Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Branching Time and Orthogonal Bisimulation Equivalence

J.A. Bergstra, A. Ponse, M.B. van der Zwaag

Software Engineering (SEN)

SEN-R0035 December 31, 2000

Report SEN-R0035
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Branching Time and Orthogonal Bisimulation Equivalence

Jan A. Bergstra[2,3] Alban Ponse[1,2] Mark B. van der Zwaag[1]

[1] *CWI, Cluster Software Engineering*
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
[2] *University of Amsterdam, Programming Research Group*
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
[3] *Utrecht University, Department of Philosophy*
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

ABSTRACT
We propose a refinement of branching bisimulation equivalence that we call orthogonal bisimulation equivalence. Typically, internal activity (i.e., the performance of τ-steps) may be compressed, but not completely discarded. Hence, a process with τ-steps cannot be equivalent to one without τ-steps. Also, we present a modal characterization of orthogonal bisimulation equivalence. This equivalence is a congruence for ACP extended with abstraction and priority operations. We provide a complete axiomatization, and describe some expressiveness results. Finally, we present the verification of a PAR protocol that is specified with use of priorities.

2000 Mathematics Subject Classification: 68Q55; 68Q60; 68Q70; 68Q85
Keywords & Phrases: Orthogonal Bisimulation, Branching Time, Process Algebra, Silent Step
Note: Work carried out under project SEN2.1 "Process Specification and Analysis".

1 Introduction

In concurrency theory, Milner's *observation equivalence* as discussed in the setting of CCS (Calculus of Communicating Systems [20], cf. [22, 23]) is a standard example of a branching time behavioral equivalence that deals with *abstraction*. Here 'branching time' refers to the fact that the branching structure of processes is taken into account, and 'abstraction' refers to a mechanism to hide actions that are assumed not to be observable or interesting for some other reason. In the process algebraic approaches based on ACP (Algebra of Communicating Processes [7], overviewed in [5, 12]), observation equivalence is named *τ-bisimulation equivalence* [8], and abstraction boils down to renaming actions into the *silent* step (or action) τ, the occurrences of which then may be eliminated according to certain axioms. Abstraction is a prominent feature in process algebra, serving both verification styles and expressive power.

A popular and relatively new semantics that deals with abstraction, proposed by van Glabbeek and Weijland in [18], is *branching bisimulation equivalence* (see also [19]). Branching bisimulation equivalence is a refinement of semantics such as observation equivalence, delay bisimulation equivalence [21] and η-bisimulation equivalence [3], and can be considered an improvement of these because it fully respects the branching structure of processes. In the words of [19]: "in two [branching] bisimilar processes every computation [sequence of steps]

in the one process corresponds to a computation in the other, in such a way that all intermediate states of these computations correspond as well, due to the [branching] bisimulation relation." We recall that in branching bisimilarity, the axiom

$$x \cdot \tau = x$$

(or, $a.\tau.x = a.x$ in a setting with action prefixing $a._-$, such as CCS [20]) is claimed to be at the very heart of abstraction (cf. [19]). This axiom expresses that the observational contents of the silent step τin a sequential context $x\tau$ (we usually omit the symbol $\cdot$ in terms) is totally void. Branching bisimulation equivalence is the behavioral equivalence that characterizes this notion of 'observational contents' in the setting of process algebra (cf. [16, 19]; we return to this point in Section 10).

In this paper we propose a refinement of branching bisimulation equivalence, called *orthogonal bisimulation equivalence*, which has the following two main characteristics.

1. Internal activity (i.e., the performance of τ-steps) may be compressed, but not completely discarded.

2. Operations that grasp at the local structure of a process, such as the *priority operator*, are compatible with this semantics and do not require any special treatment of τ.

Our bisimulation equivalence is called "orthogonal" because it respects the dichotomy between *concrete processes* [4, 17], i.e., processes in which no internal actions occur, and those that contain τ-steps: a process without τ-steps cannot be equivalent to one with τ-steps. As a consequence, orthogonal bisimilarity is a less abstract equivalence than those discussed above. Below we elaborate on the two characteristics mentioned.

Let *compression* stand for the reduction of finitary internal activity (characterized by τ-steps) to a single τ-step. Compression is valid in orthogonal bisimilarity, and after compression, the presence of a τ-step is as decisive as that of any observable action and indicates the presence of some internal activity. For example,

$$a(\tau + \tau\tau)$$

is orthogonally bisimilar to its compressed form $a\tau$, and both represent the action a followed by some internal activity. Furthermore, neither of these two is orthogonally bisimilar to a. Hence, the axiom $x = x\tau$ is not sound in orthogonal bisimulation equivalence (its weakened version $x\tau\tau = x\tau$ is sound). Typically, in orthogonal bisimilarity one may abstract from the *structure* of finitary internal activity, but not from its *presence*. This is a major difference with branching bisimulation equivalence and the coarser (larger, i.e., more identifying) semantics on abstraction mentioned above. We now consider the case of *divergence*, i.e., the occurrence of an infinite τ-path. In branching bisimulation equivalence, a *τ-loop* may be discarded in case there is an alternative available, which can be explained as a feature: often τ-loops result from the abstraction of the occurrence and recovery of an undesirable event, for example the corruption and retransmission of a data-package in a communication protocol. Discarding such a loop corresponds with the assumption that it will not be chosen infinitely often (and, following the example, with the assumption that the occurrence and recovery of an undesirable event may only be repeated consecutively a finite number of times). In process algebra, this

