Bisimulations for Communicating Transactions[⋆]
(Extended Abstract)

Vasileios Koutavas, Carlo Spaccasassi[⋆⋆], and Matthew Hennessy

Trinity College Dublin

Abstract. We develop a theory of bisimulations for a simple language containing communicating transactions, obtained by dropping the isolation requirement of standard transactions. Such constructs have emerged as a useful programming abstraction for distributed systems.
In systems with communicating transactions actions are tentative, waiting for certain transactions to commit before they become permanent. Our theory captures this by making bisimulations history-dependent, in that actions performed by transactions need to be recorded. The main requirement on bisimulations is the systems being compared need to match up exactly in the permanent actions but only those.
The resulting theory is fully abstract with respect to a natural contextual equivalence and, as we show in examples, provides an effective verification technique for comparing systems with communicating transactions.

1 Introduction

Communicating transactions, obtained by dropping the isolation requirement of standard transactions, is a novel and powerful programming construct for distributed systems. For example, it can be used to simplify the programming of complex concurrent consensus scenarios, avoiding the use of locks and explicit error handling [16]. Variants of such constructs have been proposed as extensions to programming languages [5,7,11,16] and process calculi [2,1,3]. However, before they can be adopted in mainstream programming, significant research is needed in efficient implementation strategies [6,11,16], programming paradigms [1,7], and viable verification techniques [8]. The last concern is the topic of this paper.

Bisimulations [12] provide an elegant and effective proof technique for proving equivalences between processes in a variety of settings (e.g., [15,14]). They are essentially defender strategies in a game where a challenger attempts to discover a difference in the extensional behaviour of two processes, while the defender tries to refute these attempts [17]. For example, consider the standard CCS processes

$$P_1 = a.(b.\mathbb{0} + c.\mathbb{0}) \qquad\qquad Q_1 = a.b.\mathbb{0} + a.c.\mathbb{0} \tag{1}$$

which can perform the action a followed by either b or c, with a slight difference in when this choice is taken. The challenger chooses Q_1 to perform the action a, with

[⋆] This research was supported by SFI project SFI 06 IN.1 1898.
[⋆⋆] Supported by MSR (MRL 2011-039)

residual $Q'_1 = b.\mathbb{0}$ to which the defender must respond with a matching a action from P_1; the only possibility is for P_1 to perform a with residual $P'_1 = b.\mathbb{0} + c.\mathbb{0}$. But now the challenger chooses P'_1 to perform the c action, to which the defender has no response. The defender loses the game. In fact there is no possible winning strategy for the defender in this game and thus no bisimulation containing the pair (P_1, Q_1). Therefore P_1 and Q_1 are deemed to be behaviourally distinct.

However, the appropriate notion of the bisimulation game for a language with communicating transactions or similar constructs is *a priori* unclear. An objective criterion for a potential bisimulation game is relation with contextual equivalence. If the existence of a winning defender's strategy in a game over two systems implies that the systems are contextually equivalent then this game is a *sound* verification technique. If the absence of such a strategy implies that the two systems are contextually inequivalent then the game is a *complete* technique. The main result of this paper is a weak bisimulation theory for a simple language containing communicating transactions, *TCCS*^{m}, which provides a sound and complete proof methodology with respect to a natural contextual equivalence.

TCCS^m is obtained from CCS [12] by essentially adding one construct $[\![P \rhd_k Q]\!]$ for communicating transactions, and a new command co for committing them. Here k is the name of the transaction, P the body which is expected to be completed in its entirety or not at all, and Q is the alternative, to be executed in case the transaction is aborted. However it should be emphasised that if an abort occurs not only is Q launched but P *and all its effects on the environment* are rolled back. For example consider the systems with p fresh from R and S:

$$\begin{aligned} P_2 &= \nu p.\quad [\![a.p.\mathsf{co}.R \rhd_k \mathbb{0}]\!] \mid [\![b.\overline{p}.\mathsf{co}.S \rhd_l \mathbb{0}]\!] \\ Q_2 &= [\![a.b.\mathsf{co}.(R \mid S) + b.a.\mathsf{co}.(R \mid S) \rhd_m \mathbb{0}]\!] \end{aligned} \tag{2}$$

Here P_2 consists of two independent transactions which co-operate by synchronising on a private channel p. If after this synchronisation the left-hand transaction aborts then the effect of the a action, which is a communication with the environment, must be rolled back. But the synchronisation on p must also be undone, and therefore, because of the all-or-nothing nature of transactions, the effects of the b action in the right-hand transaction will be rolled back. Indeed in our reduction semantics, given in Sect. 2, this right-hand transaction is also aborted. Because of the synchronisation on p, the destiny of both transactions is conjoined. For this reason we should be able to demonstrate that P_2 is behaviourally equivalent to the single transaction Q_2.

The standard bisimulation game outlined above cannot be easily extended to *TCCS*^{m}. Many actions, such as a, b in (2) above, are tentative, in the sense that their effect can only be considered to be permanent when the transactions k and l commit, if ever. To underline this consider the processes:

$$P_3 = [\![a.(b.\mathsf{co} + c.\mathbb{0}) \rhd_k \mathbb{0}]\!] \qquad\qquad Q_3 = [\![a.b.\mathsf{co} \rhd_l \mathbb{0}]\!] \tag{3}$$

Here P_3 commits only after performing the a, b. It would be unreasonable for the challenger, after the a action, to demand a response to c. Not only is this action tentative on the completion of transaction k but also if c is performed then k will *never* commit; so the defender should be able to ignore this challenge.

Our approach is to play the bisimulation game on *configurations*, of the form $\mathcal{C} = (H \rhd P)$ where P is a system and H a *history* of all the tentative actions taken so far in the game by P. These are of the form $k(a)$, where k is the name of a transaction which needs to commit before the action becomes a permanent a. When playing bisimulation moves, the histories of both systems being scrutinised must remain *consistent*, in that permanent actions in the respective histories must match exactly (for simplicity old permanent actions are not garbage collected). The crucial aspect of this new game is that when a system commits a transaction k, and only then, all tentative actions in its history dependent on k are made permanent. This consistency requirement then forces a response in which the corresponding actions match exactly.

For example consider the following variation on (1), using transactions:

$$P_4 = [\![a.(b.\mathtt{co} \ + \ c.\mathtt{co}) \rhd_l \mathbb{0}]\!] \qquad\qquad Q_4 = [\![a.b.\mathtt{co} \ + \ a.c.\mathtt{co} \rhd_k \mathbb{0}]\!] \tag{4}$$

Replaying the game from (1), where the challenger first chooses a from Q_4 and then c from P_4 with the same responses, we reach the configurations

$$\mathcal{C}_4 = (k(a), k(c) \rhd [\![\mathtt{co} \rhd_k \mathbb{0}]\!]) \qquad\qquad \mathcal{D}_4 = (l(a), l(b) \rhd [\![\mathtt{co} \rhd_l \mathbb{0}]\!])$$

At this stage the two histories are still consistent as they contain no permanent actions. However, now there is no possible response when the challenger chooses the commit move $\mathcal{C}_4 \to \mathcal{C}'_4 = (a, c \rhd \mathbb{0})$. This is a silent move from $\mathcal{C}_4$ in which transaction k commits, making the two actions in the history permanent. There are various ways in which $\mathcal{D}_4$ can try to respond but all lead to an inconsistent history. Thus, with our version of bisimulations P_4 and Q_4 are not bisimilar.

Note that such a successful attack by the challenger cannot be mounted for (3) above. After one round in the game we have the configurations

$$\mathcal{C}_3 = (k(a) \rhd [\![b.\mathtt{co} + c.\mathbb{0} \rhd_k \mathbb{0}]\!]) \qquad\qquad \mathcal{D}_3 = (l(a) \rhd [\![b.\mathtt{co} \rhd_l \mathbb{0}]\!])$$

and since $\mathcal{D}_3$ has no possible c actions the challenger might request a response to the action $\mathcal{C}_3 \to \mathcal{C}'_3 = (k(a), k(c) \rhd [\![\mathbb{0} \rhd_k \mathbb{0}]\!])$. However the tentative $k(c)$ recorded in the history will never become permanent and thus the defender can successfully respond with any tentative action of $\mathcal{D}_3$ which will also never become permanent. To ensure that such responses can always be made, our bisimulations will allow *any* configuration to make the degenerate silent move $(H \rhd P) \to (H, k(\star) \rhd P)$, where $\star$ is a reserved symbol. This defender move represents a weak idle move whose validity is postponed until after k commits. So $\mathcal{C}_3$'s move above can be matched by $\mathcal{D}_3$ playing this degenerate move followed by the abort of the transaction. Later we prove that P_3 and Q_3 are indeed bisimilar.

