View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Dagstuhl Research Online Publication Server

Reactive Bisimulation Semantics for a Process Algebra with Time-Outs

Rob van Glabbeek
Data61, CSIRO, Sydney, Australia
UNSW, Sydney, Australia
rvg@cs.stanford.edu

Abstract

This paper introduces the counterpart of strong bisimilarity for labelled transition systems extended with time-out transitions. It supports this concept through a modal characterisation, congruence results for a standard process algebra with recursion, and a complete axiomatisation.

2012 ACM Subject Classification Theory of computation → Process calculi

Keywords and phrases Process algebra, time-outs, labelled transition systems, reactive bisimulation semantics, Hennessy-Milner logic, modal characterisations, recursion, complete axiomatisations

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.6

Related Version Like many conference papers, this is an extended abstract, with omitted proofs. The full version of this paper is available at `http://rvg.web.cse.unsw.edu.au/pub/reactive.pdf`.

Acknowledgements My thanks to the referees for helpful feedback.

1 Introduction

This is a contribution to classic untimed non-probabilistic process algebra, modelling systems that move from state to state by performing discrete, uninterpreted actions. A system is modelled as a process-algebraic expression, whose standard semantics is a state in a labelled transition system (LTS). An LTS consists of a set of states, with action-labelled transitions between them. The execution of an action is assumed to be instantaneous, so when any time elapses the system must be in one of its states. With "untimed" I mean that I will refrain from quantifying the passage of time; however, whether a system can pause in some state or not will be part of my model.

Following [29], I consider *reactive* systems that interact with their environments through the synchronous execution of visible actions a, b, c, ... taken from an alphabet A. At any time, the environment *allows* a set of actions $X \subseteq A$, while *blocking* all other actions. At discrete moments the environment can change the set of actions it allows. In a metaphor from [29], the environment of a system can be seen as a user interacting with it. This user has a button for each action $a \in A$, on which it can exercise pressure. When the user exercises pressure *and* the system is in a state where it can perform action a, the action occurs. For the system this involves taking an a-labelled transition to a following state; for the environment it entails the button going down, thus making the action occurrence observable. This can trigger the user to alter the set of buttons on which it exercises pressure.

The current paper considers two special actions that can occur as transition labels: the traditional *hidden action* τ [29], modelling the occurrence of an instantaneous action from which we abstract, and the *time-out* action t, modelling the end of a time-consuming activity from which we abstract. The latter was introduced in [16] and constitutes the main novelty of the present paper with respect to [29] and forty years of process algebra. Both special actions are unobservable, in the sense that their occurrence cannot trigger any state-change in the environment. Conversely, the environment cannot cause or block their occurrence.

© Rob van Glabbeek;
licensed under Creative Commons License CC-BY

31st International Conference on Concurrency Theory (CONCUR 2020).
Editors: Igor Konnov and Laura Kovács; Article No. 6; pp. 6:1–6:23
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Following [16], I model the passage of time in the following way. When a system arrives in a state P, and at that time X is the set of actions allowed by the environment, there are two possibilities. If P has an outgoing transition $P \xrightarrow{\alpha} Q$ with $\alpha \in X \cup \{\tau\}$, the system immediately takes one of the outgoing transitions $P \xrightarrow{\alpha} Q$ with $\alpha \in X \cup \{\tau\}$, without spending any time in state P. The choice between these actions is entirely nondeterministic. The system cannot immediately take a transition $\xrightarrow{b}$ with $b \in A \backslash X$, because the action b is blocked by the environment. Neither can it immediately take a transition $P \xrightarrow{\mathrm{t}} Q$, because such transitions model the end of an activity with a finite but positive duration that started when reaching state P.

In case P has no outgoing transition $P \xrightarrow{\alpha} Q$ with $\alpha \in X \cup \{\tau\}$, the system idles in state P for a positive amount of time. This idling can end in two possible ways. Either one of the time-out transitions $P \xrightarrow{\mathrm{t}} Q$ occurs, or the environment spontaneously changes the set of actions it allows into a different set Y with the property that $P \xrightarrow{a} Q$ for some $a \in Y$. In the latter case a transition $P \xrightarrow{a} Q$ occurs, with $a \in Y$. The choice between the various ways to end a period of idling is entirely nondeterministic. It is possible to stay forever in state P only if there are no outgoing time-out transitions $P \xrightarrow{\mathrm{t}} Q$.

The addition of time-outs enhances the expressive power of LTSs and process algebras. The process $a.P + \mathrm{t}.b.Q$, for instance, models a choice between $a.P$ and $b.Q$ where the former has priority. In an environment where a is allowed it will always choose $a.P$ and never $b.Q$; but in an environment that blocks a the process will, after some delay, proceed with $b.Q$. Such a priority mechanism cannot be modelled in standard process algebras without time-outs, such as CCS [29], CSP [4, 24] and ACP [1, 8]. Additionally, mutual exclusion cannot be correctly modelled in any of these standard process algebras [17], but adding time-outs makes it possible – see Section 11 for a more precise statement.

In [16] I characterised the coarsest reasonable semantic equivalence on LTSs with time-outs – the one induced by *may testing*, as proposed by De Nicola & Hennessy [6]. In the absence of time-outs, may testing yields *weak trace equivalence*, where two processes are defined equivalent iff they have the same *weak traces*: sequence of actions the system can perform, while eliding hidden actions. In the presence of time-outs weak trace equivalence fails to be a congruence for common process algebraic operators, and may testing yields its congruence closure, characterised in [16] as *(rooted) failure trace equivalence*.

The present paper aims to characterise one of the finest reasonable semantic equivalences on LTSs with time-outs – the counterpart of strong bisimilarity for LTSs without time-outs. Naturally, strong bisimilarity can be applied verbatim to LTSs with time-outs – and has been in [16] – by treating t exactly like any visible action. Here, however, I aim to take into account the essence of time-outs, and propose an equivalence that satisfies some natural laws discussed in [16], such as $\tau.P + \mathrm{t}.Q = \tau.P$ and $a.P + \mathrm{t}.(Q + \tau.R + a.S) = a.P + \mathrm{t}.(Q + \tau.R)$. To motivate the last law, note that the time-out transition $a.P + \mathrm{t}.(Q + \tau.R + a.S) \xrightarrow{\mathrm{t}} Q + \tau.R + a.S$ can occur only in an environment that blocks the action a, for otherwise a would have taken place before the time-out went off. The occurrence of this transition is not observable by the environment, so right afterwards the state of the environment is unchanged, and the action a is still blocked. Therefore, the process $Q + \tau.R + a.S$ will, without further ado, proceed with the τ-transition to R, or any action from Q, just as if the $a.S$ summand were not present.

Standard process algebras and LTSs without time-outs can model systems whose behaviour is triggered by input signals from the environment in which they operate. This is why they are called “reactive systems”. By means of time-outs one can additionally model systems whose behaviour is triggered by the *absence* of input signals from the environment, during a sufficiently long period. This creates a greater symmetry between a system and its environment, as it has always been understood that the environment or user of a system can change its behaviour as a result of sustained inactivity of the system it is interacting with. Hence one could say that process algebras and LTSs enriched with time-outs form a more faithful model of reactivity. It is for this reason that I use the name *reactive bisimilarity* for the appropriate form of bisimilarity on systems modelled in this fashion.

Section 2 introduces strong reactive bisimilarity as the proper counterpart of strong bisimilarity in the presence of time-out transitions. Naturally, it coincides with strong bisimilarity when there are no time-out transitions. Section 3 derives a modal characterisation; a reactive variant of the Hennessy-Milner logic. Section 4 offers an alternative characterisation of strong reactive bisimilarity that will be more convenient in proofs, although it is lacks the intuitive appeal to be used as the initial definition.

Section 5 recalls the process algebra CCSP, a common mix of CCS and CSP, and adds the time-out action, as well as two auxiliary operators that will be used in the forthcoming axiomatisation. Section 6 states that in this process algebra one can express all countably branching transition systems, and only those, or all and only the finitely branching ones when restricting to guarded recursion.

Section 7 recalls the concept of a congruence, focusing on the congruence property for the recursion operator, which is commonly the hardest to establish. It then shows that the simple *initials equivalence*, as well as Milner's strong bisimilarity, are congruences. Due to the presence of negative premises in the operational rules for the auxiliary operators, these proofs are not entirely trivial. Using these results as a stepping stone, Section 8 shows that strong reactive bisimilarity is a congruence for my extension of CCSP. Here the congruence property for one of the auxiliary operators with negative premises is needed in establishing the result for the common CCSP operators, such as parallel composition.

Section 9 shows that guarded recursive specifications have unique solutions up to strong reactive bisimilarity. Using this, Section 10 provides a sound and complete axiomatisation for processes with guarded recursion. My completeness proof combines three innovations in establishing completeness of process algebraic axiomatisations. First of all, following [19], it applies to *all* processes in a Turing powerful language like guarded CCSP, rather than the more common fragment merely employing finite sets of recursion equations featuring only choice and action prefixing. Secondly, instead of the classic technique of *merging guarded recursive equations* [27, 28, 35, 9, 26], which in essence proves two bisimilar systems P and Q equivalent by equating both to an intermediate variant that is essentially a *product* of P and Q, I employ the novel method of *canonical solutions* [25], which equates both P and Q to a canonical representative within the bisimulation equivalence class of P and Q – one that has only one reachable state for each bisimulation equivalence class of states of P and Q. In fact I tried so hard, and in vain, to apply the traditional technique of merging guarded recursive equations, that I came to believe that it fundamentally does not work for this axiomatisation. The third innovation is the use of the axiom of choice [36] in defining the transition relation on my canonical representative, in order to keep this process finitely branching.

Section 11 describes a worthwhile gain in expressiveness caused by the addition of time-outs, and presents an agenda for future work.

Due to lack of space, all proofs have been deleted. However, some appear in the appendix – their theorems are marked @ – and all can be found in the accompanying technical report.

2 Reactive bisimilarity

A *labelled transition system* (LTS) is a triple $(\mathbb{P}, Act, \rightarrow)$ with $\mathbb{P}$ a set (of *states* or *processes*), Act a set (of *actions*) and $\rightarrow\, \in \mathbb{P} \times Act \times \mathbb{P}$. In this paper I consider LTSs with $Act := A \uplus \{\tau, \mathrm{t}\}$, where A is a set of *visible actions*, τ is the *hidden action*, and t the *time-out* action. The set of *initial* actions of a process $P \in \mathbb{P}$ is $\mathcal{I}(P) := \{\alpha \in A \cup \{\tau\} \mid P \xrightarrow{\alpha}\}$. Here $P \xrightarrow{\alpha}$ means that there is a Q with $P \xrightarrow{\alpha} Q$.

▶ **Definition 1.** *A* strong reactive bisimulation *is a symmetric relation $\mathscr{R} \subseteq (\mathbb{P} \times \mathscr{P}(A) \times \mathbb{P}) \cup (\mathbb{P} \times \mathbb{P})$ (meaning that $(P, X, Q) \in \mathscr{R} \Leftrightarrow (Q, X, P) \in \mathscr{R}$ and $(P, Q) \in \mathscr{R} \Leftrightarrow (Q, P) \in \mathscr{R}$), such that,*

- *if $(P, Q) \in \mathscr{R}$ and $P \xrightarrow{\tau} P'$, then there exists a Q' such that $Q \xrightarrow{\tau} Q'$ and $(P', Q') \in \mathscr{R}$,*
- *if $(P, Q) \in \mathscr{R}$ then $(P, X, Q) \in \mathscr{R}$ for all $X \subseteq A$,*

and for all $(P, X, Q) \in \mathscr{R}$,

- *if $P \xrightarrow{a} P'$ with $a \in X$, then there exists a Q' such that $Q \xrightarrow{a} Q'$ and $(P', Q') \in \mathscr{R}$,*
- *if $P \xrightarrow{\tau} P'$, then there exists a Q' such that $Q \xrightarrow{\tau} Q'$ and $(P', X, Q') \in \mathscr{R}$,*
- *if $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$, then $(P, Q) \in \mathscr{R}$, and*
- *if $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$, then $\exists Q'$ such that $Q \xrightarrow{\mathrm{t}} Q'$ and $(P', X, Q') \in \mathscr{R}$.*

Processes $P, Q \in \mathbb{P}$ are strongly X-bisimilar, *denoted $P \leftrightarroweq_r^X Q$, if $(P, X, Q) \in \mathscr{R}$ for some strong reactive bisimulation $\mathscr{R}$. They are* strongly reactive bisimilar, *denoted $P \leftrightarroweq_r Q$, if $(P, Q) \in \mathscr{R}$ for some strong reactive bisimulation $\mathscr{R}$.*

Intuitively, $(P, X, Q) \in \mathscr{R}$ says that processes P and Q behave the same way, as witnessed by the relation $\mathscr{R}$, when placed in the environment X – meaning any environment that allows exactly the actions in X to occur – whereas $(P, Q) \in \mathscr{R}$ says they behave the same way in an environment that has just been triggered to change. An environment can be thought of as an unknown process placed in parallel with P and Q, using the operator $\|_A$, enforcing synchronisation on all visible actions. The environment X can be seen as a process $\sum_{i \in I} a_i.R_i + \mathrm{t}.R$ where $\{a_i \mid i \in I\} = X$. A triggered environment, on the other hand, can execute a sequence of instantaneous hidden actions before stabilising as an environment Y, for $Y \subseteq A$. During this execution, actions can be blocked and allowed in rapid succession. Since the environment is unknown, the bisimulation should be robust under any such environment.

The first clause for $(P, X, Q) \in \mathscr{R}$ is like the common transfer property of strong bisimilarity [29]: a visible a-transition of P can be matched by one of Q, such that the resulting processes P' and Q' are related again. However, I require it only for actions $a \in X$, because actions $b \in A \backslash X$ cannot happen at all in the environment X, and thus need not be matched by Q. Since the occurrence of a is observable by the environment, this can trigger the environment to change the set of actions it allows, so P' and Q' ought to be related in a triggered environment.

The second clause is the transfer property for τ-transitions. Since these are not observable by the environment, they cannot trigger a change in the set of actions allowed by it, so the resulting processes P' and Q' should be related only in the same environment X.

