer.

Castagna, G., M. Dezani-Ciancaglini, E. Giachino, and L. Padovani (2009). Foundations of session types. See Porto and López-Fraguas [2009], pp. 219–230.

Castagna, G. and A. Frisch (2005). A gentle introduction to semantic subtyping. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung (Eds.), *ICALP*, Volume 3580 of *Lecture Notes in Computer Science*, pp. 30–34. Springer.

Castagna, G., N. Gesbert, and L. Padovani (2009). A theory of contracts for web services. *ACM Trans. Program. Lang. Syst. 31*(5), 1–61. Supersedes the article in POPL '08.

Cerone, A. and M. Hennessy (2010). Process behaviour: Formulae vs. tests (extended abstract). In S. B. Fröschle and F. D. Valencia (Eds.), *EXPRESS'10*, Volume 41 of *EPTCS*, pp. 31–45.

Christensen, S., Y. Hirshfeld, and F. Moller (1993). Bisimulation equivalence is decidable for basic parallel processes. See Best [1993], pp. 143–157.

Cleaveland, R. and M. Hennessy (1993). Testing equivalence as a bisimulation equivalence. *Formal Asp. Comput. 5*(1), 1–20.

Courcelle, B. (1983). Fundamental properties of infinite trees. *Theor. Comput. Sci. 25*, 95–169.

Dardha, O., E. Giachino, and D. Sangiorgi (2012). Session types revisited. In D. D. Schreye, G. Janssens, and A. King (Eds.), *PPDP*, pp. 139–150. ACM.

Davey, B. A. and H. A. Priestley (2002). *Introduction to Lattices and Order (2. ed.)*. Cambridge University Press.

De Nicola, R. (1983). A complete set of axioms for a theory of communicating sequential processes. In M. Karpinski (Ed.), *FCT*, Volume 158 of *Lecture Notes in Computer Science*, pp. 115–126. Springer.

De Nicola, R. (1985). Two complete axiom systems for a theory of communicating sequential processes. *Information and Control 64*(1-3), 136–172.

De Nicola, R. and M. Hennessy (1984). Testing equivalences for processes. *Theoretical Computer Science 34*, 83–133.

De Nicola, R. and M. Hennessy (1987). CCS without τ's. In H. Ehrig, R. A. Kowalski, G. Levi, and U. Montanari (Eds.), *TAPSOFT, Vol.1*, Volume 249 of *Lecture Notes in Computer Science*, pp. 138–152. Springer.

Degano, P., R. De Nicola, and J. Meseguer (Eds.) (2008). *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, Volume 5065 of *Lecture Notes in Computer Science*. Springer.

Demangeon, R. and K. Honda (2011). Full abstraction in a subtyped pi-calculus with linear types. In J.-P. Katoen and B. König (Eds.), *CONCUR*, Volume 6901 of *Lecture Notes in Computer Science*, pp. 280–296. Springer.

Dezani-Ciancaglini, M. and U. de'Liguoro (2009). Sessions and session types: An overview. In C. Laneve and J. Su (Eds.), *WS-FM*, Volume 6194 of *Lecture Notes in Computer Science*, pp. 1–28. Springer.

Dezani-Ciancaglini, M., U. de'Liguoro, and N. Yoshida (2007). On progress for structured communications. In G. Barthe and C. Fournet (Eds.), *TGC*, Volume 4912 of *Lecture Notes in Computer Science*, pp. 257–275. Springer.

Dezani-Ciancaglini, M., S. Drossopoulou, D. Mostrous, and N. Yoshida (2009). Objects and session types. *Inf. Comput. 207*(5), 595–641.

Fähndrich, M., M. Aiken, C. Hawblitzel, O. Hodson, G. C. Hunt, J. R. Larus, and S. Levi (2006). Language support for fast and reliable message-based communication in singularity os. In Y. Berbers and W. Zwaenepoel (Eds.), *EuroSys*, pp. 177–190. ACM.

Garralda, P., A. B. Compagnoni, and M. Dezani-Ciancaglini (2006). Bass: boxed ambients with safe sessions. In A. Bossi and M. J. Maher (Eds.), *PPDP*, pp. 61–72. ACM.

Gay, S. J. (2008). Bounded polymorphism in session types. *Mathematical Structures in Computer Science 18*(5), 895–930.

Gay, S. J., N. Gesbert, and A. Ravara (2008, May). Session types as generic process types. In *PLACES*.