Using recursion we can write *restarting transactions* as $\mathtt{rec}X.[\![P \blacktriangleright X]\!]$. Here $[\![P \blacktriangleright X]\!]$ is an uninitiated transaction which has yet to be allocated a name. These transactions can abort and re-run internal steps, thus branching differences in initial silent actions can be hidden from the challenger. Consider a compiler performing common subexpression elimination, transforming P_5 to Q_5:

$$P_5 = \mathtt{rec}X.[\![\tau.b.\mathsf{co} + \tau.c.\mathsf{co} \blacktriangleright X]\!] \qquad Q_5 = \mathtt{rec}X.[\![\tau.(b.\mathsf{co} + c.\mathsf{co}) \blacktriangleright X]\!] \qquad (5)$$

As we will show, all moves of the challenger can be matched by the defender in the bisimulation game. The interesting scenario is when (after initiating the transactions) the challenger picks the right τ action from P_5 and the defender responds with the τ action from Q_5. We then get the configurations:

$$\mathcal{C}_5 = (\varepsilon \rhd [\![c.\mathsf{co} \rhd_k P_5]\!]) \qquad \mathcal{D}_5 = (\varepsilon \rhd [\![b.\mathsf{co} + c.\mathsf{co} \rhd_l Q_5]\!])$$

The challenger then picks the b action from $\mathcal{D}_5$. The defender responds with a silent abort of $\mathcal{C}_5$ which will reinstate P_5, re-initialise the transaction, and select the left τ action in $\mathcal{P}_5$ followed by the b action. This would lead to the configurations $\mathcal{C}'_5 = (k'(b) \rhd [\![\mathsf{co} \rhd_{k'} P_5]\!])$ and $\mathcal{D}'_5 = (l(b) \rhd [\![\mathsf{co} \rhd_l Q_5]\!])$. We will in fact prove that this optimisation is sound in our setting, although it is not sound in the case of P_4, Q_4, even if we used restarting transactions.

In the remainder of this extended abstract we explain the language *TCCS*^{m} (Sect. 2), which is a simplification of that used in previous work [4] in that we do not consider nested transactions, simplifying technical development. Inspired by cJoin [1], when transactions co-operate by synchronising on an action they are virtually *merged* by acquiring the same name. Thus in P_2 from (2), when synchronisation occurs on the private channel p, the residual will be a term equivalent to the single transaction $\nu p.\,[\![\mathsf{co}.(R \,|\, S) \rhd_n \mathbb{0} \,|\, \mathbb{0}]\!]$.

This is followed by an exposition of our history dependent bisimulations (Sect. 3). As we have already stated, these bisimulations demand the appropriate matching of all actions, even those dependent on transactions which can never commit. We also give a variation, called *predictive bisimulations*, in which dependent actions need only be matched when the transaction on which they depend has some future possibility of committing (Sect. 4). We ultimately show that both equivalences coincide but the former is an easier proof technique while the latter is easier to prove sound. We then outline the proof of the main result of the paper, namely that these bisimulation equivalences coincide with contextual equivalence (Sect. 5).

2 The language *TCCS*^{m}

The syntax for terms in the language is given in Fig. 1 where $a \in \mathsf{Act}$ are actions, $\mu \in \mathsf{Act} \uplus \{\tau\} \uplus \Omega$ are prefixes, and X ranges over a collection of *recursion variables*. Here $\omega \in \Omega$ are special actions which will be used to define contextual equivalence, while $(\bar{\cdot}) : \mathsf{Act} \to \mathsf{Act}$ is a total bijection over Act, used in the standard manner to formalise CCS synchronisation between processes. The language contains all the standard constructs of CCS, with the result that CCS is a sub-language of *TCCS*^{m}. There are three extra operators, discussed in the Introduction. We assume the standard notion of free and bound occurrence of recursion variables, and only consider closed terms, those which contain

$TCCS^m$ Syntax

$$P, Q, R \quad ::= \quad \sum \mu_i.P_i \;\mid\; P \,|\, Q \;\mid\; \nu a.P \;\mid\; X \;\mid\; \mathtt{rec}X.P$$
$$\mid\; \llbracket P \rhd_k Q \rrbracket \;\mid\; \mathtt{co}.P \;\mid\; \llbracket P \blacktriangleright Q \rrbracket$$

CCS Transitions

$$\textsc{CCSsum}\quad \frac{}{\Sigma\mu_i.P_i \xrightarrow{\mu_i}_{\varepsilon} P_i} \qquad \textsc{CCSsync}\quad \frac{P \xrightarrow{a}_{\varepsilon} P' \qquad Q \xrightarrow{\overline{a}}_{\varepsilon} Q'}{P \,|\, Q \xrightarrow{\tau}_{\varepsilon} P' \,|\, Q'} \qquad \textsc{CCSRec}\quad \frac{}{\mu X.P \xrightarrow{\tau}_{\varepsilon} P[\mu X.P/X]}$$

Transactional Transitions

$$\textsc{TrTau}\quad \frac{P \xrightarrow{\tau}_{\varepsilon} P'}{\llbracket P \rhd_k Q \rrbracket \xrightarrow{\tau}_{\varepsilon} \llbracket P' \rhd_k Q \rrbracket} \qquad \textsc{TrSum}\quad \frac{}{\Sigma\mu_i.P_i \xrightarrow{k(a)}_{\varepsilon\mapsto k} \llbracket P_j \,|\, \mathtt{co} \rhd_k \Sigma\mu_i.P_i \rrbracket}\,\mu_j = a$$

$$\textsc{TrAct}\quad \frac{P \xrightarrow{a}_{\varepsilon} P'}{\llbracket P \rhd_l Q \rrbracket \xrightarrow{k(a)}_{l\mapsto k} \llbracket P' \rhd_k Q \rrbracket}\,k \sharp l \qquad \textsc{TrSync}\quad \frac{P \xrightarrow{k(a)}_{\sigma_1} P' \qquad Q \xrightarrow{k(\overline{a})}_{\sigma_2} Q'}{P \,|\, Q \xrightarrow{k(\tau)}_{(\widetilde{l_1},\widetilde{l_2})\mapsto k} P'\sigma_2 \,|\, Q'\sigma_1}\,\begin{array}{l}\sigma_1 = \widetilde{l_1} \mapsto k\\ \sigma_2 = \widetilde{l_2} \mapsto k\end{array}$$

Propagation Transitions

$$\textsc{Restr}\quad \frac{P \xrightarrow{\alpha}_{\sigma} P'}{\nu a.P \xrightarrow{\alpha}_{\sigma} \nu a.P'}\,a \notin \alpha \qquad \textsc{ParL}\quad \frac{P \xrightarrow{\alpha}_{\sigma} P'}{P \,|\, Q \xrightarrow{\alpha}_{\sigma} P' \,|\, Q\sigma}\,\mathit{range}(\sigma) \sharp Q$$

Fig. 1. Communication and internal transitions (omitting symmetric rules)

no free occurrences of variables. We use the standard abbreviations associated with CCS, and write $s \sharp s'$ when the transaction names of the syntax object s are fresh from those in s'; $\mathit{ftn}(s)$ denotes the transaction names in s. Note that unlike previous work [3,4] transaction names are never bound and we do not require that all transaction names used in a term are distinct. Thus, we allow terms of the form $\llbracket P_1 \rhd_k P_2 \rrbracket \,|\, R \,|\, \llbracket Q_1 \rhd_k Q_2 \rrbracket$. Here k should be looked upon as a *distributed* transaction whose behaviour will be approximately the same as the centralised $\llbracket P_1 \,|\, Q_1 \rhd_k P_2 \,|\, Q_2 \rrbracket$. The use of these distributed transactions will simplify considerably the exposition of the reduction semantics.