The first clause for $(P, Q) \in \mathscr{R}$ expresses the transfer property for τ-transitions in a triggered environment. Here it may happen that the τ-transition occurs before the environment stabilises, and hence P' and Q' will still be related in a triggered environment. A similar transfer property for a-transitions is already implied by the next two clauses.

The second clause allows a triggered environment to stabilise into any environment X.

The first two clauses for $(P, X, Q) \in \mathscr{R}$ imply that if $(P, X, Q) \in \mathscr{R}$ then $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \mathcal{I}(Q) \cap (X \cup \{\tau\})$. So $P \leftrightarroweq_r Q$ implies $\mathcal{I}(P) = \mathcal{I}(Q)$. The condition $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ is necessary and sufficient for the system to remain a positive amount of time in state P when

X is the set of allowed actions. The next clause says that during this time the environment may be triggered to change the set of actions it allows by an event outside our model, that is, by a time-out in the environment. So P and Q should be related in a triggered environment.

The last clause says that also a t-transition of P should be matched by one of Q. Naturally, the t-transition of P can be taken only when the system is idling in P, i.e., when $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$. The resulting processes P' and Q' should be related again, but only in the same environment allowing X.

▶ **Proposition 2** (@). *$\leftrightarroweq_r$ and $\leftrightarroweq_r^X$ for $X \subseteq A$ are equivalence relations.*

Note that the union of arbitrarily many strong reactive bisimulations is itself a strong reactive bisimulation. Consequently, the family of relations $\leftrightarroweq_r$, $\leftrightarroweq_r^X$ for $X \subseteq A$ can be seen as a strong reactive bisimulation.

To get a firm grasp on strong reactive bisimilarity, the reader is invited to check the two laws mentioned in the introduction, and then to construct a strong reactive bisimulation between the two systems depicted on Page 14.

2.1 A more general form of reactive bisimulation

The following notion of a *generalised strong reactive bisimulation* (gsrb) generalises that of a strong reactive bisimulation; yet it induces the same concept of strong reactive bisimilarity. This makes the relation convenient to use for further analysis. I did not introduce it as the original definition, because it lacks a strong motivation.

▶ **Definition 3.** *A* gsrb *is a symmetric relation $\mathscr{R} \subseteq (\mathbb{P} \times \mathscr{P}(A) \times \mathbb{P}) \cup (\mathbb{P} \times \mathbb{P})$ such that, for all $(P,Q) \in \mathscr{R}$,*

- *if $P \xrightarrow{\alpha} P'$ with $a \in A \cup \{\tau\}$, then there exists a Q' such that $Q \xrightarrow{\alpha} Q'$ and $(P',Q') \in \mathscr{R}$,*
- *if $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ with $X \subseteq A$ and $P \xrightarrow{\mathrm{t}} P'$, then there exists a Q' with $Q \xrightarrow{\mathrm{t}} Q'$ and $(P',X,Q') \in \mathscr{R}$,*

and for all $(P,Y,Q) \in \mathscr{R}$,

- *if $P \xrightarrow{a} P'$ with either $a \in Y$ or $\mathcal{I}(P) \cap (Y \cup \{\tau\}) = \emptyset$, then there exists a Q' with $Q \xrightarrow{a} Q'$ and $(P',Q') \in \mathscr{R}$,*
- *if $P \xrightarrow{\tau} P'$, then there exists a Q' such that $Q \xrightarrow{\tau} Q'$ and $(P',Y,Q') \in \mathscr{R}$,*
- *if $\mathcal{I}(P) \cap (X \cup Y \cup \{\tau\}) = \emptyset$ with $X \subseteq A$ and $P \xrightarrow{\mathrm{t}} P'$ then there exists a Q' with $Q \xrightarrow{\mathrm{t}} Q'$ and $(P',X,Q') \in \mathscr{R}$.*

▶ **Proposition 4** (@). *$P \leftrightarroweq_r Q$ iff there exists a gsrb $\mathscr{R}$ with $(P,Q) \in \mathscr{R}$.*
Likewise, $P \leftrightarroweq_r^X Q$ iff there exists a gsrb $\mathscr{R}$ with $(P,X,Q) \in \mathscr{R}$.

Unlike Definition 1, a gsrb needs the triples (P,X,Q) only after encountering a t-transition; two systems without t-transitions can be related without using these triples at all.

3 A modal characterisation of strong reactive bisimilarity

The Hennessy-Milner logic [23] expresses properties on the behaviour of processes in an LTS.

▶ **Definition 5.** *The class $\mathbb{O}$ of* infinitary HML formulas *is defined as follows, where I ranges over all index sets and α over $A \cup \{\tau\}$:*

$$\varphi \quad ::= \quad \bigwedge_{i \in I} \varphi_i \;\mid\; \neg\varphi \;\mid\; \langle\alpha\rangle\varphi$$

$\top$ abbreviates the empty conjunction, and $\varphi_1 \wedge \varphi_2$ stands for $\bigwedge_{i\in\{1,2\}} \varphi_i$.

$P \models \varphi$ denotes that process P satisfies formula φ. The first two operators represent standard Boolean operators. By definition, $P \models \langle\alpha\rangle\varphi$ iff $P \xrightarrow{\alpha} P'$ for some P' with $P' \models \varphi$.

A famous result stemming from [23] states that

$$P \mathrel{\underline{\leftrightarrow}} Q \quad\Leftrightarrow\quad \forall\varphi\in\mathbb{O}.\ (P\models\varphi \Leftrightarrow Q\models\varphi)$$

where $\mathrel{\underline{\leftrightarrow}}$ denotes strong bisimilarity [29, 23], formally defined in Section 7. It states that the Hennessy-Milner logic yields a *modal characterisation* of strong bisimilarity. I will now adapt this into a modal characterisation of strong reactive bisimilarity.

To this end I extend the Hennessy-Milner logic with a new modality $\langle X\rangle$, for $X \subseteq A$, and auxiliary satisfaction relations $\models_X \subseteq \mathbb{P}\times\mathbb{O}$ for each $X \subseteq A$. The formula $P \models \langle X\rangle\varphi$ says that in an environment X, allowing exactly the actions in X, process P can perform a time-out transition to a process that satisfies φ. $P \models_X \varphi$ says that if P satisfies φ when placed in environment X. The relations $\models$ and $\models_X$ are the smallest ones satisfying:

$$\begin{array}{lll}
P \models \bigwedge_{i\in I}\varphi_i & & \text{if}\quad \forall i\in I.\ P\models\varphi_i\\
P \models \neg\varphi & & \text{if}\quad P \not\models \varphi\\
P \models \langle\alpha\rangle\varphi & \text{with } \alpha\in A\cup\{\tau\} & \text{if}\quad \exists P'.\ P\xrightarrow{\alpha}P' \wedge P'\models\varphi\\
P \models \langle X\rangle\varphi & \text{with } X\subseteq A & \text{if}\quad \mathcal{I}(P)\cap(X\cup\{\tau\})=\emptyset \wedge \exists P'.\ P\xrightarrow{\mathrm{t}}P'\wedge P'\models_X\varphi
\end{array}$$

$$\begin{array}{lll}
P \models_X \bigwedge_{i\in I}\varphi_i & & \text{if}\quad \forall i\in I.\ P\models_X\varphi_i\\
P \models_X \neg\varphi & & \text{if}\quad P \not\models_X \varphi\\
P \models_X \langle a\rangle\varphi & \text{with } a\in A & \text{if}\quad a\in X \wedge \exists P'.\ P\xrightarrow{a}P' \wedge P'\models\varphi\\
P \models_X \langle\tau\rangle\varphi & & \text{if}\quad \exists P'.\ P\xrightarrow{\tau}P' \wedge P'\models_X\varphi\\
P \models_X \varphi & & \text{if}\quad \mathcal{I}(P)\cap(X\cup\{\tau\})=\emptyset \wedge P\models\varphi
\end{array}$$

Note that a formula $\langle a\rangle\varphi$ is less often true under $\models_X$ then under $\models$, due to the side condition $a \in X$. This reflects the fact that a cannot happen in an environment that blocks it. The last clause in the above definition reflects the fifth clause of Definition 1. If $\mathcal{I}(P)\cap(X\cup\{\tau\})=\emptyset$, then process P, operating in environment X, idles for a while, during which the environment can change. This ends the blocking of actions $a \notin X$ and makes any formula valid under $\models$ also valid under $\models_X$.

▶ **Example 6.** Both systems depicted on Page 14 satisfy $\langle\{a,b\}\rangle\langle\tau\rangle\langle b\rangle\top \wedge \langle\{a,b\}\rangle\langle\tau\rangle\neg\langle b\rangle\top \wedge \langle\{b\}\rangle\langle a\rangle\top \wedge \langle\{b\}\rangle\neg\langle a\rangle\top$ and neither satisfies $\langle\{a,b\}\rangle(\langle a\rangle \wedge \langle\tau\rangle\langle b\rangle)$ or $\langle\{b\}\rangle(\langle a\rangle \wedge \langle\tau\rangle\langle b\rangle)$.

▶ **Theorem 7** (@). *Let $P,Q\in\mathbb{P}$ and $X\subseteq A$. Then $P \mathrel{\underline{\leftrightarrow}}_r Q \Leftrightarrow \forall\varphi\in\mathbb{O}.\ (P\models\varphi\Leftrightarrow Q\models\varphi)$ and $P \mathrel{\underline{\leftrightarrow}}_r^X Q \Leftrightarrow \forall\varphi\in\mathbb{O}.\ (P\models_X\varphi\Leftrightarrow Q\models_X\varphi)$.*

4 Time-out bisimulations

I will now present a characterisation of strong reactive bisimilarity in terms of a binary relation $\mathscr{B}$ on processes – a *strong time-out bisimulation* – not parametrised by the set of allowed actions X. To this end I need a family of unary operators θ_X on processes, for $X \subseteq A$. These *environment* operators place a process in an environment that allows exactly the actions in X to occur. They are defined by the following structural operational rules.

$$\frac{x\xrightarrow{\tau}y}{\theta_X(x)\xrightarrow{\tau}\theta_X(y)} \qquad \frac{x\xrightarrow{a}y}{\theta_X(x)\xrightarrow{a}y}\ (a\in X) \qquad \frac{x\xrightarrow{\alpha}y \quad x\stackrel{\beta}{\nrightarrow}\text{ for all }\beta\in X\cup\{\tau\}}{\theta_X(x)\xrightarrow{\alpha}y}\ (\alpha\in A\cup\{\mathrm{t}\})$$

The operator θ_X modifies its argument by inhibiting all initial transitions (here including also those that occur after a τ-transition) that cannot occur in the specified environment. When an observable transition does occur, the environment may be triggered to change, and the inhibiting effect of the θ_X-operator comes to an end. The premises $x \overset{\beta}{\not\longrightarrow}$ for all $\beta \in X \cup \{\tau\}$ in the third rule guarantee that the process x will idle for a positive amount of time in its current state. During this time, the environment may be triggered to change, and again the inhibiting effect of the θ_X-operator comes to an end.

Below I assume that $\mathbb{P}$ is closed under θ, that is, if $P \in \mathbb{P}$ and $X \subseteq A$ then $\theta_X(P) \in \mathbb{P}$.

▶ **Definition 8.** *A* strong time-out bisimulation *is a symmetric relation $\mathscr{B} \subseteq \mathbb{P} \times \mathbb{P}$, such that, for $P \mathrel{\mathscr{B}} Q$,*

- *if $P \xrightarrow{\alpha} P'$ with $\alpha \in A \cup \{\tau\}$, then $\exists Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathrel{\mathscr{B}} Q'$,*
- *if $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$, then $\exists Q'$ such that $Q \xrightarrow{\mathrm{t}} Q'$ and $\theta_X(P') \mathrel{\mathscr{B}} \theta_X(Q')$.*

▶ **Proposition 9** (@). *$P \leftrightarroweq_r Q$ iff there exists a strong time-out bisimulation $\mathscr{B}$ with $P \mathrel{\mathscr{B}} Q$.*

Note that the union of arbitrarily many strong time-out bisimulations is itself a strong time-out bisimulation. Consequently, the relation $\leftrightarroweq_r$ is a strong time-out bisimulation.

5 The process algebra CCSP$_{\mathrm{t}}^{\theta}$

Let A be a set of *visible actions* and *Var* an infinite set of *variables*. The syntax of CCSP$_{\mathrm{t}}^{\theta}$ is given by

$$E ::= 0 \mid \alpha.E \mid E{+}E \mid E\|_S E \mid \tau_I(E) \mid \mathcal{R}(E) \mid \theta_L^U(E) \mid \psi_X(E) \mid x \mid \langle x|\mathcal{S}\rangle \text{ (with } x{\in}V_{\mathcal{S}})$$

with $\alpha \in Act := A \uplus \{\tau, \mathrm{t}\}$, $S, I, U, L, X \subseteq A$, $L \subseteq U$, $\mathcal{R} \subseteq A \times A$, $x \in Var$ and $\mathcal{S}$ a *recursive specification*: a set of equations $\{y = \mathcal{S}_y \mid y \in V_{\mathcal{S}}\}$ with $V_{\mathcal{S}} \subseteq Var$ (the *bound variables* of $\mathcal{S}$) and each $\mathcal{S}_y$ a CCSP$_{\mathrm{t}}^{\theta}$ expression. I require that all sets $\{b \mid (a,b) \in \mathcal{R}\}$ are finite.

The fragment of CCSP$_{\mathrm{t}}^{\theta}$ without θ_L^U and ψ_X is called CCSP$_{\mathrm{t}}$. Omitting t._ yields CCSP.

The constant 0 represents a process that is unable to perform any action. The process $\alpha.E$ first performs the action α and then proceeds as E. The process $E + F$ behaves as either E or F. $\|_S$ is a partially synchronous parallel composition operator; actions $a \in S$ must synchronise – they can occur only when both arguments are ready to perform them – whereas actions $\alpha \notin S$ from both arguments are interleaved. τ_I is an abstraction operator; it conceals the actions in I by renaming them into the hidden action τ. The operator $\mathcal{R}$ is a relational renaming: it renames a given action $a \in A$ into a choice between all actions b with $(a,b) \in \mathcal{R}$. The *environment operators* θ_L^U and ψ_X are new in this paper and explained below. Finally, $\langle x|\mathcal{S}\rangle$ represents the x-component of a solution of the system of recursive equations $\mathcal{S}$.