Gay, S. J., N. Gesbert, A. Ravara, and V. T. Vasconcelos (2012). Modular session types for objects. *CoRR abs/1205.5344*.

Gay, S. J. and M. Hole (1999). Types and subtypes for client-server interactions. In S. D. Swierstra (Ed.), *ESOP*, Volume 1576 of *Lecture Notes in Computer Science*, pp. 74–90. Springer.

Gay, S. J. and M. Hole (2005). Subtyping for session types in the pi calculus. *Acta Inf. 42*(2-3), 191–225.

Gay, S. J., V. Vasconcelos, and A. Ravara (2003, February 11). Session types for inter-process communication. Technical Report TR-2003-133, Department of Computing Science, University of Glasgow.

Gay, S. J. and V. T. Vasconcelos (2010). Linear type theory for asynchronous session types. *J. Funct. Program. 20*(1), 19–50.

Gay, S. J., V. T. Vasconcelos, A. Ravara, N. Gesbert, and A. Z. Caldeira (2010). Modular session types for distributed object-oriented programming. In M. V. Hermenegildo and J. Palsberg (Eds.), *POPL*, pp. 299–312. ACM.

Giachino, E. (2009, December). *Session Types: Semantic Foundations and Object-Oriented Applications*. Ph. D. thesis, Università degli Studi di Torino – Université Paris 7.

Girard, J.-Y. (1987). Linear logic. *Theor. Comput. Sci. 50*, 1–102.

Grillo, G. (2013, February). The end of the third republic. `http://www.beppegrillo.it/en/2013/02/the_end_of_the_third_republic.html`.

Halmos, P. (1960). *Naive Set Theory*. Undergraduate Texts in Mathematics. Princeton, NJ.

Hauslohner, A. (2011). Dispatch from 'Free Libya': The Right to Laugh at Gaddafi. `http://www.time.com/time/world/article/0,8599,2053198,00.html`.

Hennessy, M. (1985). *Algebraic theory of processes*. Cambridge, MA, USA: MIT Press.

Hennessy, M. (2007). *A Distributed Pi-Calculus*. Cambridge University Press.

Hennessy, M., J. Rathke, and N. Yoshida (2005). safedpi: a language for controlling mobile code. *Acta Inf. 42*(4-5), 227–290.

Honda, K. (1993). Types for dynamic interaction. See Best [1993], pp. 509–523.

Honda, K., E. R. B. Marques, F. Martins, N. Ng, V. T. Vasconcelos, and N. Yoshida (2012). Verification of mpi programs using session types. In J. L. Träff, S. Benkner, and J. J. Dongarra (Eds.), *EuroMPI*, Volume 7490 of *Lecture Notes in Computer Science*, pp. 291–293. Springer.

Honda, K., V. T. Vasconcelos, and M. Kubo (1998). Language primitives and type discipline for structured communication-based programming. In C. Hankin (Ed.), *ESOP*, Volume 1381 of *Lecture Notes in Computer Science*, pp. 122–138. Springer.

Honda, K., N. Yoshida, and M. Carbone (2008). Multiparty asynchronous session types. In G. C. Necula and P. Wadler (Eds.), *POPL*, pp. 273–284. ACM.

Hu, R., N. Yoshida, and K. Honda (2008). Session-based distributed programming in java. In J. Vitek (Ed.), *ECOOP*, Volume 5142 of *Lecture Notes in Computer Science*, pp. 516–541. Springer.

Igarashi, A. and N. Kobayashi (2004, January). A generic type system for the pi-calculus. *Theor. Comput. Sci. 311*, 121–163.

Imai, K., S. Yuen, and K. Agusa (2010). Session type inference in haskell. In *PLACES*.

Laneve, C. and L. Padovani (2007). The must preorder revisited. In *Proceedings of the 18th international conference on Concurrency Theory*, Berlin, Heidelberg, pp. 212–225. Springer-Verlag.

Laneve, C. and L. Padovani (2008). The pairing of contracts and session types. See Degano et al. [2008], pp. 681–700.

Mendelson, E. (1997). *Introduction to Mathematical Logic: Fourth Edition*. Wadsworth & Brooks/Cole mathematics series. Chapman & Hall.

Mezzina, L. G. (2008). How to infer finite session types in a calculus of services and sessions. In D. Lea and G. Zavattaro (Eds.), *COORDINATION*, Volume 5052 of *Lecture Notes in Computer Science*, pp. 216–231. Springer.

Milne, G. and R. Milner (1979). Concurrent processes and their syntax. *J. ACM 26*(2), 302–321.