Definition 2.1. *A closed term is called* well-formed *if in every occurrence of $\llbracket P \rhd_k Q \rrbracket$, $\llbracket P \blacktriangleright Q \rrbracket$, and $\mathtt{rec}X.P$, the subterms P and Q do not contain named transactions of the form $\llbracket - \rhd_{\_} - \rrbracket$. We refer to well-formed terms as* processes.

Note that dormant transactions *can* appear within other transactions and under recursion but they will be activated only when they end up at top-level. In the sequel we only consider well-formed terms.

The reduction semantics of the language is given as a binary relation between processes $P \to Q$. However this is defined indirectly in terms of three auxiliary relations, which will also be used in the formulation of bisimulations:

$$P \to Q \qquad \text{when} \qquad P \xrightarrow{\tau}_{\sigma} Q \qquad \text{or} \qquad P \xrightarrow{\beta} Q \qquad \text{or} \qquad P \xrightarrow{k(\tau)}_{\sigma} Q \qquad (6)$$

TrNew
$$\frac{}{[\![P \blacktriangleright Q]\!] \xrightarrow{\mathtt{new}\,k} [\![P \rhd_k Q]\!]}$$

TrAb
$$\frac{}{[\![P \rhd_k Q]\!] \xrightarrow{\mathtt{ab}k} Q}$$

TrCo
$$\frac{P \rightsquigarrow_{\mathtt{co}} P'}{[\![P \rhd_k Q]\!] \xrightarrow{\mathtt{co}k} P'}$$

TrBCast
$$\frac{P \xrightarrow{\beta} P' \qquad Q \xrightarrow{\beta} Q'}{P \,|\, Q \xrightarrow{\beta} P' \,|\, Q'}\ \beta \in \{\mathtt{co}k, \mathtt{ab}k\}$$

TrIgnore
$$\frac{P \xrightarrow{\beta} P'}{P \,|\, Q \xrightarrow{\beta} P' \,|\, Q}\ \beta \sharp Q$$

TrRestr
$$\frac{P \xrightarrow{\beta} P'}{\nu a.P \xrightarrow{\beta} \nu a.P'}$$

Fig. 2. Transactional reconfiguration transitions

The first, $P \xrightarrow{\tau}_{\sigma} Q$, is essentially synchronisation between pure CCS processes. The second, $P \xrightarrow{\beta} Q$ (Fig. 2), where β ranges over $\mathtt{co}\ k$, $\mathtt{ab}\ k$ and $\mathtt{new}\ k$, encode the creation of new named transactions, and commit/abort broadcast transitions (TrCo, TrAb, TrBCast) which eliminate distributed transactions. The notation $P \rightsquigarrow_{\mathtt{co}} P'$ means the execution of a top-level $\mathtt{co}$ in P and the replacement of all other top-level commits with τ-prefixes. We now concentrate on the third, $P \xrightarrow{k(\tau)}_{\sigma} Q$, or more generally $P \xrightarrow{k(\mu)}_{\sigma} Q$ where $\mu \in \mathsf{Act} \cup \{\tau\}$.

Action $P \xrightarrow{k(a)}_{\sigma} Q$ should be viewed as the synchronisation between P and some transaction named k in the environment which can perform the complementary $\overline{a}$. Because this transaction is external the freshness side conditions in the rules of Fig. 1 ask that the name k is fresh with respect to P. Also, the effect of this synchronisation is that the future behaviour of P, or at least any transactions involved in the execution of a, is dependent on the eventual committing of k. This dependency is implemented by σ, a substitution renaming the responsible transaction in P to k. The essential rule in the generation of these judgements is TrAct in Fig. 1. For example, this rule ensures that we can derive $[\![a.P_1 \rhd_{l_1} Q_1]\!] \xrightarrow{k(a)}_{l_1 \mapsto k} [\![P_1 \rhd_k Q_1]\!]$ for any fresh k. The substitution recorded in the action is propagated by ParL into contexts. Note that by TrSum, even pure CCS processes with no transactions can perform a $k(a)$ action; e.g., $a.P \xrightarrow{k(a)}_{\varepsilon \mapsto k} [\![P \,|\, \mathtt{co} \rhd_k a.P]\!]$. This embeds P into the k-transaction; the distributed part of the k-transaction surrounding P is always ready to commit (hence the introduction of $\mathtt{co}$). Note that this is a *communication-driven embedding*, which reduces the nondeterminism of embedding of previous work [3,4], making semantics more concrete [16]. Embedding leads to a uniform treatment of CCS processes and transactions, and a simple reduction semantics.

The conjoining of transactions is implemented in TrSync. Using it we infer:

$$[\![a.P_1 \rhd_{l_1} Q_1]\!] \;|\; [\![\overline{a}.P_2 \rhd_{l_2} Q_2]\!] \quad \xrightarrow{k(\tau)}_{(l_1,l_2)\mapsto k} \quad [\![P_1 \rhd_k Q_1]\!] \;|\; [\![P_2 \rhd_k Q_2]\!]$$

for any fresh k. Here the previously independent transactions l_1, l_2 have been merged into the transaction k (recorded in the substitution $(l_1, l_2) \mapsto k$). Note that this new transaction is distributed, in that its activity is divided in two. In order for it to commit the rules in Fig. 2 ensure that *both* components commit.

Example 2.2. Consider the process P_2 defined in (2) in the Introduction. Two applications of TrAct followed by rule Restr gives the reduction from (6) above

$$P_2 \,|\, \overline{a}.\overline{b} \rightarrow^* \nu p.\, [\![p.\mathsf{co}.R \rhd_{k_1} \mathbb{0}]\!] \,|\, [\![\overline{p}.\mathsf{co}.S \rhd_{k_2} \mathbb{0}]\!] = P_2'$$

where k_1 and k_2 are fresh names. The synchronisation rule TrSync then gives

$$P_2' \rightarrow \nu p.\, [\![\mathsf{co}.R \rhd_{k} \mathbb{0}]\!] \,|\, [\![\mathsf{co}.S \rhd_{k} \mathbb{0}]\!]$$

where k is an arbitrary fresh name. Here the residual is a single transaction named k, albeit distributed. For it to commit both components have to commit: using TrBCast this leads to the process $R \,|\, S$. □

The semantics has a number of properties: it preserves well-formedness, generates only fresh transaction names and is equivariant. The properties about transaction names are important because they give us the liberty to pick fresh enough transaction names in proofs. To state these properties we use *renamings*, ranged over by r, which are bijective substitutions of the form $\{l_1/k_1, \ldots, l_n/k_n\}$. The range of a *fresh* renaming r_{fr} has names not appearing in the proof.

Lemma 2.3 (Names). *Suppose* $P \xrightarrow{l(\mu)}_{\sigma} Q$. *Then*

1. l *is fresh to* P, Q *is well-formed, and the substitution* σ *has the form* $\widetilde{k} \mapsto l$;
2. *(Equivariance)* $Pr_{\mathsf{fr}} \xrightarrow{l'(\mu)}_{\sigma r_{\mathsf{fr}}} Qr_{\mathsf{fr}}$, *where* $r_{\mathsf{fr}}(l) = l'$. □

Based on this semantics we give a natural contextual equivalence, using standard formulations [15]. We write $\Rightarrow$ for the reflexive transitive closure of $\rightarrow$.

Definition 2.4 (Barb). $P{\Downarrow}\omega$ $(\omega \in \Omega)$ *if* $\exists\, Q, Q'$ *such that* $P \Rightarrow Q \xrightarrow{\omega}_{\epsilon} Q'$.

Definition 2.5 (Reduction Barbed Equivalence ($\cong_{\mathsf{rbe}}$)). ($\cong_{\mathsf{rbe}}$) *is the largest relation for which* $P \cong_{\mathsf{rbe}} Q$ *when:*

1. $P{\Downarrow}\omega$ *iff* $Q{\Downarrow}\omega$,
2. *if* $P \rightarrow P'$ *then there exists* Q' *such that* $Q \Rightarrow Q'$ *and* $P' \cong_{\mathsf{rbe}} Q'$,
3. *if* $Q \rightarrow Q'$ *then there exists* P' *such that* $P \Rightarrow P'$ *and* $P' \cong_{\mathsf{rbe}} Q'$,
4. $P \,|\, R \cong_{\mathsf{rbe}} Q \,|\, R$ *for any* R *with* $R \sharp P, Q$.