The language CCSP is a common mix of the process algebras CCS [29] and CSP [4, 24]. It first appeared in [30], where it was named following a suggestion by M. Nielsen. The family of parallel composition operators $\|_S$ stems from [31], and incorporates the two CSP parallel composition operators from [4]. The relation renaming operators $\mathcal{R}(\_)$ stem from [34]; they combine both the (functional) renaming operators that are common to CCS and CSP, and the inverse image operators of CSP. The choice operator + stems from CCS, and the abstraction operator from CSP, while the inaction constant 0, action prefixing operators $a.\_$ for $a \in A$, and the recursion construct are common to CCS and CSP. The time-out prefixing operator t._ was added by me in [16]. The syntactic form of inaction 0, action prefixing $\alpha.E$ and choice $E + F$ follows CCS, whereas the syntax of abstraction $\tau_I(\_)$ and recursion $\langle x|\mathcal{S}\rangle$ follows ACP [1, 8].

Table 1 Structural operational interleaving semantics of $\mathrm{CCSP}_{\mathrm{t}}^{\theta}$.

$$\alpha.x \xrightarrow{\alpha} x \qquad \frac{x \xrightarrow{\alpha} x'}{x + y \xrightarrow{\alpha} x'} \qquad \frac{y \xrightarrow{\alpha} y'}{x + y \xrightarrow{\alpha} y'} \qquad \frac{x \xrightarrow{\alpha} x'}{\mathcal{R}(x) \xrightarrow{\beta} \mathcal{R}(x')} \left(\begin{array}{r} \alpha{=}\beta{=}\tau \\ \vee\ \ \alpha{=}\beta{=}\mathrm{t} \\ \vee\ (\alpha,\beta){\in}\mathcal{R} \end{array}\right)$$

$$\frac{x \xrightarrow{\alpha} x'}{x \parallel_S y \xrightarrow{\alpha} x' \parallel_S y}\ (\alpha \notin S) \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \parallel_S y \xrightarrow{a} x' \parallel_S y'}\ (a \in S) \qquad \frac{y \xrightarrow{\alpha} y'}{x \parallel_S y \xrightarrow{\alpha} x \parallel_S y'}\ (\alpha \notin S)$$

$$\frac{x \xrightarrow{\alpha} x'}{\tau_I(x) \xrightarrow{\alpha} \tau_I(x')}\ (\alpha \notin I) \qquad \frac{x \xrightarrow{a} x'}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}\ (a \in I) \qquad \frac{\langle \mathcal{S}_x | \mathcal{S} \rangle \xrightarrow{\alpha} y}{\langle x | \mathcal{S} \rangle \xrightarrow{\alpha} y}$$

$$\frac{x \xrightarrow{\tau} y}{\theta_L^U(x) \xrightarrow{\tau} \theta_L^U(y)} \qquad \frac{x \xrightarrow{a} y}{\theta_L^U(x) \xrightarrow{a} y}\ (a \in U) \qquad \frac{x \xrightarrow{\alpha} y \quad x \overset{\beta}{\not\longrightarrow} \text{ for all } \beta \in L{\cup}\{\tau\}}{\theta_L^U(x) \xrightarrow{\alpha} y}\ (\alpha{\in}A{\cup}\{\mathrm{t}\})$$

$$\frac{x \xrightarrow{\alpha} y}{\psi_X(x) \xrightarrow{\alpha} y}\ (\alpha \in A \cup \{\tau\}) \qquad \frac{x \xrightarrow{\mathrm{t}} y \quad x \overset{\beta}{\not\longrightarrow} \text{ for all } \beta \in X \cup \{\tau\}}{\psi_X(x) \xrightarrow{\mathrm{t}} \theta_X(y)}$$

An occurrence of a variable x in a $\mathrm{CCSP}_{\mathrm{t}}^{\theta}$ expression E is *bound* iff it occurs in a subexpression $\langle y|\mathcal{S}\rangle$ of E with $x \in V_{\mathcal{S}}$; otherwise it is *free*. Here each $\mathcal{S}_y$ for $y \in V_{\mathcal{S}}$ counts as a subexpression of $\langle x|\mathcal{S}\rangle$. An expression E is *invalid* if it has a subexpression $\theta_L^U(F)$ or $\psi_X(F)$ such that a variable occurrence in F is free in F but bound in E. Let $\mathbb{E}$ be the set of valid $\mathrm{CCSP}_{\mathrm{t}}^{\theta}$ expressions. Furthermore, $\mathbb{P} \subseteq \mathbb{E}$ is the set of *closed* valid $\mathrm{CCSP}_{\mathrm{t}}^{\theta}$ expressions, or *processes*; those in which every variable occurrence is bound.

A substitution is a partial function $\rho\colon \mathit{Var} \rightharpoonup \mathbb{E}$. The application $E[\rho]$ of a substitution ρ to an expression $E \in \mathbb{E}$ is the result of simultaneous replacement, for all $x \in \mathrm{dom}(\rho)$, of each free occurrence of x in E by the expression $\rho(x)$, while renaming bound variables in E if necessary to present name clashes.

The semantics of $\mathrm{CCSP}_{\mathrm{t}}^{\theta}$ is given by the labelled transition relation $\rightarrow \subseteq \mathbb{P} \times \mathit{Act} \times \mathbb{P}$, where the transitions $P \xrightarrow{\alpha} Q$ are derived from the rules of Table 1. Here $\langle E|\mathcal{S}\rangle$ for $E \in \mathbb{E}$ and $\mathcal{S}$ a recursive specification denotes the result of substituting $\langle y|\mathcal{S}\rangle$ for y in E, for all $y \in V_{\mathcal{S}}$. Even though negative premises occur in Table 1, the meaning of this transition system specification is well-defined, for instance by the method of *stratification* [21, 13]. This is explained in the appendix (@).

The auxiliary operators θ_L^U and ψ_X are added here to facilitate complete axiomatisation, similar to the left merge and communication merge of ACP [1, 8]. The operator θ_X^X is the same as what was called θ_X in Section 4. It inhibits those transitions of its argument that are blocked in the environment X, allowing only the actions from $X \subseteq A$. It stops inhibiting as soon as the system performs a visible action or takes a break, as this may trigger a change in the environment. The operator θ_L^U preserves those transitions that are allowed in some environment X with $L \subseteq X \subseteq U$. The letters L and U stand for *lower* and *upper*bound. The operator ψ_X places a process in the environment X when a time-out transition occurs; it is inert if any other transition occurs. If $P \xrightarrow{\beta}$ for $\beta \in A \cup \{\tau\}$, then a time-out transition $P \xrightarrow{\mathrm{t}} Q$ cannot occur in an environment that allows β. Thus the transition $P \xrightarrow{\mathrm{t}} Q$ survives only when considering an environments that blocks β, meaning $\beta \notin X \cup \{\tau\}$. Taking the contrapositive, $\beta \in X \cup \{\tau\}$ implies $P \overset{\beta}{\not\longrightarrow}$.

The operator $\theta_\emptyset^U$ features in the forthcoming law (L3), which is a convenient addition to my axiomatisation, although only ψ_X and θ_X $(= \theta_X^X)$ are necessary for completeness.

6 Guarded recursion and finitely branching processes

In many process algebraic treatments, only guarded recursive specifications are allowed.

▶ **Definition 10.** *An occurrence of a variable x in an expression E is* guarded *if x occurs in a subexpression $\alpha.F$ of E, with $\alpha \in Act$. An expression E is* guarded *if all free occurrences of variables in E are guarded. A recursive specification $\mathcal{S}$ is* manifestly guarded *if all expressions $\mathcal{S}_y$ for $y \in V_\mathcal{S}$ are guarded. It is* guarded *if it can be converted into a manifestly guarded recursive specification by repeated substitution of expressions $\mathcal{S}_y$ for variables $y \in V_\mathcal{S}$ occurring in the expressions $\mathcal{S}_z$ for $z \in V_\mathcal{S}$. Let* guarded *$CCSP_\mathrm{t}^\theta$ be the fragment of $CCSP_\mathrm{t}^\theta$ allowing only guarded recursion.*

▶ **Definition 11.** *The set of processes* reachable *from a process P is inductively defined by*
(i) *P is reachable from P, and*
(ii) *if Q is reachable from P and $Q \xrightarrow{\alpha} R$ for some $\alpha \in Act$ then R is reachable from P.*
A process P is finitely branching *if for all $Q \in \mathbb{P}$ reachable from P there are only finitely many processes R such that $Q \xrightarrow{\alpha} R$ for some $\alpha \in Act$. Likewise, P is* countably branching *if there are countably many.*

▶ **Proposition 12** (@)**.** *Each $CCSP_\mathrm{t}^\theta$ process is countably branching.*

▶ **Proposition 13** (@)**.** *Each $CCSP_\mathrm{t}^\theta$ process with guarded recursion is finitely branching.*

▶ **Proposition 14** ([11], @)**.** *Each finitely branching processes in an LTS can be denoted by a $CCSP_\mathrm{t}$ expression with guarded recursion, only using the operations 0, $\alpha.\_$ and $+$.*

▶ **Proposition 15** ([11], @)**.** *Each countably branching processes in an LTS can be denoted by a $CCSP_\mathrm{t}$ expression. Again I only need the $CCSP_\mathrm{t}$ operations 0, $\alpha.\_$, $+$ and recursion.*

7 Congruence

Given an arbitrary process algebra with a collection of operators f, each with an arity n, and a recursion construct $\langle x|\mathcal{S}\rangle$ as in Section 5, let $\mathbb{P}$ and $\mathbb{E}$ be the sets of [closed] valid expressions, and let a substitution instance $E[\rho] \in \mathbb{E}$ for $E \in \mathbb{E}$ and $\rho : Var \rightharpoonup \mathbb{E}$ be defined as in Section 5. Any semantic equivalence $\sim \subseteq \mathbb{P} \times \mathbb{P}$ extends to $\sim \subseteq \mathbb{E} \times \mathbb{E}$ by defining $E \sim F$ iff $E[\rho] \sim F[\rho]$ for each closed substitution $\rho : Var \to \mathbb{P}$. It extends to substitutions $\rho, \nu : Var \rightharpoonup \mathbb{E}$ by $\rho \sim \nu$ iff $\rho(x) \sim \nu(x)$ for each $x \in \mathrm{dom}(\rho)$.

▶ **Definition 16** ([14])**.** A semantic equivalence $\sim$ is a *lean congruence* if $E[\rho] \sim E[\nu]$ for any expression $E \in \mathbb{E}$ and any substitutions ρ and ν with $\rho \sim \nu$. It is a *full congruence* if

$$P_i \sim Q_i \text{ for all } i = 1, ..., n \quad \Rightarrow \quad f(P_1, ..., P_n) \sim f(Q_1, ..., Q_n) \tag{1}$$
$$\mathcal{S}_y \sim \mathcal{S}'_y \text{ for all } y \in V_\mathcal{S} \quad \Rightarrow \quad \langle x|\mathcal{S}\rangle \sim \langle x|\mathcal{S}'\rangle \tag{2}$$

for all functions f of arity n, processes $P_i, Q_i \in \mathbb{P}$, and recursive specifications $\mathcal{S}, \mathcal{S}'$ with $x \in V_\mathcal{S} = V_{\mathcal{S}'}$ and $\langle x|\mathcal{S}\rangle, \langle x|\mathcal{S}'\rangle \in \mathbb{P}$.

Clearly, each full congruence is also a lean congruence, and each lean congruence satisfies (1) above. Both implications are strict, as illustrated in [14].

A main result of the present paper will be that strong reactive bisimilarity is a full congruence for the process algebra $\mathrm{CCSP}_\mathrm{t}^\theta$. To achieve it I need to establish first that strong bisimilarity [29], $\leftrightarroweq$, and initials equivalence [12, Section 16], $=_\mathcal{I}$, are full congruences.

▶ **Definition 17.** *$CCSP_{\mathrm{t}}^{\theta}$ processes P and Q are* initials equivalent*, $P =_{\mathcal{I}} Q$, if $\mathcal{I}(P) = \mathcal{I}(Q)$.*

▶ **Theorem 18.** *Initials equivalence is a full congruence for $CCSP_{\mathrm{t}}^{\theta}$.*

▶ **Definition 19.** *A* strong bisimulation *is a symmetric relation $\mathscr{B}$, such that, when $P \mathrel{\mathscr{B}} Q$,*

- *if $P \xrightarrow{\alpha} P'$ with $\alpha \in Act$ then $Q \xrightarrow{\alpha} Q'$ for some Q' with $P' \mathrel{\mathscr{B}} Q'$.*

Two processes $P, Q \in \mathbb{P}$ are strongly bisimilar*, $P \leftrightarroweq Q$, if $P \mathrel{\mathscr{R}} Q$ for a strong bisimulation $\mathscr{B}$.*

Contrary to reactive bisimilarity, strong bisimilarity treats the time-out action t, as well as the hidden action τ, just like any visible action. In the absence of time-out actions, there is no difference between a strong bisimulation and a time-out bisimulation, so $\leftrightarroweq_r$ and $\leftrightarroweq$ coincide. In general, strong bisimulation is a finer equivalence relation: $P \leftrightarroweq Q \Rightarrow P \leftrightarroweq_r Q \Rightarrow P =_{\mathcal{I}} Q$, and both implications are strict.

▶ **Theorem 20** (@). *Strong bisimilarity is a full congruence for $CCSP_{\mathrm{t}}^{\theta}$.*

8 Strong reactive bisimilarity is a full congruence for $\mathbf{CCSP_{t}^{\theta}}$

The proofs showing that $\leftrightarroweq_r$ is a full congruence for $\mathrm{CCSP}_{\mathrm{t}}^{\theta}$ follow the lines of Milner [29], but are more complicated due to the nature of reactive bisimilarity. A crucial tool is Milner's notion of *bisimilarity up-to*. Even if we would not be interested in the operators θ_L^U and ψ_X, the proof needs to take the operator θ_X $(= \theta_X^X)$ along in order to deal with the other operators. This is a consequence of the occurrence of θ_X in Definition 8.