assumption is called *fairness* and it often plays an important role in verifications. Whereas in branching bisimilarity τ-loops can always be discarded, this is not the case in orthogonal bisimilarity. According to the first characteristic above, a τ-loop may be discarded only if one of its exits starts with an initial τ-step. We also distinguish a second, more restricted variant of orthogonal bisimulation equivalence that preserves divergence in all circumstances, *divergence sensitive orthogonal bisimilarity* (reminiscent of *branching bisimulation equivalence with explicit divergence* as defined in [19]).

The priority operator θ was introduced in [1]. It can for example be used to give priority to interrupts or internal behavior in a process algebra specification of some protocol, or to give lowest priority to the execution of time-outs or error messages. Essentially, the priority operator is based on a (fixed, partial) ordering on actions, and prevents an action (and its subsequent behavior) to be executed in the case that there is an alternative with a higher priority. Right at its introduction, it was recognized that the priority operator θ and abstraction are difficult to combine, and a modular approach was advocated for using both in a process algebra verification: first eliminate all occurrences of the priority operator, and then apply abstraction as to arrive at a concise characterization of the *external* behavior. The priority operation not being fully compatible with any known semantics that deals with abstraction[1] is an immediate consequence of the axiom $x\tau = x$. The main cause for this problem is that on the term level τ can hide alternatives, by which $x\tau y$ can be different from xy in the scope of the priority operator. For example, assume for actions a, b, c the priority ordering $a < \{b, c\}$. Then the process term $\theta(a \,||\, b\tau c)$, where $a \,||\, b\tau c$ represents a in parallel with b followed by τ followed by c, defines a behavior in which the action a may be executed before c, a situation that cannot occur in $\theta(a \,||\, bc)$. This shows that without any special measures, the priority operator is not compatible with the axiom $x\tau = x$. However, orthogonal bisimilarity is a congruence for the priority operator (even in the case that τ has a priority).

In the above we informally introduced orthogonal bisimulation equivalence. In the remainder of this paper we establish its definition (Section 2) and provide a modal characterization (Section 3). Furthermore, we define the system $\mathrm{ACP}_\tau^{\mathrm{orth}}$ in Section 4, and we prove some completeness results in Section 5. Then in Section 6 we consider the priority operator, and argue that it is compatible with orthogonal bisimilarity. In Section 7 we introduce some forms of *iteration* in order to describe infinite processes, and we shortly discuss fairness in the present setting. Section 8 is on expressiveness modulo orthogonal bisimilarity. Finally, in Section 9 we describe as an example the specification and verification of a PAR protocol [26] in orthogonal bisimulation equivalence. The paper is ended with some remarks and conclusions in Section 10. We added two appendices, containing some congruence proofs and examples on expressiveness.

Note. In earlier work [28, 29], orthogonal bisimilarity was defined using a constant ι instead of τ. We now consider this use of the symbol ι obsolete.

[1] In the literature, several solutions for this problem were proposed, but none of these are totally satisfactory and generally accepted; we return to this issue in our conclusions in Section 10.

2 Transition Systems and Orthogonal Bisimilarity

A transition system (over L) is a triple (S, L, T), where S is a nonempty set of states, L is a set of labels and $T \subseteq S \times L \times S$ is a transition relation.

We usually write $s \xrightarrow{a} s'$ for a transition (s, a, s'); we call it an a-transition and refer to s as its source and to s' as its target. We write $s \xrightarrow{a}$ if s has an outgoing a-transition. We call s' a (a-)successor of s, if s has an outgoing (a-)transition with target s'.

In the process algebras introduced in Section 4, we distinguish states that have the option to terminate successfully. This is achieved by considering transition systems together with a valuation function that assigns propositions to states: if P is a set of proposition letters, then a valuation is a function V from the state set to the power set of P. In this paper we shall use $\surd$ as the termination symbol; we say that a state s has the option to terminate if $\surd \in V(s)$.

The special label τ represents an unobservable, or *silent*, action. We write L_τ for a set of transition labels containing τ; for a set L, we let $L_\tau = L \cup \{\tau\}$.

For any state s, we define the set of finite τ-paths starting in s inductively as follows:

1. $s \in \tau\text{-}paths(s)$;
2. if $s_0 \cdots s_n \in \tau\text{-}paths(s)$ for some $n \geq 0$, and $s_n \xrightarrow{\tau} s'$, then $s_0 \cdots s_n s' \in \tau\text{-}paths(s)$.

We define orthogonal bisimulation equivalence of states. Below, we compare this equivalence relation with strong bisimulation equivalence and branching bisimulation equivalence, that are well-known from the literature.