Here we consider contexts with fresh transaction names to enforce that observer transactions are distinct from process transactions before communication occurs. If this was not the case then transaction names would be observable: $[\![P \rhd_k Q]\!]$ would not be equivalent to $[\![P \rhd_l Q]\!]$ because by introducing the context $[\![\mathbb{0} \rhd_k \mathbb{0}]\!]$ the k-transaction can no longer commit but l still can. Thus, *all* transaction names are considered local here; the side condition $R \sharp P, Q$ enforces this without the syntactic overhead of a ν-binder for all transaction names.

To see why in the above definition we use barbs from a distinct Ω consider:

$$P = [\![a.\mathsf{co} \rhd_k \mathbb{0}]\!] \qquad\qquad Q = a.\mathbb{0} + \tau.\mathbb{0}$$

Intuitively, we would expect P to have exactly the same behaviour as Q, eventually executing the single action a, or failing with a τ step. But if we allowed the barb ${\Downarrow}a$ in Def. 2.5 then they would not be equivalent because $P{\not\Downarrow}a$ and $Q{\Downarrow}a$.

Example 2.6. P_4 and Q_4 from (4) in the Introduction are indeed inequivalent.
Assume $P_4 \cong_{\mathsf{rbe}} Q_4$. Take $C_1 = \bar{a}$. Then $P_4 \mid C_1 \cong_{\mathsf{rbe}} Q_4 \mid C_1$. We have:

$$Q_4 \mid C_1 \xrightarrow{l(\tau)} [\![b.\mathsf{co} \rhd_l \mathbb{0}]\!] \mid [\![\mathsf{co} \rhd_l C_1]\!] = Q_4' \qquad \text{thus} \qquad Q_4 \mid C_1 \to Q_4'$$

Process Q_4' should be equivalent to $P_4 \mid C_1$ or one of its successors:

1. $P_4 \mid C_1$. Let $C_2 = \bar{c}.\omega$; then $Q_4' \mid C_2 \not\Downarrow \omega$, $P_4 \mid C_1 \mid C_2 \Downarrow \omega$. Thus $P_4 \mid C_1 \not\cong_{\mathsf{rbe}} Q_4'$.
2. $P_4' = [\![b.\mathsf{co} + c.\mathsf{co} \rhd_m \mathbb{0}]\!] \mid [\![\mathbb{0} \rhd_m C_1]\!]$. Again, $P_4' \mid C_2 \Downarrow \omega$, thus $P_4' \not\cong_{\mathsf{rbe}} Q_4'$.
3. C_1 (after an abort). $Q_4' \mid \bar{b}.\omega \Downarrow \omega$ but $C_1 \mid \bar{b}.\omega \not\Downarrow \omega$, so $C_1 \not\cong_{\mathsf{rbe}} Q_4'$.

Thus $Q_4' \not\cong_{\mathsf{rbe}} P_4 \mid C_1$ or any later state (contradiction), and $P_4 \not\cong_{\mathsf{rbe}} Q_4$. □

Note that the difference in the branching structure of P_4 and Q_4 is not observable by the may- and must-testing equivalences [3,4]. These equivalences are characterised by so-called *clean traces*, which are traces in which all tentative actions are committed. If bisimulations were developed using such clean traces, P_4 and Q_4 would also be identified in the resulting bisimulation theory but it would not correspond to a natural definition of $(\cong_{\mathsf{rbe}})$.

3 Bisimulations

As mentioned in the Introduction, our bisimulations will be over configurations $(H \rhd P)$ with a process P and a history H of the tentative interactions of P with its environment. An element of such a history can be a $k(a)$, a, ab, $k(\star)$, or $\star$. A past tentative action $k(a)$ that has not been committed or aborted is recorded as is in the history. If the k-transaction that performed this action commits, the action becomes a; if k aborts, it becomes ab. Histories also record the trivial actions $k(\star)$ which can be performed by any process. If k commits, $k(\star)$-recordings become $\star$, which terminates the bisimulation game in favour of the attacker; if k aborts they become ab. For technical convenience in proofs, elements in a history are uniquely indexed and permanent actions are not garbage-collected.

Definition 3.1 (History). *A* history H *is a partial function from objects i of a countable set I to the set* $\{a, \star, k(a), k(\star), \mathsf{ab} \mid a \in \mathit{Act}\}$.

We often write histories as *lists*, omitting the indices of their elements. History composition, written as H_1, H_2, is defined when $dom(H_1) \cap dom(H_2) = \emptyset$. We also let $\hat{a}$ and $\hat{b}$ range over $\mathsf{Act} \cup \{\star\}$ and $\hat{\mu}$ range over $\mathsf{Act} \cup \Omega \cup \{\tau, \star\}$. To express the effect of commits and aborts to histories we define the following operations.

Definition 3.2. *$H \setminus_{\mathsf{co}} k$ and $H \setminus_{\mathsf{ab}} k$ are the lifting to lists of the operations:*

$$\begin{array}{lll}
(i \mapsto k(\hat{a})) \setminus_{\mathsf{co}} k = (i \mapsto \hat{a}) & (i \mapsto k(\hat{a})) \setminus_{\mathsf{ab}} k = (i \mapsto \mathsf{ab}) & \\
(i \mapsto l(\hat{a})) \setminus_{\mathsf{co}} k \ = (i \mapsto l(\hat{a})) & (i \mapsto l(\hat{a})) \setminus_{\mathsf{ab}} k \ = (i \mapsto l(\hat{a})) & \text{when } k \sharp l \\
(i \mapsto \hat{a}) \setminus_{\mathsf{co}} k \ \ = (i \mapsto \hat{a}) & (i \mapsto \hat{a}) \setminus_{\mathsf{ab}} k \ \ = (i \mapsto \hat{a}) & \\
(i \mapsto \mathsf{ab}) \setminus_{\mathsf{co}} k \ \ = (i \mapsto \mathsf{ab}) & (i \mapsto \mathsf{ab}) \setminus_{\mathsf{ab}} k \ \ = (i \mapsto \mathsf{ab}) &
\end{array}$$

For the reasons we explained in the Introduction, weak bisimulations for $TCCS^m$ require configurations to agree on the committed actions in their histories, and only those actions. Soundness of our technique will establish this as a sufficient requirement for contextual equivalence between processes.

Definition 3.3 (Consistency). *H_1 and H_2 are* consistent *when they have the same domain and for all $i \in I$, $a \in \mathsf{Act}$: $H_1(i) = a$ iff $H_2(i) = a$.*

History consistency is one of the two main requirements for weakly bisimilar configurations; the other is to have the same barbs. Thus the weak bisimulation game for $TCCS^m$ will be over transitions with three simple labels: $\zeta ::= \tau \mid k \mid \omega$ annotating internal (τ), tentative synchronisation (k), and barb (ω) transitions.