▶ **Definition 21.** *Given a relation $\sim \subseteq \mathbb{P} \times \mathbb{P}$, a* strong time-out bisimulation up to $\sim$ *is a symmetric relation $\mathscr{B} \subseteq \mathbb{P} \times \mathbb{P}$, such that, for $P \mathrel{\mathscr{B}} Q$,*

- *if $P \xrightarrow{\alpha} P'$ with $\alpha \in A \cup \{\tau\}$, then $\exists Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \sim\mathscr{B}\sim Q'$,*
- *if $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$, then $\exists Q'$ with $Q \xrightarrow{\mathrm{t}} Q'$ and $\theta_X(P') \sim\mathscr{B}\sim \theta_X(Q')$.*

Here $\sim\mathscr{B}\sim := \{(R,T) \mid \exists R', T'.\ R \sim R' \mathrel{\mathscr{B}} T' \sim T\}$.

▶ **Proposition 22.** *If $P \mathrel{\mathscr{B}} Q$ for a strong time-out bisimulation $\mathscr{B}$ up to $\leftrightarroweq$, then $P \leftrightarroweq_r Q$.*

▶ **Theorem 23.** *Strong reactive bisimilarity is a lean congruence for $CCSP_{\mathrm{t}}^{\theta}$. In other words, if $\rho, \nu \colon Var \rightharpoonup \mathbb{E}$ are substitutions with $\rho \leftrightarroweq_r \nu$, then $E[\rho] \leftrightarroweq_r E[\nu]$ for any expression $E \in \mathbb{E}$.*

▶ **Proposition 24.** *If $P \mathrel{\mathscr{B}} Q$ for a strong time-out bisimulation $\mathscr{B}$ up to $\leftrightarroweq_r$, then $P \leftrightarroweq_r Q$.*

▶ **Theorem 25.** *Strong reactive bisimilarity is a full congruence for $CCSP_{\mathrm{t}}^{\theta}$.*

9 The Recursive Specification Principle

For $W \subseteq Var$ a set of variables, a W-tuple of expressions is a function $\vec{E} \in \mathbb{E}^W$. It has a component $\vec{E}(x)$ for each variable $x \in W$. Note that a W-tuple of expressions is nothing else than a substitution. Let id_W be the identity function, given by $id_W(x) = x$ for all $x \in W$. If $G \in \mathbb{E}$ and $\vec{E} \in \mathbb{E}^W$ then $G[\vec{E}]$ denotes the result of simultaneous substitution of $\vec{E}(x)$ for x in G, for all $x \in W$. Likewise, if $\vec{G} \in \mathbb{E}^V$ and $\vec{E} \in \mathbb{E}^W$ then $\vec{G}[\vec{E}] \in \mathbb{E}^V$ denotes the V-tuple with components $G(y)[\vec{E}]$ for $y \in V$. Henceforth, I regard a recursive specification $\mathcal{S}$ as a $V_{\mathcal{S}}$-tuple with components $\mathcal{S}(y) = \mathcal{S}_y$ for $y \in V_{\mathcal{S}}$. If $\vec{E} \in \mathbb{E}^W$ and $\mathcal{S} \in \mathbb{E}^V$, then $\langle \vec{E} | \mathcal{S} \rangle \in \mathbb{E}^W$ is the W-tuple with components $\langle \vec{E}(x) | \mathcal{S} \rangle \in \mathbb{E}^W$ for $x \in W$.

For $\mathcal{S}$ a recursive specification and $\vec{E} \in \mathbb{E}^{V_{\mathcal{S}}}$ a $V_{\mathcal{S}}$-tuple of expressions, $\vec{E} \leftrightarroweq_r \mathcal{S}[\vec{E}]$ states that $\vec{E}$ is a *solution* of $\mathcal{S}$, up to strong reactive bisimilarity. The tuple $\langle id_{V_{\mathcal{S}}} | \mathcal{S} \rangle \in \mathbb{E}^{V_{\mathcal{S}}}$ is called the *default solution*.

Table 2 A complete axiomatisation of strong bisimilarity on guarded CCSP$_\mathrm{t}$.

$x+(y+z) = (x+y)+z$	$\tau_I(x+y) = \tau_I(x)+\tau_I(y)$		$\mathcal{R}(x+y) = \mathcal{R}(x)+\mathcal{R}(y)$
$x+y = y+x$	$\tau_I(\alpha.x) = \alpha.\tau_I(x)$	if $\alpha\notin I$	$\mathcal{R}(\tau.x) = \tau.\mathcal{R}(x)$
$x+x = x$	$\tau_I(\alpha.x) = \tau.\tau_I(x)$	if $\alpha\in I$	$\mathcal{R}(\mathrm{t}.x) = \mathrm{t}.\mathcal{R}(x)$
$x+0 = 0$	$\langle x \mid \mathcal{S}\rangle = \langle \mathcal{S}_x \mid \mathcal{S}\rangle$		$\mathcal{R}(a.x) = \sum_{\{b \mid (a,b)\in\mathcal{R}\}} b.\mathcal{R}(x)$
If $P=\sum_{i\in I}\alpha_i.P_i$ and $Q=\sum_{j\in J}\beta_j.Q_j$ then $P \mathbin{\Vert_S} Q = \sum_{i\in I,\ \alpha_i\notin S}(\alpha_i.P_i \mathbin{\Vert_S} Q) + \sum_{j\in J,\ \beta_j\notin S}(P \mathbin{\Vert_S} \beta_j.Q_j) + \sum_{i\in I,\ j\in J,\ \alpha_i=\beta_j\in S}\alpha_i.(P_i \mathbin{\Vert_S} Q_j)$			
Recursive Specification Principle (RSP)	$\mathcal{S} \Rightarrow x = \langle x \mid \mathcal{S}\rangle$		($\mathcal{S}$ guarded)

In [1, 8] two requirements occur for process algebras with recursion. The *recursive definition principle* (RDP) says that each recursive specification must have a solution, and the *recursive specification principle* (RSP) says that guarded recursive specifications have at most one solution. When dealing with process algebras where the meaning of a closed expression is a semantic equivalence class of processes, these principles become requirements on the semantic equivalence employed.

▶ **Proposition 26.** *Let $\mathcal{S}$ be a guarded recursive specification, and $x\in V_\mathcal{S}$. Then $\langle x|\mathcal{S}\rangle \mathrel{\underline{\leftrightarrow}_r} \langle \mathcal{S}_x|\mathcal{S}\rangle$.*

Proposition 26 says that the recursive definition principle holds for strong reactive bisimulation semantics. The "default solution" of a recursive specification is in fact a solution. Note that the conclusion of Proposition 26 can be restated as $\langle id_{V_\mathcal{S}}|\mathcal{S}\rangle \mathrel{\underline{\leftrightarrow}_r} \langle \mathcal{S}|\mathcal{S}\rangle$, and that $\mathcal{S}[\langle id_{V_\mathcal{S}}|\mathcal{S}\rangle] = \langle \mathcal{S}|\mathcal{S}\rangle$.

The following theorem establishes the recursive specification principle for strong reactive bisimulation semantics.

▶ **Theorem 27.** *Let $\mathcal{S}$ be a guarded recursive specification. If $\vec{E} \mathrel{\underline{\leftrightarrow}_r} \mathcal{S}[\vec{E}]$ and $\vec{F} \mathrel{\underline{\leftrightarrow}_r} \mathcal{S}[\vec{F}]$ with $\vec{E},\vec{F}\in\mathbb{E}^{V_\mathcal{S}}$, then $\vec{E} \mathrel{\underline{\leftrightarrow}_r} \vec{F}$.*

10 A complete axiomatisation for processes with guarded recursion

The well-known axioms of Table 2 are *sound* for strong bisimilarity, meaning that writing $\underline{\leftrightarrow}$ for $=$, and substituting arbitrary expressions for the free variables x,y,z, or the meta-variables P_i and Q_j, turns them into true statements. In these axioms α,β range over *Act* and a,b over A. All axioms involving variables are equations. The axiom involving P and Q is a template that stands for a family of equations, one for each fitting choice of P and Q. This is the CCSP$_\mathrm{t}$ version of the *expansion law* from [29]. The axiom $\langle x|\mathcal{S}\rangle = \langle \mathcal{S}_x|\mathcal{S}\rangle$ says that recursively defined processes $\langle x|\mathcal{S}\rangle$ satisfy their set of defining equations $\mathcal{S}$. As discussed in the previous section, this entails that each recursive specification has a solution. The axiom RSP [1, 8] is a conditional equation, with as antecedents the equations of a guarded recursive specification $\mathcal{S}$. It says that the x-component of any solution of $\mathcal{S}$ – a vector of processes substituted for the variables $V_\mathcal{S}$ – equals $\langle x|\mathcal{S}\rangle$. In other words, each solution of $\mathcal{S}$ equals the default solution. This is a compact way of saying that solutions of guarded recursive specifications are unique.

▶ **Theorem 28.** *For CCSP$_\mathrm{t}$ processes $P,Q\in\mathbb{P}$ with guarded recursion, one has $P \mathrel{\underline{\leftrightarrow}} Q$, that is, P and Q are strongly bisimilar, iff $P=Q$ is derivable from the axioms of Table 2.*

Table 3 A complete axiomatisation of strong bisimilarity on guarded CCSP$_{\mathrm{t}}^{\theta}$.

$\theta_L^U(\sum_{i\in I}\alpha_i.x_i) = \sum_{i\in I}\alpha_i.x_i$	$(\alpha_i \notin L\cup\{\tau\}$ for all $i\in I)$
$\theta_L^U(x+\alpha.y+\beta.z) = \theta_L^U(x+\alpha.y)$	$(\alpha\in L\cup\{\tau\} \wedge \beta\notin U\cup\{\tau\})$
$\theta_L^U(x+\alpha.y+\beta.z) = \theta_L^U(x+\alpha.y)+\theta_L^U(\beta.z)$	$(\alpha\in L\cup\{\tau\} \wedge \beta\in U\cup\{\tau\})$
$\theta_L^U(\beta.x) = \beta.x$	$(\beta\neq\tau)$
$\theta_L^U(\tau.x) = \tau.\theta_L^U(x)$	
$\psi_X(x+\alpha.z) = \psi_X(x)+\alpha.z$	$(\alpha\notin X\cup\{\tau,\mathrm{t}\})$
$\psi_X(x+\alpha.y+\mathrm{t}.z) = \psi_X(x+\alpha.y)$	$(\alpha\in X\cup\{\tau\})$
$\psi_X(x+\alpha.y+\beta.z) = \psi_X(x+\alpha.y)+\beta.z$	$(\alpha,\beta\in X\cup\{\tau\})$
$\psi_X(\alpha.x) = \alpha.x$	$(\alpha\neq\mathrm{t})$
$\psi_X(\sum_{j\in I}\mathrm{t}.y_i) = \sum_{j\in I}\mathrm{t}.\theta_X(y_j)$	

In this theorem, "if", the *soundness* of the axiomatisation of Table 2, is an immediate consequence of the soundness of the individual axioms. "Only if" states the *completeness* of the axiomatisation.

A crucial tool in its proof is the simple observation that the axioms from the first box of Table 2 allow any CCSP$_{\mathrm{t}}$ process with guarded recursion to be brought in the form $\sum_{i\in I}\alpha_i.P_i$ – a *head normal form*. Using this, the rest of the proof is a standard argument employing RSP, independent of the choice of the specific process algebra. It can be found in [29], [1], [8] and many other places. However, in the literature this completeness theorem was always stated and proved for a small fragment of the process algebra, allowing only guarded recursive specifications with a finite number of equations, and whose right-hand sides $\mathcal{S}_y$ involve only the basic operators inaction, action prefixing and choice. Since the set of true statements $P \mathrel{\underline{\leftrightarrow}} Q$, with P and Q processes in a process algebra like guarded CCSP$_{\mathrm{t}}$, is well-known to be undecidable, and even not recursively enumerable, is was widely believed that no sound and complete axiomatisation of strong bisimilarity could exist. Only in March 2017, Kees Middelburg observed (in the setting of the process algebra ACP [1, 8]) that the standard proof applies almost verbatim to arbitrary processes with guarded recursion, although one has to be a bit careful in dealing with the infinite nature of recursive specifications. The argument has been carefully documented in [19], in the setting of the process algebra ACP. This result does not contradict the non-enumerability of the set of true statements $P \mathrel{\underline{\leftrightarrow}} Q$, due to the fact that RSP is a proof rule with infinitely many premises.

Table 3 extends Table 2 with axioms for the auxiliary operators θ_L^U and ψ_X. With Table 1 it is straightforward to check the soundness of these axioms. The fourth axiom, for instance, follows from the second or third rule for θ_L^U in Table 1, depending on whether $\beta\in L\cup\{\mathrm{t}\}$. Moreover, a straightforward induction shows that these axioms suffice to convert each process into head normal form. Using this, we immediately obtain:

▶ **Theorem 29.** *For CCSP$_{\mathrm{t}}^{\theta}$ processes $P,Q\in\mathbb{P}$ with guarded recursion, one has $P \mathrel{\underline{\leftrightarrow}} Q$ iff $P=Q$ is derivable from the axioms of Tables 2 and 3.*

A law that turns out to be particularly useful in verifications modulo strong reactive bisimilarity is

$$\boxed{\theta_K^V(\theta_L^U(x)) \mathrel{\underline{\leftrightarrow}} \theta_{K\cup L}^{V\cap U}(x) \qquad \begin{array}{l}\text{provided } U=V \text{ or } K=L \text{ or}\\ K\subseteq L\subseteq U\subseteq V \text{ or } L\subseteq K\subseteq V\subseteq U\end{array} \qquad \text{(L1)}}\;.$$

Table 4 A complete axiomatisation of strong reactive bisimilarity on guarded CCSP$_\mathrm{t}^\theta$.

$$\boxed{\frac{\psi_X(x)=\psi_X(y) \text{ for all } X\subseteq A}{x=y} \qquad \text{(RA)}}$$

Note that the right-hand side only exists if $(K\cup L)\subseteq(V\cap U)$. This law is sound for strong bisimilarity, as demonstrated by the following proposition. Yet it is not needed to add it to Table 3, as all its closed instances are derivable. In fact, this is a consequence of the above completeness theorem.