Milner, R. (1975). Processes: A mathematical model of computing agents. In H. Rose and J. Shepherdson (Eds.), *Logic Colloquium '73 Proceedings of the Logic Colloquium*, Volume 80 of *Studies in Logic and the Foundations of Mathematics*, pp. 157 – 173. Elsevier.

Milner, R. (1980). *A Calculus of Communicating Systems*, Volume 92 of *Lecture Notes in Computer Science*. Springer.

Milner, R. (1989). *Communication and concurrency*. PHI Series in computer science. Prentice Hall.

Milner, R. (1999). *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press.

Milner, R., J. Parrow, and D. Walker (1992). A calculus of mobile processes, i. *Inf. Comput. 100*(1), 1–40.

Mostrous, D. and N. Yoshida (2007). Two session typing systems for higher-order mobile processes. In S. R. D. Rocca (Ed.), *TLCA*, Volume 4583 of *Lecture Notes in Computer Science*, pp. 321–335. Springer.

Neubauer, M. and P. Thiemann (2004). An implementation of session types. In B. Jayaraman (Ed.), *PADL*, Volume 3057 of *Lecture Notes in Computer Science*, pp. 56–70. Springer.

Padovani, L. (2010). Contract-based discovery of web services modulo simple orchestrators. *Theor. Comput. Sci. 411*(37), 3328–3347.

Padovani, L. (2011). Fair Subtyping for Multi-Party Session Types. In *Proceedings of the 13th Conference on Coordination Models and Languages*, Volume LNCS 6721, pp. 127–141. Springer.

Padovani, L. (2013, January). Fair Subtyping for Multi-Party Session Types. Submitted for publication. Available at `http://www.di.unito.it/~padovani/Papers/FairSubtypingLong.pdf`.

Pierce, B. C. (2002, February). *Types and Programming Languages*. MIT Press.

Pierce, B. C. and D. Sangiorgi (1996). Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science 6*(5), 409–453.

Porto, A. and F. J. López-Fraguas (Eds.) (2009). *Proceedings of the 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, September 7-9, 2009, Coimbra, Portugal*. ACM.

Pucella, R. and J. A. Tov (2008). Haskell session types with (almost) no class. In A. Gill (Ed.), *Haskell*, pp. 25–36. ACM.

Rensink, A. and W. Vogler (2007). Fair testing. *Information and Computation 205*(2), 125–198.

Rose, M. (1988, November). Post Office Protocol: Version 3. RFC 1081. Obsoleted by RFC 1225.

Sackman, M. and S. Eisenbach (2008, July). Session Types in Haskell: Updating Message Passing for the 21st Century. Technical report, Imperial College London.

Sangiorgi, D. (1993). *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. Ph. D. thesis, The University of Edinburgh.

Sangiorgi, D. (2012). *An introduction to bisimulation and coinduction*. Cambridge, New York: Cambridge University Press.

Sangiorgi, D. and J. Rutten (2011). *Advanced Topics in Bisimulation and Coinduction*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Sangiorgi, D. and D. Walker (2001). *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press.

Takeuchi, K., K. Honda, and M. Kubo (1994). An interaction-based language and its typing system. In C. Halatsis, D. G. Maritsas, G. Philokyprou, and S. Theodoridis (Eds.), *PARLE*, Volume 817 of *Lecture Notes in Computer Science*, pp. 398–413. Springer.

AWS Team (2012, July 2). Summary of the aws service event in the us east region.

OASIS (2007). Web services business process execution language version 2.0.

w3c (2004, 11 February). Web services architecture requirements.

Treynor, B. (2011, March 1). Gmail back soon for everyone.

Vasconcelos, V. T. (2009a). Fundamentals of session types. See Bernardo et al. [2009], pp. 158–186.

Vasconcelos, V. T. (2009b). Session types for linear multithreaded functional programming. See Porto and López-Fraguas [2009], pp. 1–6.

Vasconcelos, V. T., S. J. Gay, and A. Ravara (2006). Type checking a multithreaded functional language with session types. *Theor. Comput. Sci. 368*(1-2), 64–87.

Wadler, P. (2012). Propositions as sessions. In P. Thiemann and R. B. Findler (Eds.), *ICFP*, pp. 273–286. ACM.

Winskel, G. (1993). *The formal semantics of programming languages: an introduction.* Cambridge, MA, USA: MIT Press.

Woodward, P. (2011). The role of the internet in the libyan uprising. `"http://warincontext.org/2011/02/23/the-role-of-the-internet-in-the-libyan-uprising/"`.