Definition 3.4 (Bisimulation Transitions). *$\mathcal{C} \xrightarrow{\zeta} \mathcal{C}'$ is derived by the rules:*

$$\begin{array}{lll}
(H \rhd P) \xrightarrow{\tau} (\sigma(H) \rhd Q) & \textit{if } P \xrightarrow{\tau}_{\sigma} Q & (\text{LTS}\tau) \\
(H \rhd P) \xrightarrow{\tau} (\sigma(H) \rhd Q) & \textit{if } P \xrightarrow{k(\tau)}_{\sigma} Q \textit{ and } k \sharp H & (\text{LTS}k(\tau)) \\
(H \rhd P) \xrightarrow{\tau} (H \rhd Q) & \textit{if } P \xrightarrow{\mathtt{new}\,k} Q \textit{ and } k \sharp H & (\text{LTS}\mathtt{new}) \\
(H \rhd P) \xrightarrow{\tau} (H \setminus_{\mathtt{co}} k \rhd Q) & \textit{if } P \xrightarrow{\mathtt{co}\,k} Q & (\text{LTS}\mathtt{co}) \\
(H \rhd P) \xrightarrow{\tau} (H \setminus_{\mathtt{ab}} k \rhd Q) & \textit{if } P \xrightarrow{\mathtt{ab}\,k} Q & (\text{LTS}\mathtt{ab}) \\
(H \rhd P) \xrightarrow{k} (\sigma(H), k(a) \rhd Q) & \textit{if } P \xrightarrow{k(a)}_{\sigma} Q \textit{ and } k \sharp H & (\text{LTS}k(a)) \\
(H \rhd P) \xrightarrow{k} (H, k(\star) \rhd P) & \textit{if } k \sharp H, P & (\text{LTS}\star) \\
(H \rhd P) \xrightarrow{\omega} (\sigma(H) \rhd Q) & \textit{if } P \xrightarrow{\omega}_{\sigma} Q & (\text{LTS}\omega)
\end{array}$$

We define $\overset{\zeta}{\Rightarrow}$ to be $\xrightarrow{\tau}^*$ when $\zeta = \tau$, and $\overset{\tau}{\Rightarrow}\xrightarrow{\zeta}\overset{\tau}{\Rightarrow}$ otherwise.

The first five rules encode the $TCCS^m$ reduction semantics of (6) in Sect. 2, updating the history of the configurations accordingly. LTS$k(a)$ encodes the synchronisation between a transaction in the process and its environment, yielding a fresh transaction k; this tentative action is recorded in the history. LTSω encodes top-level barbs and LTS$\star$ records a trivial defender synchronisation move. Weak τ- and k-transitions can always be performed by the defender in the bisimulation game. Moreover, there are no top-level a-transitions because they can always be simulated by a $k(a)$-transition followed by the commit of k.

Lemma 3.5. *Suppose $P \xrightarrow{a}_{\varepsilon} Q$; then $P \xrightarrow{k(a)}_{\varepsilon \mapsto k} P' \xrightarrow{\mathtt{co}k} Q$ and $(H \rhd P) \xrightarrow{k} (H, k(a) \rhd P') \xrightarrow{\tau} (H, a \rhd Q)$ for some P', any H, and any $k \sharp P, H$.* □

To keep histories and the bisimulation game finite in examples, the challenger of this bisimulation game performs all-but-$\star$ transitions.

Definition 3.6. *$\mathcal{C}_1 \xrightarrow{\zeta} \mathcal{C}_2$ is a* challenger move *if it is derived without using* LTS$\star$.

We now give the definition for a weak bisimulation over the above transitions.

Definition 3.7 (Weak Bisimulation). *A binary relation $\mathcal{R}$ over configurations is a weak bisimulation when for all $\mathcal{C}_1 \mathrel{\mathcal{R}} \mathcal{C}_2$:*

$$
\begin{aligned}
\mathcal{R}_3 \overset{\text{def}}{=} \{\; & ((\varepsilon \triangleright P_3),\ (\varepsilon \triangleright Q_3)), \quad ((k(a) \triangleright \llbracket b.\mathtt{co} + c.\mathbb{0} \triangleright_k \mathbb{0} \rrbracket),\ (k(a) \triangleright \llbracket b.\mathtt{co} \triangleright_k \mathbb{0} \rrbracket)),\\
& ((k(a), k(b) \triangleright \llbracket \mathtt{co} \triangleright_k \mathbb{0} \rrbracket),\ (k(a), k(b) \triangleright \llbracket \mathtt{co} \triangleright_k \mathbb{0} \rrbracket)), \quad ((a, b \triangleright \mathbb{0}),\ (a, b \triangleright \mathbb{0})),\\
& ((k(a), k(c) \triangleright \llbracket \mathbb{0} \triangleright_k \mathbb{0} \rrbracket),\ (\mathtt{ab}, \mathtt{ab} \triangleright \mathbb{0})), \quad ((\mathtt{ab}, \dots \triangleright \mathbb{0}),\ (\mathtt{ab}, \dots \triangleright \mathbb{0})) \ \mid\ \text{any } k\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{R}_2 \overset{\text{def}}{=} \{\; & ((\varepsilon \triangleright P_2),\ (\varepsilon \triangleright Q_2)), \quad ((\mathtt{ab}, \dots \triangleright \mathbb{0}),\ (\mathtt{ab}, \dots \triangleright \mathbb{0})),\\
& (k_1(a) \triangleright \nu p.\ \llbracket p.\mathtt{co}.R \triangleright_{k_1} \mathbb{0} \rrbracket \mid \llbracket b.p.\mathtt{co}.S \triangleright_{k_2} \mathbb{0} \rrbracket),\ (k_1(a) \triangleright \llbracket b.\mathtt{co}.(R \mid S) \triangleright_{k_1} \mathbb{0} \rrbracket)),\\
& (k_2(b) \triangleright \nu p.\ \llbracket a.p.\mathtt{co}.R \triangleright_{k_1} \mathbb{0} \rrbracket \mid \llbracket p.\mathtt{co}.S \triangleright_{k_2} \mathbb{0} \rrbracket),\ (k_2(b) \triangleright \llbracket a.\mathtt{co}.(R \mid S) \triangleright_{k_2} \mathbb{0} \rrbracket)),\\
& ((\boldsymbol{k_1}(a), \boldsymbol{k_2}(b) \triangleright \nu p.\ \llbracket p.\mathtt{co}.R \triangleright_{k_1} \mathbb{0} \rrbracket \mid \llbracket p.\mathtt{co}.S \triangleright_{k_2} \mathbb{0} \rrbracket),\\
& \quad (k_2(a), k_2(b) \triangleright \llbracket \mathtt{co}.(R \mid S) \triangleright_{k_2} \mathbb{0} \rrbracket)),\\
& ((\boldsymbol{k_2}(b), \boldsymbol{k_1}(a) \triangleright \nu p.\ \llbracket p.\mathtt{co}.R \triangleright_{k_1} \mathbb{0} \rrbracket \mid \llbracket p.\mathtt{co}.S \triangleright_{k_2} \mathbb{0} \rrbracket),\\
& \quad (k_1(b), k_1(a) \triangleright \llbracket \mathtt{co}.(R \mid S) \triangleright_{k_1} \mathbb{0} \rrbracket)),\\
& ((\boldsymbol{k}(x), \boldsymbol{k}(y) \triangleright \nu p.\ \llbracket \mathtt{co}.R \triangleright_k \mathbb{0} \rrbracket \mid \llbracket \mathtt{co}.S \triangleright_k \mathbb{0} \rrbracket), (k(x), k(y) \triangleright \llbracket \mathtt{co}.(R \mid S) \triangleright_k \mathbb{0} \rrbracket))\\
& ((x, y, H \triangleright \nu p.\ R \mid S), (x, y, H \triangleright R \mid S))\\
& \ \mid\ \text{any } k, k_1, k_2, R, S, H \text{ and } (x, y) = (a, b) \text{ or } (b, a)\ \}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{R}_5 \overset{\text{def}}{=} \{\; & ((H \triangleright P_5),\ (H \triangleright Q_5)), \quad ((H, x \triangleright \mathbb{0}),\ (H, x \triangleright \mathbb{0}))\\
& ((H \triangleright \llbracket \tau.b.\mathtt{co} + \tau.c.\mathtt{co} \blacktriangleright P_5 \rrbracket),\ (H \triangleright \llbracket \tau.(b.\mathtt{co} + c.\mathtt{co}) \blacktriangleright Q_5 \rrbracket)),\\
& ((H \triangleright \llbracket \tau.b.\mathtt{co} + \tau.c.\mathtt{co} \triangleright_k P_5 \rrbracket),\ (H \triangleright \llbracket \tau.(b.\mathtt{co} + c.\mathtt{co}) \triangleright_l Q_5 \rrbracket)),\\
& ((H \triangleright \llbracket b.\mathtt{co} \triangleright_k P_5 \rrbracket),\ (H \triangleright \llbracket (b.\mathtt{co} + c.\mathtt{co}) \triangleright_l Q_5 \rrbracket)),\\
& ((H \triangleright \llbracket c.\mathtt{co} \triangleright_k P_5 \rrbracket),\ (H \triangleright \llbracket (b.\mathtt{co} + c.\mathtt{co}) \triangleright_l Q_5 \rrbracket)),\\
& ((H, k(x) \triangleright \llbracket \mathtt{co} \triangleright_k P_5 \rrbracket),\ (H, k(x) \triangleright \llbracket \mathtt{co} \triangleright_k Q_5 \rrbracket)),\\
& \mid \text{any } k, l, H = (\mathtt{ab}, \dots), \text{ and } x = a \text{ or } b\ \}
\end{aligned}
$$

Fig. 3. Relations used to prove the equivalences in Ex(s). 3.10 to 3.12.