▶ **Proposition 30.** *$\theta_K^V(\theta_L^U(P)) \leftrightarrow \theta_{K\cup L}^{V\cap U}(P)$, provided $(K\cup L)\subseteq(V\cap U)$ and either $U=V$ or $K=L$ or $K\subseteq L\subseteq U\subseteq V$ or $L\subseteq K\subseteq V\subseteq U$.*

To obtain a sound and complete axiomatisation of strong reactive bisimilarity for CCSP$_\mathrm{t}^\theta$ with guarded recursion, one needs to combine the axioms of Tables 2, 3 and 4. These axioms are useful only in combination with the full congruence property of strong reactive bisimilarity, Theorem 25. This is what allows us to apply these axioms within subexpressions of a given expression. Since $\leftrightarrow\ \subseteq\ \leftrightarrow_r$, the soundness of all equational axioms for strong reactive bisimilarity follows from their soundness for strong bisimilarity. The soundness of RSP has been established as Theorem 27. The soundness of (RA), the *reactive approximation axiom*, is contributed by the following proposition.

▶ **Proposition 31.** *Let $P,Q\in\mathbb{P}$. If $\psi_X(P)\leftrightarrow_r\psi_X(Q)$ for all $X\subseteq A$, then $P\leftrightarrow_r Q$.*

The completeness of this axiomatisation is documented in the technical report accompanying this paper. Instead of the classic technique of *merging guarded recursive equations* [27, 28, 35, 9, 26], which in essence proves two bisimilar systems P and Q equivalent by equating both to an intermediate variant that is essentially a *product* of P and Q, it employs the novel method of *canonical solutions* [25], which equates both P and Q to a canonical representative within the bisimulation equivalence class of P and Q – one that has only one reachable state for each bisimulation equivalence class of states of P and Q.

Moreover, my proof employs the axiom of choice [36] in defining the transition relation on my canonical representative, in order to keep this process finitely branching.

At first sight it appears that axiom (RA) is not very handy, as the number of premises to verify is exponential in the size of the alphabet A of visible actions. In case A is infinite, there are even uncountably many premises. However, in practical verifications this is hardly an issue, as one uses a partition of the premises into a small number of equivalence classes, each of which requires only one common proof. This technique will be illustrated on three examples below. Furthermore, one could calculate the set of visible actions $\mathcal{J}(P)$ of a process P that can be encountered as initial actions after one t-transition followed by a sequence of τ-transitions. For large classes of processes, $\mathcal{J}(P)$ will be a finite set. Now axiom (RA) can be modified by changing $X\subseteq A$ into $X\subseteq\mathcal{J}(P)\cup\mathcal{J}(Q)$. This preserves the soundness of the axiom, because only the actions in $\mathcal{J}(P)$ play any rôle in evaluating $\psi_X(P)$.

A crucial property of strong reactive bisimilarity was mentioned in the introduction:

$$\boxed{\tau.P+\mathrm{t}.Q=\tau.P \qquad \text{(L2)}}\ .$$

It is an immediate consequence of (RA), since $\psi_X(\tau.P+\mathrm{t}.Q)=\psi_X(\tau.P)$ for any $X\subseteq A$, by Table 3. Another useful law in verifications modulo strong reactive bisimilarity is

$$\boxed{\textstyle\sum_{i\in I}a_i.x_i+\mathrm{t}.y=\sum_{i\in I}a_i.x_i+\mathrm{t}.\theta_\emptyset^{A\setminus In}(y), \text{ where } In=\{a_i\mid i\in I\}. \qquad \text{(L3)}}$$

Its soundness is intuitively obvious: the t-transition to y will be taken only in an environment X with $X \cap In = \emptyset$. Hence one can just as well restrict the behaviour of y to those transitions that are allowed in one such environment. This law was one of the prime reasons for extending the family of operators θ_X $(=\theta_X^X)$, which were needed to establish the key theorems of this paper, to the larger family θ_L^U. Law (L3) for finite I is effortlessly derivable from its simple instance

$$\boxed{a.x + \mathrm{t}.y = a.x + \mathrm{t}.\theta_\emptyset^{A\setminus\{a\}}(y). \qquad \text{(L3}'\text{)}}$$

in combination with (L1). I now show how to derive (L3) from (RA). For this proof I need to partition the set of premises of (RA) in only two equivalence classes.

First let $X \cap In \neq \emptyset$. Then $\psi_X(\sum_{i\in I} a_i.x_i + \mathrm{t}.y) = \sum_{i\in I} a_i.x_i = \psi_X(\sum_{i\in I} a_i.x_i + \mathrm{t}.\theta_\emptyset^{A\setminus In}(y))$.

Next let $X \cap In = \emptyset$. Then

$$\begin{aligned}\psi_X(\textstyle\sum_{i\in I} a_i.x_i + \mathrm{t}.y) &= \textstyle\sum_{i\in I} a_i.x_i + \mathrm{t}.\theta_X(y)\\ &= \textstyle\sum_{i\in I} a_i.x_i + \mathrm{t}.\theta_X(\theta_\emptyset^{A\setminus In}(y))\\ &= \psi_X(\textstyle\sum_{i\in I} a_i.x_i + \mathrm{t}.\theta_\emptyset^{A\setminus In}(y))\ ,\end{aligned}$$

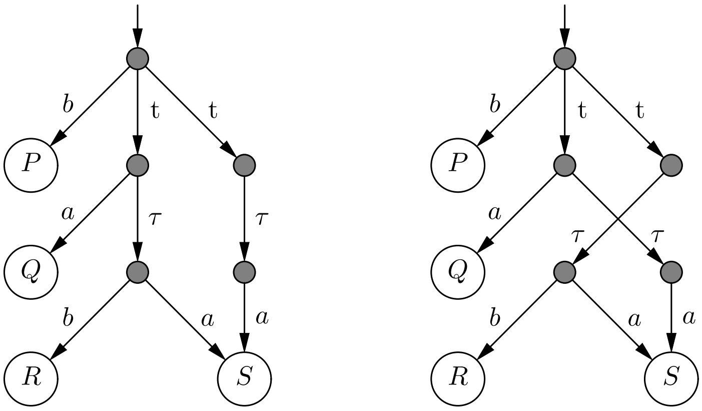
where the second step is an application of (L1).

As an application of (L3′) one obtains the law that was justified in the introduction:

$$\begin{aligned}a.P + \mathrm{t}.(Q + \tau.R + a.S) &= a.P + \mathrm{t}.\theta_\emptyset^{A\setminus\{a\}}(Q + \tau.R + a.S)\\ &= a.P + \mathrm{t}.\theta_\emptyset^{A\setminus\{a\}}(Q + \tau.R)\\ &= a.P + \mathrm{t}.(Q + \tau.R)\ .\end{aligned}$$

As a third illustration of the use of (RA) I derive an equational law that does not follow from (L1), (L2) and (L3), namely

$$b.P + \mathrm{t}.(a.Q + \tau.(b.R + a.S)) + \mathrm{t}.\tau.a.S = b.P + \mathrm{t}.(a.Q + \tau.a.S) + \mathrm{t}.\tau.(b.R + a.S)$$



These system are surely not strongly bisimilar. Moreover, (L3) does not help in proving them equivalent, as applying $\theta_\emptyset^{A\setminus\{b\}}$ to any of the four targets of a t-transition does not kill any of the transitions of those processes. In particular, $\theta_\emptyset^{A\setminus\{b\}}(b.R + a.S) = b.R + a.S$. To derive this law from (RA), I partition $\mathscr{P}(A)$ into three equivalence classes.

First let $b \in X$. Then

$$\begin{aligned}&\psi_X(b.P + \mathrm{t}.(a.Q + \tau.(b.R + a.S)) + \mathrm{t}.\tau.a.S)\\ &= b.P\\ &= \psi_X(b.P + \mathrm{t}.(a.Q + \tau.a.S) + \mathrm{t}.\tau.(b.R + a.S)).\end{aligned}$$

Next let $b \notin X$ and $a \in X$. Then

$$\begin{aligned}
&\psi_X\big(b.P + \mathrm{t}.(a.Q + \tau.(b.R + a.S)) + \mathrm{t}.\tau.a.S\big)\\
&= b.P + \mathrm{t}.\theta_X\big(a.Q + \tau.(b.R + a.S)\big) + \mathrm{t}.\theta_X\big(\tau.a.S\big)\\
&= b.P + \mathrm{t}.(a.Q + \tau.\theta_X(b.R + a.S)) + \mathrm{t}.\tau.\theta_X(a.S)\\
&= b.P + \mathrm{t}.(a.Q + \tau.a.S) + \mathrm{t}.\tau.a.S\\
&= b.P + \mathrm{t}.(a.Q + \tau.\theta_X(a.S)) + \mathrm{t}.\tau.\theta_X\big(b.R + a.S\big)\\
&= b.P + \mathrm{t}.\theta_X\big(a.Q + \tau.a.S\big) + \mathrm{t}.\theta_X\big(\tau.(b.R + a.S)\big)\\
&= \psi_X\big(b.P + \mathrm{t}.(a.Q + \tau.a.S) + \mathrm{t}.\tau.(b.R + a.S)\big)\ .
\end{aligned}$$

Finally let $a, b \notin X$. Then

$$\begin{aligned}
&\psi_X\big(b.P + \mathrm{t}.(a.Q + \tau.(b.R + a.S)) + \mathrm{t}.\tau.a.S\big)\\
&= b.P + \mathrm{t}.\theta_X\big(a.Q + \tau.(b.R + a.S)\big) + \mathrm{t}.\theta_X\big(\tau.a.S\big)\\
&= b.P + \mathrm{t}.\tau.\theta_X(b.R + a.S) + \mathrm{t}.\tau.\theta_X(a.S)\\
&= b.P + \mathrm{t}.\tau.(b.R + a.S) + \mathrm{t}.\tau.a.S\\
&= b.P + \mathrm{t}.\tau.a.S + \mathrm{t}.\tau.(b.R + a.S)\\
&= b.P + \mathrm{t}.\tau.\theta_X(a.S) + \mathrm{t}.\tau.\theta_X\big(b.R + a.S\big)\\
&= b.P + \mathrm{t}.\theta_X\big(a.Q + \tau.a.S\big) + \mathrm{t}.\theta_X\big(\tau.(b.R + a.S)\big)\\
&= \psi_X\big(b.P + \mathrm{t}.(a.Q + \tau.a.S) + \mathrm{t}.\tau.(b.R + a.S)\big)\ .
\end{aligned}$$

In words, the 4 processes that are targets of t-transitions always run in an environment that blocks b. In an environment that allows a, the branch $b.R$ disappears, so that the left branch of the first process can matched with the left branch of the second process, and similarly for the two right branches. In an environment that blocks a, this matching won't fly, as the branch $b.R$ now survives. However, the branches $a.Q$ will disappear, so that the left branch of the first process can be matched with the right branch of the second, and vice versa.

11 Concluding remarks

This paper laid the foundations of the proper analogue of strong bisimulation semantics for a process algebra with time-outs. This makes it possible to specify systems in this setting and verify their correctness properties. The addition of time-outs comes with considerable gains in expressive power. An illustration of this is mutual exclusion.

As shown in [17], it is fundamentally impossible to correctly specify mutual exclusion protocols in standard process algebras, such as CCS [29], CSP [4, 24], ACP [1, 8] or CCSP, unless the correctness of the specified protocol hinges on a fairness assumption. The latter, in the view of [17], does not provide an adequate solution, as fairness assumptions are in many situations unwarranted and lead to false conclusions. In [7] a correct process-algebraic rendering of mutual exclusion is given, but only after making two important modifications to standard process algebra. The first involves making a justness assumption. Here *justness* [18] is an alternative to fairness, in some sense a much weaker form of fairness – meaning weaker than weak fairness. Unlike (strong or weak) fairness, its use typically is warranted and does not lead to false conclusions. The second modification is the addition of a new construct – *signals* – to CCS, or any other standard process algebra. Interestingly, both modifications are necessary; just using justness, or just adding signals, is insufficient. Bouwman [2, 3] points out that since the justness requirement was fairly new, and needed to be carefully defined to describe its interaction with signals anyway, it is possible to specify mutual exclusion without adding signals to the language at all, instead reformulating the justness requirement in such a way that it effectively turns some actions into signals. Yet justness is essential in all these approaches. This may be seen as problematic, because large parts of the foundations of process algebra are incompatible with justness, and hence need to be thoroughly reformulated in a justness-friendly way. This is pointed out in [15].

The addition of time-outs to standard process algebra makes it possible to specify mutual exclusion without assuming justness! Instead, one should make the assumption called *progress* in [18], which is weaker than justness, uncontroversial, unproblematic, and made (explicitly or implicitly) in virtually all papers dealing with issues like mutual exclusion. I will defer substantiation of this claim to a future occasion.

Besides applications to protocol verification, future work includes adapting the work done here to a form of reactive bisimilarity that abstracts from hidden actions, that is, to provide a counterpart for process algebras with time-outs of, for instance, branching bisimilarity [20], weak bisimilarity [29] or coupled similarity [32, 10]. Other topics worth exploring are the extension to probabilistic processes, and especially the relations with timed process algebras. Davies & Schneider in [5], for instance, added a construct with a quantified time-out to the process algebra CSP [4, 24], elaborating the timed model of CSP presented by Reed & Roscoe in [33].

References

1 J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990. doi:10.1017/CBO9780511624193.

2 M.S. Bouwman. Liveness analysis in process algebra: simpler techniques to model mutex algorithms. Technical report, Eindhoven University of Technology, 2018. URL: http://www.win.tue.nl/~timw/downloads/bouwman_seminar.pdf.

3 M.S. Bouwman, B. Luttik, and T.A.C. Willemse. Off-the-shelf automated analysis of liveness properties for just paths. *Acta Informatica*, 57(3-5):551–590, 2020. doi:10.1007/s00236-020-00371-w.

4 S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984. doi:10.1145/828.833.

5 J. Davies and S. Schneider. Recursion induction for real-time processes. *Formal Aspects of Computing*, 5(6):530–553, 1993. doi:10.1007/BF01211248.