Yoshida, N. (2004). Channel dependent types for higher-order mobile processes. In N. D. Jones and X. Leroy (Eds.), *POPL*, pp. 147–160. ACM.

Yoshida, N. and M. Hennessy (2002). Assigning types to processes. *Inf. Comput. 174*(2), 143–179.

Result Index

Notation Index

Chapter 2

Chapter 3

Chapter 4

$\precsim_{\text{P2P}}$	Semantic MUST peer pre-order	90
$\precsim_{\text{SVR}}$	Semantic MUST server pre-order	59
$\mathsf{ua}_{\not\checkmark}\ s$	Usable actions after unsuccesful trace s	70
$\sqsubseteq_{C}$	MUST client pre-order	63
$\sqsubseteq_{S}$	MUST server pre-order	52
$\sqsubseteq_{\text{P2P}}$	MUST peer pre-order	83
ACC	Acceptance set	57

Chapter 5

$\text{ACC}^{\checkmark}$	Acceptance sets with ✓	115
$\mathcal{U}^{\dashv_{\text{P2P}}}$	Usable peers	123
$\downdownarrows_{\text{P2P}}\ s$	Peer compliance convergence after trace s	125
$\downdownarrows\ s$	Convergence after trace s	100
$\preceq_{\text{CLT}}$	Semantic compliance client pre-order	115
$\preceq_{\text{P2P}}$	Semantic compliance peer pre-order	127
$\preceq_{\text{SVR}}$	Semantic compliance server pre-order	102
$\mathsf{usbl}\ s$	Usability after trace s	114
$\sqsubseteq_{\text{CLT}}$	Compliance client pre-order	110
$\sqsubseteq_{\text{SVR}}$	Compliance server pre-order	98
$\sqsubseteq_{\text{P2P}}$	Compliance peer pre-order	122
$\mathsf{ua}^{\checkmark}$	Usable actions and ✓ after trace	115

Chapter 6

$\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$	Restricted MUST client pre-order	140
$\sqsubseteq^{\mathsf{fo}}_{\text{SVR}}$	Restricted MUST server pre-order	135
$\preceq^{syn}_{\text{CLT}}$	Syntactic compliance client pre-order	156
$\precsim^{syn}_{\text{CLT}}$	Syntactic MUST client pre-order	143
$\sqsubseteq^{\mathsf{fo}}_{\text{P2P}}$	Session contract pre-order	145
$\sqsubseteq^{\mathsf{fo}}_{\text{CLT}}$	Restricted compliance client pre-order	152
$\sqsubseteq^{\mathsf{fo}}_{\text{SVR}}$	Restricted compliance server pre-order	150

Chapter 7

$\dashv^{s}_{\mathcal{B}}$	$\mathcal{B}$-syntactic compliance relation	173
$\dashv_{\mathcal{B}}$	$\mathcal{B}$-dependent compliance relation	172
ST_{ho}	Higher-order session types	166
$\bowtie_{\mathcal{B}}$	$\mathcal{B}$-co action relation	171
$\mathcal{B}, \mathcal{R}, \mathcal{T}$	Binary, reflexive, and transitive relations on SC_{HO}	170
SC_{HO}	Language of higher-order session contracts	170

Chapter 8

$\sqsubseteq^{\mathsf{ho}}_{\text{P2P}}$	Session contract pre-order	192
$\sqsubseteq^{\mathcal{B}}_{\text{CLT}}$	$\mathcal{B}$-dependent client pre-order	180
$\preceq^{\mathcal{B}}_{\text{CLT}}$	$\mathcal{B}$-syntactic client pre-order	182
$\preceq^{\mathcal{B}}_{\text{SVR}}$	$\mathcal{B}$-syntactic dependent server pre-order	187
$\sqsubseteq^{\mathcal{B}}_{\text{SVR}}$	$\mathcal{B}$-dependent server pre-order	186
$\sqsubseteq^{\mathsf{ho}}_{\text{CLT}}$	Client pre-order	188
$\sqsubseteq^{\mathsf{ho}}_{\text{SVR}}$	Server pre-order	190
$\textsc{Pre}(\mathsf{SC}^2_{\text{HO}})$	Pre-orders on higher-order session contracts	183

Chapter 9

$\pi\mathsf{SC}$	π-processes with session contracts	198
FN	Free names of a $\pi\mathsf{SC}$ process	198
$\vdash$	Type relation	203