1. *$hist(\mathcal{C}_1)$ and $hist(\mathcal{C}_2)$ are consistent,*
2. *if $\mathcal{C}_1 \xrightarrow{\zeta} \mathcal{C}_1'$ is a challenger move and $\zeta \sharp \mathcal{C}_2$ then $\exists\, \mathcal{C}_2'$: $\mathcal{C}_2 \overset{\zeta}{\Rightarrow} \mathcal{C}_2'$ and $\mathcal{C}_1' \mathrel{\mathcal{R}} \mathcal{C}_2'$,*
3. *the converse of the preceding condition.*

Condition $\zeta \sharp \mathcal{C}_2$ guarantees that the choice of fresh transaction names in ζ does not hinder the transition from $\mathcal{C}_2$. Weak bisimilarity ($\approx$) is the largest weak bisimulation, and extends to processes $P \approx Q$ if $(\emptyset \triangleright P) \approx (\emptyset \triangleright Q)$. Bisimulation transitions and weak bisimulations are unaffected by fresh renaming. Thus, the name selected in a challenger move is unimportant.

Lemma 3.8 (ζ-Equivariance). *If $\mathcal{C} \xrightarrow{\zeta} \mathcal{C}'$ then $\mathcal{C} r_{\mathsf{fr}} \xrightarrow{\zeta r_{\mathsf{fr}}} \mathcal{C}' r_{\mathsf{fr}}$.* □

Lemma 3.9 (Equivariance of ($\approx$)). *If $\mathcal{C} \approx \mathcal{D}$ then $\mathcal{C} \approx \mathcal{D} r$.* □

We close this section by showing the equivalence of the processes in the introduction by proving them weakly bisimilar. The soundness of our bisimulation technique establishes a proof of contextual equivalence between these processes.

Example 3.10. Recall P_3 and Q_3 from (3) in the Introduction. We show that $P_3 \approx Q_3$; i.e., $(\emptyset \triangleright \mathcal{P}_3) \approx (\emptyset \triangleright \mathcal{Q}_3)$. It suffices to check that $\mathcal{R}_3$ in Fig. 3 is a weak bisimulation. Related histories in $\mathcal{R}_3$ are consistent. The interesting case is when $(k(a) \triangleright \llbracket b.\mathtt{co} + c.\mathbb{0} \triangleright_k \mathbb{0} \rrbracket) \xrightarrow{k'} (k'(a), k'(c) \triangleright \llbracket \mathbb{0} \triangleright_{k'} \mathbb{0} \rrbracket)$. The defender responds:

$(k(a) \rhd [\![b.\mathtt{co} \rhd_k \mathbb{0}]\!]) \xrightarrow{k'}_{\text{LTS}\star} (k'(a), k'(\star) \rhd [\![b.\mathtt{co} \rhd_k \mathbb{0}]\!]) \xrightarrow{\tau}_{\text{LTS}\mathtt{ab}} (\mathtt{ab}, \mathtt{ab} \rhd \mathbb{0})$ and get $(k'(a), k'(c) \rhd [\![\mathbb{0} \rhd_{k'} \mathbb{0}]\!]) \ \mathcal{R}_3 \ (\mathtt{ab}, \mathtt{ab} \rhd \mathbb{0})$. The rest is trivial, thus the defender always wins, therefore $\mathcal{R}_3$ is a weak bisimulation and $P_3 \approx Q_3$. □

Example 3.11. Let us now prove $P_2 \approx Q_2$ from (2) in the Introduction. For this proof we construct relation $\mathcal{R}_2$ in Fig. 3. It is easy to verify that all histories related in $\mathcal{R}_2$ are consistent and all challenger moves can be matched by the defender. Here it is noteworthy that the two tentative actions a and b are recorded in the left-hand history under different transaction names (k_1 and k_2, respectively) until the synchronisation on p merges the two transactions; these history annotations are highlighted in bold typeface. □

Example 3.12. In our final example proof we show that $P_5 \approx Q_5$ from (5) in the Introduction. Here we construct relation $\mathcal{R}_5$ in Fig. 3. In this construction, H is a history with zero or more aborted actions; we add this to our configurations because restarting transactions can nondeterministically abort and restart. The proof that $\mathcal{R}_5$ is a weak bisimulation is again by an easy inspection of the moves of the challenger. The important move is when from the pair $((H \rhd [\![b.\mathtt{co} \rhd_k P_5]\!]),\ (H \rhd [\![(b.\mathtt{co} + c.\mathtt{co}) \rhd_l Q_5]\!]))$ the challenger picks the transition $(H \rhd [\![(b.\mathtt{co} + c.\mathtt{co}) \rhd_l Q_5]\!])) \xrightarrow{l'} (H, l'(c) \rhd [\![\mathtt{co} \rhd_{l'} Q_5]\!]))$ and the defender:

$$\begin{aligned}
&((H \rhd [\![b.\mathtt{co} \rhd_k P_5]\!]) && \xrightarrow{\tau}_{(\text{LTS}\mathtt{ab})} (H \rhd P_5) \\
&\xrightarrow{\tau}_{(\text{LTS}\tau)} (H \rhd [\![\tau.b.\mathtt{co} + \tau.c.\mathtt{co} \blacktriangleright P_5]\!]) && \xrightarrow{\tau}_{(\text{LTS}\mathtt{new})} (H \rhd [\![\tau.b.\mathtt{co} + \tau.c.\mathtt{co} \rhd_k P_5]\!]) \\
&\xrightarrow{\tau}_{(\text{LTS}k(\tau))} (H \rhd [\![c.\mathtt{co} \rhd_{k'} P_5]\!]) && \xrightarrow{l'}_{(\text{LTS}k(a))} (H, l'(c) \rhd [\![\mathtt{co} \rhd_{l'} P_5]\!])
\end{aligned}$$

and get $(H, l'(c) \rhd [\![\mathtt{co} \rhd_{l'} Q_5]\!])) \ \mathcal{R}_5 \ (H, l'(c) \rhd [\![\mathtt{co} \rhd_{l'} P_5]\!])$. □

4 Predictive bisimulations

In the previous section we showed that weak bisimulations provide an effective verification technique of equivalence. However, it does not enable a direct soundness proof. The difficulty is in proving weak bisimulation *compositional* (i.e., a congruence). Here we define *predictive bisimulations*, an alternative notion of bisimulations that allows us to give an indirect proof of soundness of $(\approx)$. First we explain the problem with directly proving $(\approx)$ compositional.