6 R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984. doi:10.1016/0304-3975(84)90113-0.

7 V. Dyseryn, R.J. van Glabbeek, and P. Höfner. Analysing mutual exclusion using process algebra with signals. In K. Peters and S. Tini, editors, Proceedings Combined 24th International Workshop on *Expressiveness in Concurrency* and 14th Workshop on *Structural Operational Semantics,* Berlin, Germany, 4th September 2017, volume 255 of *Electronic Proceedings in Theoretical Computer Science*, pages 18–34. Open Publishing Association, 2017. doi:10.4204/EPTCS.255.2.

8 W. J. Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science, An EATCS Series. Springer, 2000. doi:10.1007/978-3-662-04293-9.

9 R.J. van Glabbeek. A complete axiomatization for branching bisimulation congruence of finite-state behaviours. In A.M. Borzyszkowski and S. Sokołowski, editors, Proceedings 18^{th} International Symposium on *Mathematical Foundations of Computer Science,* MFCS '93, Gdansk, Poland, August/September 1993, volume 711 of LNCS, pages 473–484. Springer, 1993. doi:10.1007/3-540-57182-5_39.

10 R.J. van Glabbeek. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves. In E. Best, editor, Proceedings *CONCUR'93,* 4^{th} International Conference on *Concurrency Theory,* Hildesheim, Germany, August 1993, volume 715 of LNCS, pages 66–81. Springer, 1993. doi:10.1007/3-540-57208-2_6.

11 R.J. van Glabbeek. On the expressiveness of ACP (extended abstract). In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, Proceedings First Workshop on the *Algebra of Communicating Processes,* ACP'94, Utrecht, The Netherlands, May 1994, Workshops in Computing, pages 188–217. Springer, 1994. doi:10.1007/978-1-4471-2120-6_8.

12 R.J. van Glabbeek. The linear time – branching time spectrum I; the semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier, 2001. doi:10.1016/B978-044482830-9/50019-9.

13 R.J. van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming*, 60–61:229–258, 2004. doi:10.1016/j.jlap.2004.03.007.

14 R.J. van Glabbeek. Lean and full congruence formats for recursion. In Proceedings 32^{nd} Annual ACM/IEEE Symposium on *Logic in Computer Science,* LICS'17, Reykjavik, Iceland, June 2017. IEEE Computer Society Press, 2017. doi:10.1109/LICS.2017.8005142.

15 R.J. van Glabbeek. Ensuring liveness properties of distributed systems: Open problems. *Journal of Logical and Algebraic Methods in Programming*, 109, 2019. doi:10.1016/j.jlamp.2019.100480.

16 R.J. van Glabbeek. Failure trace semantics for a process algebra with time-outs, 2020. arXiv:2002.10814.

17 R.J. van Glabbeek and P. Höfner. CCS: it's not fair! Fair schedulers cannot be implemented in CCS-like languages even under progress and certain fairness assumptions. *Acta Informatica*, 52(2-3):175–205, 2015. doi:10.1007/s00236-015-0221-6.

18 R.J. van Glabbeek and P. Höfner. Progress, justness and fairness. *ACM Computing Surveys*, 52(4), August 2019. doi:10.1145/3329125.

19 R.J. van Glabbeek and C.A. Middelburg. On infinite guarded recursive specifications in process algebra, 2020. arXiv:2005.00746.

20 R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996. doi:10.1145/233551.233556.

21 J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993. doi:10.1016/0304-3975(93)90111-6.

22 J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992. doi:10.1016/0890-5401(92)90013-6.

23 M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985. doi:10.1145/2455.2460.

24 C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, Englewood Cliffs, 1985.

25 X. Liu and T.Yu. Canonical solutions to recursive equations and completeness of equational axiomatisations. In I. Konnov and L. Kovacs, editors, Proceedings 31st International Conference on *Concurrency Theory* (CONCUR 2020), Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. To appear.

26 M. Lohrey, P.R. D'Argenio, and H. Hermanns. Axiomatising divergence. *Information and Computation*, 203(2):115–144, 2005. doi:10.1016/j.ic.2005.05.007.

27 R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984. doi:10.1016/0022-0000(84)90023-0.

28 R. Milner. A complete axiomatisation for observational congruence of finite-state behaviors. *Information and Computation*, 81(2):227–247, 1989. doi:10.1016/0890-5401(89)90070-9.

29 R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 19, pages 1201–1242. Elsevier Science Publishers B.V. (North-Holland), 1990. Alternatively see *Communication and Concurrency*, Prentice-Hall, Englewood Cliffs, 1989, of which an earlier version appeared as *A Calculus of Communicating Systems*, LNCS 92, Springer, 1980, doi:10.1007/3-540-10235-3.

30 E.-R. Olderog. Operational Petri net semantics for CCSP. In G. Rozenberg, editor, *Advances in Petri Nets 1987*, volume 266 of LNCS, pages 196–223. Springer, 1987. doi:10.1007/3-540-18086-9_27.

31 E.-R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23:9–66, 1986. doi:10.1007/BF00268075.

32 J. Parrow and P. Sjödin. Multiway synchronization verified with coupled simulation. In W.R. Cleaveland, editor, *Proceedings CONCUR 92,* Stony Brook, NY, USA, volume 630 of LNCS, pages 518–533. Springer, 1992. doi:10.1007/BFb0084813.

33 G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988. doi:10.1016/0304-3975(88)90030-8.

34 F.W. Vaandrager. Expressiveness results for process algebras. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX Workshop on Semantics: Foundations and Applications,* Beekbergen, The Netherlands, June 1992, volume 666 of LNCS, pages 609–638. Springer, 1993. doi:10.1007/3-540-56596-5_49.

35 D.J. Walker. Bisimulation and divergence. *Information and Computation*, 85(2):202–241, 1990. doi:10.1016/0890-5401(90)90048-M.

36 Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre I. *Mathematische Annalen*, 65(2):261–281, 1908. doi:10.1007/bf01449999.

A Proofs

▶ **Proposition 2.** $\mathrel{\underline{\leftrightarrow}}_r$ and $\mathrel{\underline{\leftrightarrow}}_r^X$ for $X \subseteq A$ are equivalence relations.

Proof. $\mathrel{\underline{\leftrightarrow}}_r^X$, $\mathrel{\underline{\leftrightarrow}}_r$ are reflexive, as $\{(P,X,P),(P,P) \mid P \in \mathbb{P} \wedge X \subseteq A\}$ is a strong reactive bisimulation.
$\mathrel{\underline{\leftrightarrow}}_r^X$ and $\mathrel{\underline{\leftrightarrow}}_r$ are symmetric, since strong reactive bisimulations are symmetric by definition.
$\mathrel{\underline{\leftrightarrow}}_r^X$ and $\mathrel{\underline{\leftrightarrow}}_r$ are transitive, for if $\mathscr{R}$ and $\mathscr{S}$ are strong reactive bisimulations, then so is
$\mathscr{R};\mathscr{S} = \{(P,X,R) \mid (P,X,Q) \in \mathscr{R} \wedge (Q,X,R) \in \mathscr{S}\} \cup \{(P,R) \mid (P,Q) \in \mathscr{R} \wedge (Q,R) \in \mathscr{S}\}$. ◀

▶ **Proposition 4.** $P \mathrel{\underline{\leftrightarrow}}_r Q$ iff there exists a gsrb $\mathscr{R}$ with $(P,Q) \in \mathscr{R}$.
Likewise, $P \mathrel{\underline{\leftrightarrow}}_r^X Q$ iff there exists a gsrb $\mathscr{R}$ with $(P,X,Q) \in \mathscr{R}$.

Proof. Clearly, each strong reactive bisimulation satisfies the five clauses of Definition 3 and thus is a gsrb. In the other direction, given a gsrb $\mathscr{B}$, let

$$\begin{aligned}\mathscr{R} := \mathscr{B} &\cup \{(P,X,Q) \mid (P,Q) \in \mathscr{B} \wedge X \subseteq A\}\\ &\cup \{(P,Q),(P,X,Q) \mid (P,Y,Q) \in \mathscr{B} \wedge \mathcal{I}(P) \cap (Y \cup \{\tau\}) = \emptyset\}.\end{aligned}$$

It is straightforward to check that $\mathscr{R}$ satisfies the six clauses of Definition 1. ◀

▶ **Theorem 7.** Let $P,Q \in \mathbb{P}$ and $X \subseteq A$. Then $P \mathrel{\underline{\leftrightarrow}}_r Q \quad\Leftrightarrow\quad \forall\varphi \in \mathbb{O}.\ (P \models \varphi \Leftrightarrow Q \models \varphi)$
and $P \mathrel{\underline{\leftrightarrow}}_r^X Q \quad\Leftrightarrow\quad \forall\varphi \in \mathbb{O}.\ (P \models_X \varphi \Leftrightarrow Q \models_X \varphi)$.

Proof. "$\Rightarrow$": I prove by simultaneous structural induction on $\varphi \in \mathbb{O}$ that, for all $P,Q \in \mathbb{P}$ and $X \subseteq A$, $P \mathrel{\underline{\leftrightarrow}}_r Q \wedge P \models \varphi \;\Rightarrow\; Q \models \varphi$ and $P \mathrel{\underline{\leftrightarrow}}_r^X Q \wedge P \models_X \varphi \;\Rightarrow\; Q \models_X \varphi$. The converse implications ($Q \models \varphi \Rightarrow P \models \varphi$ and $Q \models_X \varphi \Rightarrow P \models_X \varphi$) follow by symmetry.

- Let $P \mathrel{\underline{\leftrightarrow}}_r Q \wedge P \models \varphi$.
  - Let $\varphi = \bigwedge_{i\in I} \varphi_i$. Then $P \models \varphi_i$ for all $i \in I$. By induction $Q \models \varphi_i$ for all i, so $q \models \bigwedge_{i\in I} \varphi_i$.
  - Let $\varphi = \neg\psi$. Then $P \not\models \psi$. By induction $Q \not\models \psi$, so $Q \models \neg\psi$.
  - Let $\varphi = \langle\alpha\rangle\psi$ with $\alpha \in A \cup \{\tau\}$. Then $P \xrightarrow{\alpha} P'$ for some P' with $P' \models \psi$. By Definition 3, $Q \xrightarrow{\alpha} Q'$ for some Q' with $P' \mathrel{\underline{\leftrightarrow}}_r Q'$. So by induction $Q' \models \psi$, and thus $Q \models \langle\alpha\rangle\psi$.

  - Let $\varphi = \langle X\rangle\psi$ for some $X \subseteq A$. Then $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$ for some P' with $P' \models_X \psi$. By Definition 3, $Q \xrightarrow{\mathrm{t}} Q'$ for some Q' with $P' \underline{\leftrightarrow}_r^X Q'$. So by induction $Q' \models_X \psi$. Moreover, $\mathcal{I}(Q) = \mathcal{I}(P)$, as $P \underline{\leftrightarrow}_r Q$, so $\mathcal{I}(Q) \cap (X \cup \{\tau\}) = \emptyset$. Thus $Q \models \langle X\rangle\psi$.

- Let $P \underline{\leftrightarrow}_r^X Q \wedge P \models_X \varphi$.
  - Let $\varphi = \bigwedge_{i\in I} \varphi_i$, and $P \models_X \varphi_i$ for all $i \in I$. By induction $Q \models_X \varphi_i$ for all $i \in I$, so $q \models_X \bigwedge_{i\in I} \varphi_i$.
  - Let $\varphi = \neg\psi$, and $P \not\models_X \psi$. By induction $Q \not\models_X \psi$, so $Q \models_X \neg\psi$.
  - Let $\varphi = \langle a\rangle\psi$ with $a \in X$ and $P \xrightarrow{a} P'$ for some P' with $P' \models \psi$. By Definition 1, $Q \xrightarrow{a} Q'$ for some Q' with $P' \underline{\leftrightarrow}_r Q'$. By induction $Q' \models \psi$, so $Q \models_X \langle a\rangle\psi$.
  - Let $\varphi = \langle\tau\rangle\psi$, and $P \xrightarrow{\tau} P'$ for some P' with $P' \models_X \psi$. By Definition 1, $Q \xrightarrow{\tau} Q'$ for some Q' with $P' \underline{\leftrightarrow}_r^X Q'$. By induction $Q' \models_X \psi$, so $Q \models_X \langle\tau\rangle\psi$.
  - Let $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \models \varphi$. By the fifth clause of Definition 1, $P \underline{\leftrightarrow}_r Q$. Hence, by the previous case in this proof, $Q \models \varphi$. Moreover, $\mathcal{I}(Q) \cap (X \cup \{\tau\}) = \mathcal{I}(P) \cap (X \cup \{\tau\})$, since $P \underline{\leftrightarrow}_r^X Q$. Thus $Q \models_X \varphi$.

"$\Leftarrow$": Write $P \equiv Q$ for $\forall\varphi \in \mathbb{O}.\ (P \models \varphi \Leftrightarrow Q \models \varphi)$, and $P \equiv_X Q$ for $\forall\varphi \in \mathbb{O}.\ (P \models_X \varphi \Leftrightarrow Q \models_X \varphi)$. I show that the family of relations $\equiv$, $\equiv_X$ for $X \subseteq A$ constitutes a gsrb.

- Suppose $P \equiv Q$ and $P \xrightarrow{\alpha} P'$ with $\alpha \in A \cup \{\tau\}$. Let $\mathcal{Q}^\dagger := \{Q^\dagger \in \mathbb{P} \mid Q \xrightarrow{\alpha} Q^\dagger \wedge P' \not\equiv Q^\dagger\}$. For each $Q^\dagger \in \mathcal{Q}^\dagger$, let $\varphi_{Q^\dagger} \in \mathbb{O}$ be a formula such that $P' \models \varphi_{Q^\dagger}$ and $Q^\dagger \not\models \varphi_{Q^\dagger}$. (Such a formula always exists because $\mathbb{O}$ is closed under negation.)

  Define $\varphi := \bigwedge_{Q^\dagger\in\mathcal{Q}^\dagger} \varphi_{Q^\dagger}$. Then $P' \models \varphi$, so $P \models \langle a\rangle\varphi$. Consequently, also $Q \models \langle a\rangle\varphi$. Hence there is a Q' with $Q \xrightarrow{\alpha} Q'$ and $Q' \models \varphi$. Since none of the $Q^\dagger \in \mathcal{Q}^\dagger$ satisfies φ, one obtains $Q' \notin \mathcal{Q}^\dagger$ and thus $P' \equiv Q'$.

- Let $X \subseteq A$ and suppose $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$. Let

  $$\mathcal{Q}^\dagger := \{Q^\dagger \in \mathbb{P} \mid Q \xrightarrow{\mathrm{t}} Q^\dagger \wedge P' \not\equiv_X Q^\dagger\}.$$

  For each $Q^\dagger \in \mathcal{Q}^\dagger$, let $\varphi_{Q^\dagger} \in \mathbb{O}$ be a formula such that $P' \models_X \varphi_{Q^\dagger}$ and $Q^\dagger \not\models_X \varphi_{Q^\dagger}$. Define $\varphi := \bigwedge_{Q^\dagger\in\mathcal{Q}^\dagger} \varphi_{Q^\dagger}$. Then $P' \models_X \varphi$, so $P \models \langle X\rangle\varphi$. Consequently, also $Q \models \langle X\rangle\varphi$. Hence there is a Q' with $Q \xrightarrow{\mathrm{t}} Q'$ and $Q' \models_X \varphi$. Again $Q' \notin \mathcal{Q}^\dagger$ and thus $P' \equiv_X Q'$.

- Suppose $P \equiv_Y Q$ and $P \xrightarrow{\alpha} P'$ with $a \in A$ and either $a \in Y$ or $\mathcal{I}(P) \cap (Y \cup \{\tau\}) = \emptyset$. Let $\mathcal{Q}^\dagger := \{Q^\dagger \in \mathbb{P} \mid Q \xrightarrow{\alpha} Q^\dagger \wedge P' \not\equiv Q^\dagger\}$. For each $Q^\dagger \in \mathcal{Q}^\dagger$, let $\varphi_{Q^\dagger} \in \mathbb{O}$ be a formula such that $P' \models \varphi_{Q^\dagger}$ and $Q^\dagger \not\models \varphi_{Q^\dagger}$. Define $\varphi := \bigwedge_{Q^\dagger\in\mathcal{Q}^\dagger} \varphi_{Q^\dagger}$. Then $P' \models \varphi$, so $P \models \langle a\rangle\varphi$, and also $P \models_Y \langle a\rangle\varphi$, using either the third or last clause in the definition of $\models_X$. Hence also $Q \models_Y \langle a\rangle\varphi$. Therefore there is a Q' with $Q \xrightarrow{\alpha} Q'$ and $Q' \models \varphi$, using the third clause of either $\models_X$ or $\models$. Since none of the $Q^\dagger \in \mathcal{Q}^\dagger$ satisfies φ, one obtains $Q' \notin \mathcal{Q}^\dagger$ and thus $P' \equiv Q'$.

- The fourth clause of Definition 3 is obtained exactly like the first, but using $\models_Y$ instead of $\models$.

- Suppose $P \equiv_Y Q$, $P \xrightarrow{\mathrm{t}} P'$ and $\mathcal{I}(P) \cap (X \cup Y \cup \{\tau\}) = \emptyset$, with $X \subseteq A$. Let

  $$\mathcal{Q}^\dagger := \{Q^\dagger \in \mathbb{P} \mid Q \xrightarrow{\mathrm{t}} Q^\dagger \wedge P' \not\equiv_X Q^\dagger\}.$$

  For each $Q^\dagger \in \mathcal{Q}^\dagger$, let $\varphi_{Q^\dagger} \in \mathbb{O}$ be a formula such that $P' \models_X \varphi_{Q^\dagger}$ and $Q^\dagger \not\models_X \varphi_{Q^\dagger}$. Define $\varphi := \bigwedge_{Q^\dagger\in\mathcal{Q}^\dagger} \varphi_{Q^\dagger}$. Then $P' \models_X \varphi$, so $P \models \langle X\rangle\varphi$, and thus $P \models_Y \langle X\rangle\varphi$. Consequently, also $Q \models_Y \langle X\rangle\varphi$ and therefore $Q \models \langle X\rangle\varphi$. Hence there is a Q' with $Q \xrightarrow{\mathrm{t}} Q'$ and $Q' \models_X \varphi$. Again $Q' \notin \mathcal{Q}^\dagger$ and thus $P' \equiv_X Q'$. ◀

▶ **Proposition 9.** $P \mathrel{\underline{\leftrightarrow}}_r Q$ iff there exists a strong time-out bisimulation $\mathscr{B}$ with $P \mathrel{\mathscr{B}} Q$.

Proof. Let $\mathscr{R}$ be a gsrb on $\mathbb{P}$. Define $\mathscr{B} \subseteq \mathbb{P} \times \mathbb{P}$ by $P \mathrel{\mathscr{B}} Q$ iff either $(P,Q) \in \mathscr{R}$ or $P = \theta_X(P^\dagger)$, $Q = \theta_X(Q^\dagger)$ and $(P^\dagger, X, Q^\dagger) \in \mathscr{R}$. I show that $\mathscr{B}$ is a strong time-out bisimulation.

- Let $P \mathrel{\mathscr{B}} Q$ and $P \xrightarrow{a} P'$ with $a \in A$. First suppose $(P,Q) \in \mathscr{R}$. Then, by the first clause of Definition 3, there exists a Q' such that $Q \xrightarrow{a} Q'$ and $(P',Q') \in \mathscr{R}$. So $P' \mathrel{\mathscr{B}} Q'$.
  Next suppose $P = \theta_X(P^\dagger)$, $Q = \theta_X(Q^\dagger)$ and $(P^\dagger, X, Q^\dagger) \in \mathscr{R}$. Since $\theta_X(P^\dagger) \xrightarrow{a} P'$ it must be that $P^\dagger \xrightarrow{a} P'$ and either $a \in X$ or $P^\dagger \overset{\beta}{\not\longrightarrow}$ for all $\beta \in X \cup \{\tau\}$. Hence there exists a Q' such that $Q^\dagger \xrightarrow{a} Q'$ and $(P',Q') \in \mathscr{R}$, using the third clause of Definition 3. Recall that $P^\dagger \mathrel{\underline{\leftrightarrow}}_r^X Q^\dagger$ implies $I(P^\dagger) \cap (X \cup \{\tau\}) = I(Q^\dagger) \cap (X \cup \{\tau\})$, and thus either $a \in X$ or $Q^\dagger \overset{\beta}{\not\longrightarrow}$ for all $\beta \in X \cup \{\tau\}$. It follows that $Q = \theta_X(Q^\dagger) \xrightarrow{a} Q'$ and $P' \mathrel{\mathscr{B}} Q'$.
- Let $P \mathrel{\mathscr{B}} Q$ and $P \xrightarrow{\tau} P'$. First suppose $(P,Q) \in \mathscr{R}$. Then, using the first clause of Definition 3, there is a Q' with $Q \xrightarrow{\tau} Q'$ and $(P',Q') \in \mathscr{R}$. So $P' \mathrel{\mathscr{B}} Q'$.
  Next suppose $P = \theta_X(P^\dagger)$, $Q = \theta_X(Q^\dagger)$ and $(P^\dagger, X, Q^\dagger) \in \mathscr{R}$. Since $\theta_X(P^\dagger) \xrightarrow{\tau} P'$, it must be that P' has the form $\theta_X(P^\ddagger)$, and $P^\dagger \xrightarrow{\tau} P^\ddagger$. Thus, by the fourth clause of Definition 3, there is a $Q^\ddagger$ with $Q^\dagger \xrightarrow{\tau} Q^\ddagger$ and $(P^\ddagger, X, Q^\ddagger) \in \mathscr{R}$. Now $Q = \theta_X(Q^\dagger) \xrightarrow{\tau} \theta_X(Q^\ddagger) =: Q'$ and $P' \mathrel{\mathscr{B}} Q'$.
- Let $P \mathrel{\mathscr{B}} Q$, $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$. First suppose $(P,Q) \in \mathscr{R}$. Then, by the second clause of Definition 3, there is a Q' with $Q \xrightarrow{\mathrm{t}} Q'$ and $(P', X, Q') \in \mathscr{R}$. So $\theta_X(P') \mathrel{\mathscr{B}} \theta_X(Q')$.
  Next suppose $P = \theta_Y(P^\dagger)$, $Q = \theta_Y(Q^\dagger)$ and $(P^\dagger, Y, Q^\dagger) \in \mathscr{R}$. Since $\theta_Y(P^\dagger) \xrightarrow{\mathrm{t}} P'$, it must be that $P^\dagger \xrightarrow{\mathrm{t}} P'$ and $P^\dagger \overset{\beta}{\not\longrightarrow}$ for all $\beta \in Y \cup \{\tau\}$. Consequently, $\mathcal{I}(P^\dagger) = \mathcal{I}(P)$ and thus $\mathcal{I}(P^\dagger) \cap (X \cup Y \cup \{\tau\}) = \emptyset$. By the last clause of Definition 3 there is a Q' such that $Q^\dagger \xrightarrow{\mathrm{t}} Q'$ and $(P, X, Q') \in \mathscr{R}$. So $\theta_X(P') \mathrel{\mathscr{B}} \theta_X(Q')$. From $(P^\dagger, Y, Q^\dagger) \in \mathscr{R}$ and $\mathcal{I}(P^\dagger) \cap (Y \cup \{\tau\}) = \emptyset$, I infer $\mathcal{I}(Q^\dagger) \cap (Y \cup \{\tau\}) = \emptyset$. So $Q^\dagger \overset{\beta}{\not\longrightarrow}$ for all $\beta \in Y \cup \{\tau\}$. This yields $Q = \theta_Y(Q^\dagger) \xrightarrow{\mathrm{t}} Q'$.

Now let $\mathscr{B}$ be a time-out bisimulation. Define $\mathscr{R} \subseteq \mathbb{P} \times \mathscr{P}(A) \times \mathbb{P}$ by $(P,Q) \in \mathscr{R}$ iff $P \mathrel{\mathscr{B}} Q$, and $(P,X,Q) \in \mathscr{R}$ iff $\theta_X(P) \mathrel{\mathscr{B}} \theta_X(Q)$. I need to show that $\mathscr{R}$ is a gsrb.

- Suppose $(P,Q) \in \mathscr{R}$ and $P \xrightarrow{\alpha} P'$ with $\alpha \in A \cup \{\tau\}$. Then $P \mathrel{\mathscr{B}} Q$, so there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathrel{\mathscr{B}} Q'$. Hence $(P',Q') \in \mathscr{R}$.
- Suppose $(P,Q) \in \mathscr{R}$, $X \subseteq A$, $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$. Then $P \mathrel{\mathscr{B}} Q$, so $\exists Q'$ such that $Q \xrightarrow{\mathrm{t}} Q'$ and $\theta_X(P') \mathrel{\mathscr{B}} \theta_X(Q')$. Thus $(P', X, Q') \in \mathscr{R}$.
- Suppose $(P,X,Q) \in \mathscr{R}$ and $P \xrightarrow{a} P'$ with either $a \in X$ or $\mathcal{I}(P) \cap (X \cup \{\tau\}) = \emptyset$. Then $\theta_X(P) \mathrel{\mathscr{B}} \theta_X(Q)$. Moreover, $\theta_X(P) \xrightarrow{a} P'$. Hence there is a Q' such that $\theta_X(Q) \xrightarrow{a} Q'$ and $P' \mathrel{\mathscr{B}} Q'$. It must be that $Q \xrightarrow{a} Q'$. Moreover, $(P',Q') \in \mathscr{R}$.
- Suppose $(P,X,Q) \in \mathscr{R}$ and $P \xrightarrow{\tau} P'$. Then $\theta_X(P) \mathrel{\mathscr{B}} \theta_X(Q)$. Since $P \xrightarrow{\tau} P'$, one has $\theta_X(P) \xrightarrow{\tau} \theta_X(P')$. Hence there is an R such that $\theta_X(Q) \xrightarrow{\tau} R$ and $\theta_X(P') \mathrel{\mathscr{B}} R$. The process R must have the form $\theta_X(Q')$ for some Q' with $Q \xrightarrow{\tau} Q'$. It follows that $(P', X, Q') \in \mathscr{R}$.
- Suppose $(P,Y,Q) \in \mathscr{R}$, $X \subseteq A$, $\mathcal{I}(P) \cap (X \cup Y \cup \{\tau\}) = \emptyset$ and $P \xrightarrow{\mathrm{t}} P'$. Then $\theta_Y(P) \mathrel{\mathscr{B}} \theta_Y(Q)$ and $\theta_Y(P) \xrightarrow{\mathrm{t}} P'$. Moreover, $\mathcal{I}(\theta_Y(P)) = \mathcal{I}(P)$, so by the second clause of Definition 8 there exists a Q' such that $\theta_Y(Q) \xrightarrow{\mathrm{t}} Q'$ and $\theta_X(P') \mathrel{\mathscr{B}} \theta_X(Q')$. So $Q \xrightarrow{\mathrm{t}} Q'$ and $(P', X, Q') \in \mathscr{R}$. ◀

Stratification. Even though negative premises occur in Table 1, the meaning of this transition system specification is well-defined, for instance by the method of *stratification* explained in [21, 13]. Assign inductively to each expression $E \in \mathbb{E}$ an ordinal λ_E that counts the nesting depth of recursive specifications: if $E = \langle x|\mathcal{S}\rangle$ then λ_E is 1 more than the supremum of the $\lambda_{\mathcal{S}_y}$ for $y \in V_{\mathcal{S}}$; otherwise λ_E is the supremum of $\lambda\langle x|\mathcal{S}\rangle$ for all subterms $\langle x|\mathcal{S}\rangle$ of E. Moreover $\kappa_E \in \mathbb{N}$ is the nesting depth of θ_L^U and ψ_X operators in E that remain after replacing any subterm F of E with $\lambda_F < \lambda_E$ by 0. Now the ordered pair (λ_P, κ_P) constitutes a valid stratification for closed literals $P \xrightarrow{\alpha} P'$. Namely, whenever a transition $P \xrightarrow{\alpha} P'$ depends on a transition $Q \xrightarrow{\beta} Q'$, in the sense that that there is a closed substitution instance $\mathfrak{r}$ of a rule from Table 1 with conclusion $P \xrightarrow{\alpha} P'$, and $Q \xrightarrow{\beta} Q'$ occurring in its premises, then either $\lambda_Q < \lambda_P$, or $\lambda_Q = \lambda_P$ and $\kappa_Q \leq \kappa_P$. Moreover, when $P \xrightarrow{\alpha} P'$ depends on a negative literal $Q \not\xrightarrow{\beta}$, then $\lambda_Q = \lambda_P$ and $\kappa_Q < \kappa_P$.