Failing Proof (Compositionality) We need to prove $\mathcal{R}$ a weak bisimulation, when

$$(H_1 \rhd P \,|\, R) \ \mathcal{R} \ (H_2 \rhd Q \,|\, R) \tag{7}$$

for any context R and $(H_1 \rhd P) \approx (H_2 \rhd Q)$. Let $(H_1 \rhd P) \xrightarrow{k} (H_1, k(a) \rhd P')$ and R can perform $k(\overline{a})$ to become R'. We have $(H_1 \rhd P \,|\, R) \xrightarrow{\tau} (H_1 \rhd P' \,|\, R')$. We need to show that there exist H_2', Q'', and R'' such that $(H_2 \rhd Q \,|\, R) \overset{\tau}{\Rightarrow} (H_2' \rhd Q'' \,|\, R'')$ and $(H_1 \rhd P' \,|\, R') \ \mathcal{R} \ (H_2' \rhd Q'' \,|\, R'')$. Weak bisimilarity gives

$$(H_2 \rhd Q) \overset{k}{\Rightarrow} (H_2, k(\hat{b}) \rhd Q') \quad \text{and} \quad (H_1, k(a) \rhd P') \approx (H_2, k(\hat{b}) \rhd Q')$$

for some $\hat{b}$, Q', and the new parts $k(a)$ and $k(\hat{b})$ of the histories are consistent. However, consistency does not restrict the values of $\hat{b}$; it can be a name different than a, or even $\star$, provided that k does not commit in any extension of the bisimulation game. When $\hat{b} = a$ we can complete the proof by taking $H_2' = (H_2, k(\hat{a}))$, $Q'' = Q'$ and $R'' = R'$, showing $(H_1, k(a) \rhd P' \mid R') \mathrel{\mathcal{R}} (H_2, k(a) \rhd Q' \mid R')$ because it is of the same shape as (7). However, when $\hat{b} = c \neq a$ or $\hat{b} = \star$ it is unclear how to proceed in this direct proof. $\not\square$

We instead prove soundness by defining $(\approx_{\mathsf{prd}})$ and showing:

$$P \approx Q \text{ implies } P \approx_{\mathsf{prd}} Q \text{ implies } P \cong_{\mathsf{rbe}} Q \tag{8}$$

To show the second implication we need to prove $(\approx_{\mathsf{prd}})$ compositional. The intuition of why this is possible is because $(\approx_{\mathsf{prd}})$ only takes into account those challenger transitions that have the possibility to be committed later in the bisimulation game. This allows us to define a stronger definition of consistency which avoids the problematic cases of the above failed direct proof. Strong consistency is a reflexive, symmetric, and transitive relation.

Definition 4.1 (Strong Consistency ($\doteqdot$)). *$H_1 \doteqdot H_2$ when:*

$$(H_1(i) = \hat{a} \;\textit{ iff }\; H_2(i) = \hat{a}) \qquad \textit{and} \qquad (\exists k. H_1(i) = k(\hat{a}) \;\textit{ iff }\; \exists l. H_2(i) = l(\hat{a}))$$

In a predictive bisimulation game the challenger only performs transitions within transactions when those transactions can commit later in the game. Thus, here we differentiate between τ- and $k(\tau)$-transitions. Moreover, we emphasise that a challenger $k(a)$ move has to be matched with an identical defender move. Thus predictive bisimulations are over the actions $\eta ::= \tau \mid k(\tau) \mid k(a) \mid \omega$.

Definition 4.2 (Pred. Bisim. Transitions). *$\mathcal{C} \xrightarrow{\eta} \mathcal{C}'$ is derived by the rules:*

$$\begin{array}{lll}
(H \rhd P) \xrightarrow{\tau} (H \rhd Q) & \textit{if } P \xrightarrow{\tau}_{\varepsilon} Q & (\text{LTS}'\tau) \\
(H \rhd P) \xrightarrow{k(\tau)} (\sigma(H) \rhd Q) & \textit{if } P \xrightarrow{k(\tau)}_{\sigma} Q \textit{ and } k \sharp H & (\text{LTS}'k(\tau)) \\
(H \rhd P) \xrightarrow{\tau} (H \rhd Q) & \textit{if } P \xrightarrow{\mathtt{new}\,k} Q \textit{ and } k \sharp H & (\text{LTS}'\mathtt{new}) \\
(H \rhd P) \xrightarrow{\tau} (H \setminus_{\mathsf{co}} k \rhd Q) & \textit{if } P \xrightarrow{\mathtt{co}k} Q & (\text{LTS}'\mathtt{co}) \\
(H \rhd P) \xrightarrow{\tau} (H \setminus_{\mathsf{ab}} k \rhd Q) & \textit{if } P \xrightarrow{\mathtt{ab}k} Q & (\text{LTS}'\mathtt{ab}) \\
(H \rhd P) \xrightarrow{k(a)} (\sigma(H), k(a) \rhd Q) & \textit{if } P \xrightarrow{k(a)}_{\sigma} Q \textit{ and } k \sharp H & (\text{LTS}'k(a)) \\
(H \rhd P) \xrightarrow{\omega} (\sigma(H) \rhd \mathbb{0}) & \textit{if } P \xrightarrow{\omega}_{\sigma} Q & (\text{LTS}'\omega)
\end{array}$$

We define $\overset{\eta}{\Rightarrow}$ to be $\left((\xrightarrow{\tau}) \cup (\xrightarrow{k(\tau)})\right)^*$ when $\eta \in \{\tau, k(\tau)\}$, and $\overset{\tau}{\Rightarrow}\xrightarrow{\eta}\overset{\tau}{\Rightarrow}$ otherwise.

In a predictive bisimulation game, defender moves are weak η-transitions, with no need for $k(\star)$-transitions. Challenger moves are *commitable* transitions.

Definition 4.3 (Commitable Transition). *$\mathcal{C} \xrightarrow{\eta} \mathcal{C}'$ with $\eta \in \{\tau, \omega\}$ is commitable; $\mathcal{C} \xrightarrow{k(\mu)} (H_1 \rhd P_1)$ is commitable when there exists $(H_1 \rhd P_1) \xrightarrow{\eta_1} \dots \xrightarrow{\eta_{(n+1)}} (H_n \rhd P_n)$ such that for any a and $i \notin dom(H_1)$:*

$$(H_1, (i \mapsto k(a)) \rhd P_1) \xrightarrow{\eta_1} \dots \xrightarrow{\eta_{(n+1)}} (H_n, (i \mapsto a) \rhd P_n)$$

Weak predictive bisimulations are thus defined as follows.

Definition 4.4 (Weak Predictive Bisimulation). *A binary relation $\mathcal{R}$ is a weak predictive bisimulation when for all $\mathcal{C}_1 \mathrel{\mathcal{R}} \mathcal{C}_2$:*

1. *$hist(\mathcal{C}_1)$ and $hist(\mathcal{C}_2)$ are strongly consistent,*
2. *if $\mathcal{C}_1 \xrightarrow{\eta} \mathcal{C}_1'$ is a commitable transition and $ftn(\eta) \sharp \mathcal{C}_2$ then $\exists\ \mathcal{C}_2'$ such that $\mathcal{C}_2 \overset{\eta}{\Rightarrow} \mathcal{C}_2'$ and $\mathcal{C}_1' \mathrel{\mathcal{R}} \mathcal{C}_2'$, and its converse.*

Weak predictive bisimilarity $(\approx_{\mathsf{prd}})$ is the largest such bisimulation and $(\approx_{\mathsf{prd}})$ extends to processes in the same way as $(\approx)$. The first part of (8) follows by:

Theorem 4.5. *Let $\mathcal{C} \approx \mathcal{C}'$ with strongly consistent histories; then $\mathcal{C} \approx_{\mathsf{prd}} \mathcal{C}'$.*

Proof. The proof of this proposition relies on showing that strong consistency is closed under commitable transitions of weakly bisimilar configurations. □

To prove the second part of (8) we need to show that $(\approx_{\mathsf{prd}})$ is compositional.

Theorem 4.6. *If $P \approx_{\mathsf{prd}} Q$ and $ftn(R) \sharp P, Q$ then $P \mid R \approx_{\mathsf{prd}} Q \mid R$.*

Proof. This relies on de-/re-composition of actions; e.g., we need to decompose $(H \rhd P \mid R) \xrightarrow{\eta} (H \rhd P' \mid R')$ into the constituent sub-actions from P, R with appropriate histories. This is facilitated by strongly consistent histories. □

5 Full abstraction

Using $(\approx)$ we can prove soundness of our original bisimulation game.