The above argument hinges on the exclusion of invalid CCSP$_{\rm t}^\theta$ expressions. The invalid expression $P := \langle x \mid \{x = \theta_{\{a\}}^{\{a\}}(b.0 + \mathcal{R}(x))\}\rangle$ for instance, with $\mathcal{R} = \{(b,a)\}$, does not have a well-defined meaning, since the transition $P \xrightarrow{b} 0$ is derivable iff one has the premise $P \not\xrightarrow{b}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{b.0 \xrightarrow{b} 0}}{b.0 + \mathcal{R}(P) \xrightarrow{b} 0} \qquad \dfrac{\dfrac{P \not\xrightarrow{b}}{\mathcal{R}(P) \not\xrightarrow{a}}}{b.0 + \mathcal{R}(P) \not\xrightarrow{a}} \qquad \dfrac{\dfrac{P \not\xrightarrow{\tau}}{\mathcal{R}(P) \not\xrightarrow{\tau}}\ \text{(OK)}}{b.0 + \mathcal{R}(P) \not\xrightarrow{\tau}}}{\theta_{\{a\}}^{\{a\}}(b.0 + \mathcal{R}(P)) \xrightarrow{b} 0}}{P \xrightarrow{b} 0}$$

However, the meaning of the valid expression $\langle x \mid \{x = \theta_{\{a\}}^{\{a\}}(\langle y|\{y = b.y\}\rangle) \|_\emptyset \mathcal{R}(x)\}\rangle$, for instance, is entirely unproblematic.

▶ **Proposition 12.** Each CCSP$_{\rm t}^\theta$ process is countably branching.

Proof. I show that for each CCSP$_{\rm t}^\theta$ process Q there are only countably many transitions $Q \xrightarrow{\alpha} R$. Each such transition must be derivable from the rules of Table 1. So it suffices to show that for each Q there are only countably many derivations of transitions $Q \xrightarrow{\alpha} R$.

A derivation of a transition is a well-founded, upwardly branching tree, in which each node models an application of one of the rules of Table 1. Since each of these rules has finitely many positive premises, such a proof tree is finitely branching, and thus finite. Let $d(\pi)$, the *depth* of π, be the length of the longest branch in a derivation π. If π derives a transition $Q \xrightarrow{\alpha} R$, then I call Q the *source* of π.

It suffices to show that for each $n \in \mathbb{N}$ there are only finitely many derivations of depth n with a given source. This I do by induction on n.

In case $Q = f(Q_1, \dots, Q_k)$, with f an k-ary CCSP$_{\rm t}^\theta$ operator, a derivation π of depth n is completely determined by the concluding rule from Table 1, deriving a transition $Q \xrightarrow{\beta} R$, the subderivations of π with source Q_i for some of the $i \in \{1, \dots, k\}$, and the transition label β. The choice of the concluding rule depends on f, and for each f there are at most three choices. The subderivations of π with source Q_i have depth $< n$, so by induction there are only finitely many. When f is not a renaming operator $\mathcal{R}$, there is no further choice for the transition label β, as it is completely determined by the premises of the rule, and thus by the subderivations of those premises. In case $f = \mathcal{R}$, there are finitely many choices for β when faced with a given transition label α contributed by the premise of the rule for renaming. Here I use the requirement of Section 5 that all sets $\{b \mid (a,b) \in \mathcal{R}\}$ are finite. This shows there are only finitely many choices for π.

In case $Q = \langle x|\mathcal{S}\rangle$, the last step in π must be application of the rule for recursion, so π is completely determined by a subderivation π' of a transition with source $\langle \mathcal{S}_x|\mathcal{S}\rangle$. By induction there are only finitely many choices for π', and hence also for π. ◀

▶ **Proposition 13.** Each CCSP$_\mathrm{t}^\theta$ process with guarded recursion is finitely branching.

Proof. A trivial structural induction shows that if P is a CCSP$_\mathrm{t}^\theta$ process with guarded recursion and Q is reachable from P, then also Q has with guarded recursion. Hence it suffices to show that for each CCSP$_\mathrm{t}^\theta$ process Q with guarded recursion there are only finitely many derivations with source Q.

Let $\xrightarrow{u}$ be the smallest binary relation on $\mathbb{P}$ such that (i) $f(P_1,\dots,P_k) \xrightarrow{u} P_i$ for each k-ary CCSP$_\mathrm{t}^\theta$ operator f except action prefixing, and each $i \in \{1,\dots,k\}$, and (ii) $\langle x|\mathcal{S}\rangle \xrightarrow{u} \langle \mathcal{S}_x|\mathcal{S}\rangle$. This relation is finitely branching. Moreover, on processes with guarded recursion, $\xrightarrow{u}$ has no forward infinite chains $P_0 \xrightarrow{u} P_1 \xrightarrow{u} \dots$. In fact, this could have been used as an alternative definition of guarded recursion. Let, for any process Q with guarded recursion, $e(Q)$ be the length of the longest forward chain $Q \xrightarrow{u} P_1 \xrightarrow{u} \dots \xrightarrow{u} P_{e(Q)}$. I show with induction on $e(Q)$ that there are only finitely many derivations with source Q. In fact, this proceeds exactly as in the previous proof. ◀

▶ **Proposition 14.** Each finitely branching processes in an LTS can be denoted by a CCSP$_\mathrm{t}$ expression with guarded recursion. Here I only need the operations inaction, action prefixing, choice and recursion.

Proof. Let P be a finitely branching process in an LTS $(\mathbb{P}', \mathit{Act}, \rightarrow)$. Let

$$V_\mathcal{S} := \{x_Q \mid Q \in \mathbb{P}' \text{ is reachable from } P\} \subseteq \mathit{Var}.$$

For each Q reachable from P, let $\mathit{next}(Q)$ be the finite set of pairs $(\alpha, R) \in \mathit{Act} \times \mathbb{P}'$ such that there is a transition $Q \xrightarrow{\alpha} R$. Let $\mathcal{S} := \{x_Q = \sum_{(\alpha,R)\in next(Q)} \alpha.x_R \mid x_Q \in V_\mathcal{S}\}$. Here the finite choice operator $\sum_{i\in I} \alpha_i.P_i$ can easily be expressed in terms of inaction, action prefixing and choice. Now the CCSP$_\mathrm{t}$ process $\langle x_P|\mathcal{S}\rangle$ denotes P. ◀

In fact, $\langle x_P|\mathcal{S}\rangle \leftrightarrow P$, where $\leftrightarrow$ denotes strong bisimilarity [29], formally defined in the next section.

▶ **Proposition 15.** Each countably branching processes in an LTS can be denoted by a CCSP$_\mathrm{t}$ expression. Again I only need the CCSP$_\mathrm{t}$ operations 0, $\alpha.\_$, $+$ and recursion.

Proof. The proof is the same as the previous one, except that $\mathit{next}(Q)$ now is a countable set, rather than a finite one, and consequently I need a countable choice operator $\sum_{i\in\mathbb{N}} \alpha_i.P_i$. The latter can be expressed in CCSP$_\mathrm{t}$ with unguarded recursion by

$$\sum_{i\in\mathbb{N}} \alpha_i.P_i := \langle z_o|\{z_i = \alpha_i.P_i + z_{i+1} \mid i \in \mathbb{N}\}\rangle. \quad ◀$$

▶ **Lemma 32.** For each CCSP$_\mathrm{t}^\theta$ process P there exists a CCSP$_\mathrm{t}^\theta$ process Q only built using inaction, action prefixing, choice and recursion, such that $P \leftrightarrow Q$.

Proof. Immediately from Propositions 12 and 15. ◀

▶ **Theorem 20.** Strong bisimilarity is a full congruence for CCSP$_\mathrm{t}^\theta$.

Proof. The structural operational rules for CCSP$_\mathrm{t}$ fit the *tyft/tyxt format with recursion* of [14]. By [14, Theorem 3] this implies that $\leftrightarrow$ is a full congruence for CCSP$_\mathrm{t}$. (In fact, when omitting the recursion construct, the operational rules for CCSP$_\mathrm{t}$ fit the *tyft/tyxt format* of [22], and by the main theorem of [22], $\leftrightarrow$ is a congruence *for the operators of* CCSP$_\mathrm{t}$, that is, it satisfies (1) in Definition 16. The work of [14] extends this result of [22] with recursion.)

The structural operational rules for all of $\mathrm{CCSP}_\mathrm{t}^\theta$ fit the *ntyft/ntyxt format with recursion* of [14]. By [14, Theorem 2] this implies that $\underline{\leftrightarrow}$ is a lean congruence for $\mathrm{CCSP}_\mathrm{t}^\theta$. (In fact, when omitting the recursion construct, the operational rules for $\mathrm{CCSP}_\mathrm{t}^\theta$ fit the *ntyft/ntyxt format* of [21], and by the main theorem of [21], $\underline{\leftrightarrow}$ is a congruence for the operators of $\mathrm{CCSP}_\mathrm{t}^\theta$. The work of [14] extends this result of [21] with recursion.)

To verify (2) for the whole language $\mathrm{CCSP}_\mathrm{t}^\theta$, let $\mathcal{S}$ and $\mathcal{S}'$ be recursive specifications with $x \in V_\mathcal{S} = V_{\mathcal{S}'}$, such that $\langle x|\mathcal{S}\rangle, \langle x|\mathcal{S}'\rangle \in \mathbb{P}$ and $\mathcal{S}_y \;\underline{\leftrightarrow}\; \mathcal{S}'_y$ for all $y \in V_\mathcal{S}$. Let $\{P_i \mid i \in I\}$ be the collection of processes of the form $\theta_L^U(Q)$ or $\psi_X(Q)$, for some L, U, X, that occur as a closed subexpression of $\mathcal{S}_y$ or $\mathcal{S}'_y$ for one of the $y \in V_\mathcal{S}$, not counting strict subexpressions of a closed subexpression R of $\mathcal{S}_y$ or $\mathcal{S}'_y$ that is itself of the form $\theta_L^U(Q)$ or $\psi_X(Q)$. Pick a fresh variable $z_i \notin V_\mathcal{S}$ for each $i \in I$, and let, for $y \in Y$, $\widehat{\mathcal{S}}_y$ be the result of replacing each occurrence of P_i in $\mathcal{S}_y$ by z_i. Then $\widehat{\mathcal{S}}_y$ does not contain the operators $\theta_L^U(Q)$ or $\psi_X(Q)$. In deriving this conclusion it is essential that $\langle x|\mathcal{S}\rangle$ is a valid expression, for this implies that the term $\mathcal{S}_y \in \mathbb{E}$, which may contain free occurrences of the variables $y \in V_\mathcal{S}$, does not have a subterm of the form $\theta_L^U(F)$ or $\psi_X(F)$ that contains free occurrences of these variables. Let $\widehat{\mathcal{S}} := \{y = \widehat{\mathcal{S}}_y \mid y \in V_\mathcal{S}\}$; it is a recursive specification in the language CCSP_t. The recursive specification $\widehat{\mathcal{S}'}$ is defined in the same way.

For each $i \in I$ there is, by Lemma 32, a process Q_i in the language CCSP_t such that $P_i \;\underline{\leftrightarrow}\; Q_i$. Now let $\rho, \eta : \{z_i \mid i \in I\} \rightarrow \mathbb{P}$ be the substitutions defined by $\rho(z_i) = P_i$ and $\eta(z_i) = Q_i$ for all $i \in I$. Then $\rho \;\underline{\leftrightarrow}\; \eta$. Since $\underline{\leftrightarrow}$ is a lean congruence for $\mathrm{CCSP}_\mathrm{t}^\theta$, one has $\langle x|\widehat{\mathcal{S}}\rangle[\rho] \;\underline{\leftrightarrow}\; \langle x|\widehat{\mathcal{S}}\rangle[\eta]$ and likewise $\langle x|\widehat{\mathcal{S}'}\rangle[\rho] \;\underline{\leftrightarrow}\; \langle x|\widehat{\mathcal{S}'}\rangle[\eta]$. For the same reason one has $\widehat{\mathcal{S}}_y[\eta] \;\underline{\leftrightarrow}\; \widehat{\mathcal{S}}_y[\rho] = \mathcal{S}_y \;\underline{\leftrightarrow}\; \mathcal{S}'_y \;\underline{\leftrightarrow}\; \widehat{\mathcal{S}}'_y[\rho] \;\underline{\leftrightarrow}\; \widehat{\mathcal{S}}'_y[\eta]$ for all $y \in V_\mathcal{S}$. Since $\widehat{\mathcal{S}}[\eta]$ and $\widehat{\mathcal{S}}'[\eta]$ are recursive specifications over CCSP_t, $\langle x|\widehat{\mathcal{S}}[\eta]\rangle \;\underline{\leftrightarrow}\; \langle x|\widehat{\mathcal{S}}'[\eta]\rangle$. Hence

$$\langle x|\mathcal{S}\rangle = \langle x|\widehat{\mathcal{S}}[\rho]\rangle = \langle x|\widehat{\mathcal{S}}\rangle[\rho] \;\underline{\leftrightarrow}\; \langle x|\widehat{\mathcal{S}}\rangle[\eta] = \langle x|\widehat{\mathcal{S}}[\eta]\rangle \;\underline{\leftrightarrow}\; \langle x|\widehat{\mathcal{S}}'[\eta]\rangle \;\underline{\leftrightarrow}\; \langle x|\widehat{\mathcal{S}}'[\rho]\rangle = \langle x|\mathcal{S}'\rangle. \qquad \blacktriangleleft$$