Theorem 5.1 (Soundness). *If $P \approx Q$ then $P \cong_{\mathsf{rbe}} Q$.*

Proof. In view of Thm. 4.5 it is sufficient to prove the result for $(\approx_{\mathsf{prd}})$. The major step in this proof is already established in Thm. 4.6. □

We prove completeness for $\mathcal{L}_{\mathsf{Act}}$ by first translating any history H into a process $(\!|H|\!)$. Then we show that the transitions of a configuration $(H \rhd P)$ examined by bisimulations can be modelled by reductions of the process $(\!|H|\!) \mid P$, when put in parallel with the appropriate contexts. The translation of H is the parallel composition of the translation of each element in H according to:

$$\begin{array}{lll} (\!|i \mapsto k(a)|\!) = [\![\mathsf{co} \mid \omega_{ai}^{\text{commit}} \rhd_k \omega_i^{\text{abort}}]\!] & (\!|i \mapsto a|\!) = \omega_{ai}^{\text{commit}} & (\!|i \mapsto \mathsf{ab}|\!) = \omega_i^{\text{abort}} \\ (\!|i \mapsto k(\star)|\!) = [\![\mathsf{co} \rhd_k \omega_i^{\text{abort}}]\!] & (\!|i \mapsto \star|\!) = \mathbb{0} & \end{array}$$

A tentative $k(a)$-action, recorded in the history as $(i \mapsto k(a))$, corresponds to a particular move of the bisimulation game—say the ith move. This is translated to a k-transaction which is ready to commit. When k commits because *all* distributed parts of k commit, a unique "success" barb $\omega_{ai}^{\text{commit}}$ becomes observable, signalling that the ith move in the bisimulation game was a synchronisation on a which became permanent. In the history this is recorded as $(i \mapsto a)$. If the

k-transaction aborts then a unique ω_i^{abort} barb signals the abortion of the ith move, corresponding to $(i \mapsto \mathsf{ab})$ in the history. The translation of a defender's $(i \mapsto k(\star))$ move is similar, with the exception that this is a no-action that has no "success" barb associated with it. A key proposition is that reductions of translated configurations model silent bisimulation transitions.

Proposition 5.2. $(H_1 \rhd P) \xrightarrow{\tau} (H_2 \rhd Q)$ *iff* $(\!|H_1|\!) \mid P \to (\!|H_2|\!) \mid Q$.

Moreover, the ith tentative k-transition in the bisimulation game is modelled by the reduction induced by the context:

$$(\!|k|\!)^i = [\![\mathsf{co} \mid (\textstyle\sum_{a \in \mathsf{Act}} \overline{a}.\omega_{ai}^{\text{commit}}) + \tau.\mathbb{0} + \omega_i^{\text{before}} \rhd_k \omega_i^{\text{abort}}]\!]$$

When synchronising with a process, this context becomes $(\!|i \mapsto k(a)|\!)$ (for any a), modelling a tentative $k(a)$-transition. It may also spontaneously become $(\!|i \mapsto k(\star)|\!)$, modelling a $k(\star)$-transition. In any case it loses the weak barb ω_i^{before}.

Proposition 5.3. *Let* $H_2 = \sigma(H_1), (i \mapsto k(\hat{a}))$ *and* $k' \sharp k, H_1, P$, *and* $ftn(H_1) \subseteq ftn(P)$*; then* $(H_1 \rhd P) \xrightarrow{k} (H_2 \rhd Q)$ *iff* $(\!|H_1|\!) \mid P \mid (\!|k'|\!)^i \to (\!|H_2|\!) \mid Q \not\Downarrow \omega_i^{before}$. □

Theorem 5.4 (Completeness). *If* $P, Q \in \mathcal{L}_{\mathsf{Act}}$ *and* $P \cong_{\mathsf{rbe}} Q$ *then* $P \approx Q$.

Proof. Using the above propositions we show $\mathcal{X}$ is a weak bisimulation, where $\mathcal{X} = \{((H \rhd P), (H' \rhd Q)) \mid H, H' \text{ cons.}, P, Q \in \mathcal{L}_{\mathsf{Act}}, (\!|H|\!) \mid P \cong_{\mathsf{rbe}} (\!|H'|\!) \mid Q\}$. □

6 Conclusions

We presented a weak bisimulation theory for *TCCS*^{m}, a simple language with communicating transactions. In *TCCS*^{m}, two transactions that communicate are conjoined by being renamed to the same name forming a *distributed* version of cJoin's merged transactions [1]. When a transaction communicates with a non-transactional process, the latter is embedded in the former in line with the semantics of previous work [3]. Compared to that semantics, embedding and merging in *TCCS*^{m} is communication-driven limiting nondeterminism. For simplicity this language has only single-level transactions, although we believe that the results of this paper can be adapted to a language with nested transactions.

The bisimulation equivalence is sound and complete with respect to a natural contextual equivalence which equates transactions that differ only in uncommitable actions; these are destined to eventually be rolled back. We motivated this with examples in the Introduction which we verified in Sect. 3 using our technique. In related work about reversible calculi [10,9], contextual equivalence can discriminate between transactions $[\![b + a.b.\mathsf{co} \rhd_k \mathbb{0}]\!]$ and $[\![a.b.\mathsf{co} \rhd_k \mathbb{0}]\!]$ by virtue of the fact that the former contains an extra uncommitable b-action.

Our bisimulations provide an effective verification technique for the aforementioned contextual equivalence which can be applied to related programming languages where uncommitable actions have no effect [5,7,11,16]. They can also serve as a verification technique for other forms of contextual equivalences, such as may- and must-testing equivalences [3,4]. Other bisimulation methods for reversible computation [2,8,9,10] may also be used in these settings, but they are more fine-grained, discriminating even between the above two processes. Forward-reverse and hereditary history-preserving bisimulations [13] differentiate between forward and reverse transitions, which would discriminate between the processes P_5 and Q_5 shown in the Introduction capturing a possible compiler optimisation. To our knowledge, the bisimulation technique presented here is the only one that can be used to prove the correctness of such compiler optimisations.

References

1. R. Bruni, H. C. Melgratti, and U. Montanari. Nested commits for mobile calculi: extending Join. In *IFIP TCS*, pages 563–576, 2004.
2. V. Danos and J. Krivine. Transactions in RCCS. In M. Abadi and L. Alfaro, editors, *CONCUR*, volume 3653 of *LNCS*, pages 398–412. Springer, 2005.
3. E. de Vries, V. Koutavas, and M. Hennessy. Communicating transactions. In P. Gastin and F. Laroussinie, editors, *CONCUR*, volume 6269 of *LNCS*, pages 569–583. Springer, 2010.
4. E. De Vries, V. Koutavas, and M. Hennessy. Liveness of communicating transactions. In K. Ueda, editor, *APLAS*, volume 6461 of *LNCS*, pages 392–407. Springer, 2010.
5. K. Donnelly and M. Fluet. Transactional events. In *ICFP*, pages 124–135. ACM, 2006.
6. L. Effinger-Dean, M. Kehrt, and D. Grossman. Transactional events for ML. In *ICFP*, pages 103–114. ACM, 2008.
7. J. Field and C. A. Varela. Transactors: a programming model for maintaining globally consistent distributed state in unreliable environments. In *POPL*, pages 195–208. ACM, 2005.
8. J. Krivine. A verification technique for reversible process algebra. In R. Glck and T. Yokoyama, editors, *RC*, volume 7581 of *LNCS*, pages 204–217. Springer, 2013.
9. I. Lanese, M. Lienhardt, C. A. Mezzina, A. Schmitt, and J.-B. Stefani. Concurrent flexible reversibility. In M. Felleisen and P. Gardner, editors, *ESOP*, volume 7792 of *LNCS*, pages 370–390. Springer, 2013.
10. I. Lanese, C. A. Mezzina, and J.-B. Stefani. Reversing higher-order pi. In P. Gastin and F. Laroussinie, editors, *CONCUR*, volume 6269 of *LNCS*, pages 478–493. Springer, 2010.
11. M. Lesani and J. Palsberg. Communicating memory transactions. In *PPoPP*, pages 157–168. ACM, 2011.
12. R. Milner. *A Calculus of Communicating Systems*. Springer, 1982.
13. I. Phillips and I. Ulidowski. Reversibility and models for concurrency. *ENTCS*, 192(1):93 – 108, 2007. (SOS 2007).
14. D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *LICS*, pages 293–302. IEEE Computer Society, 2007.
15. D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
16. C. Spaccasassi and V. Koutavas. Towards efficient abstractions for concurrent consensus. In *TFP*, 2013. To appear.
17. Colin Stirling. Playing games and proving properties of concurrent systems. *J. Comput. Sci. Technol.*, 13(6):482, 1998.