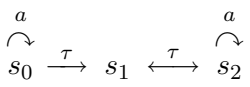
Definition 2.1 Consider a transition system (S, L_τ, T) with valuation V. A binary relation R on S is an *orthogonal bisimulation*, if it is symmetric, and, whenever sRr, then

1. $V(s) = V(r)$, and
2. if $s \xrightarrow{a} s'$ for some a and s' with $a \neq \tau$, then $r \xrightarrow{a} r'$ for some r' with $s'Rr'$, and
3. if $s \xrightarrow{\tau} s'$ for some s', then $r \xrightarrow{\tau}$, and, for some $n \geq 0$, there is a $r_0 \cdots r_n \in \tau\text{-}paths(r)$ such that $s'Rr_n$ and sRr_i for all $i < n$.

States s and r are orthogonally bisimilar, notation $s \leftrightarrows_o r$, if they are related by some orthogonal bisimulation.

If states s and r are orthogonally bisimilar and $s \xrightarrow{a}$, then also $r \xrightarrow{a}$.

Example 2.1 Consider the transition system below. Observe that $s_0 \leftrightarrows_o s_2$. Also, it holds that $s_0 \leftrightarrows_o s_1$ if, and only if, $a = \tau$.



We see that from a state (s_0) an equivalent state (s_2) may be reached by τ-steps via nonequivalent states (s_1).

We defined bisimilarity of states in a single transition system. This can easily be extended to bisimilarity of states in different systems by first taking the disjoint union of the systems. The disjoint union of two transition systems is obtained by taking the disjoint union of the states, the union of the labels and the corresponding disjoint union of the transitions.

Lemma 2.1 If $s \leftrightarrows_o r$ and, for some $n \geq 0$, it holds that $s_0 \cdots s_n \in \tau\text{-}paths(s)$, then there is, for every $i \leq n$, an $m_i \geq 0$, such that r has a τ-path with $r_0^0 = r$ and $m_n = 0$ and

1. for all $i < n$, $r_i^0 \cdots r_i^{m_i} \in \tau\text{-}paths(r_i^0)$ and $r_i^{m_i} = r_{i+1}^0$, and
2. for all $i \leq n$, $r_i^j \leftrightarrows_o s_i$, if $j < m_i$ or $j = 0$.

Proof. Straightforward. □

Theorem 2.1 Orthogonal bisimilarity is an equivalence relation.

Proof. Consider a transition system (S, L_τ, T) with valuation V. We show that $\leftrightarrows_o$ is transitive. Reflexivity and symmetry are trivial. Let $s_1 \leftrightarrows_o s_2 \leftrightarrows_o s_3$. We show that the symmetric relation

$$R = \{(s,r),(r,s) \mid s \leftrightarrows_o t \leftrightarrows_o r \text{ for some } t \in S\}$$

is an orthogonal bisimulation, and thereby that $s_1 \leftrightarrows_o s_3$.

Take any $(s,r) \in R$. By definition of R we have that, for some t, either $s \leftrightarrows_o t \leftrightarrows_o r$, or $r \leftrightarrows_o t \leftrightarrows_o s$. Assume the former; the latter case is symmetric.

First, observe that $V(s) = V(t) = V(r)$.

Assume that $s \xrightarrow{a} s'$. We must show that r matches this transition appropriately. The case with $a \neq \tau$ is easy; assume that $a = \tau$. It is straightforward to verify that $r \xrightarrow{\tau}$.

Since $s \leftrightarrows_o t$, it holds that t matches the τ-step to s' in zero or more transitions, i.e. for some $n \geq 0$, there is a $t_0 \cdots t_n \in \tau\text{-}paths(t)$ such that $s \leftrightarrows_o t_i$ for all $i < n$ and $s' \leftrightarrows_o t_n$. The proof is finished straightforwardly using Lemma 2.1. □

Orthogonal bisimilarity is not a congruence with respect to the alternative composition in process algebra. We define here the equivalence relation *rooted* orthogonal bisimilarity, that will prove to be a congruence with respect to the process algebraic operators.

Definition 2.2 An orthogonal bisimulation R is *rooted* between states s and r, if sRr, and

1. if $s \xrightarrow{\tau} s'$ for some s', then $r \xrightarrow{\tau} r'$ for some r' with $s'Rr'$, and
2. if $r \xrightarrow{\tau} r'$ for some r', then $s \xrightarrow{\tau} s'$ for some s' with $s'Rr'$.

States s and r are rooted orthogonally bisimilar, notation $s \leftrightarrows_{ro} r$, if there is an orthogonal bisimulation that is rooted between s and r.

Theorem 2.2 Rooted orthogonal bisimilarity is an equivalence relation.

Proof. Straightforward using Theorem 2.1.

Other process equivalences

We recall the definitions of strong bisimulation equivalence [25] and branching bisimulation equivalence [19].

Definition 2.3 Consider a transition system (S, L, T) with valuation V. A binary relation R on S is a *strong bisimulation*, if it is symmetric, and whenever sRr, then

1. $V(s) = V(r)$, and
2. if $s \xrightarrow{a} s'$ for some a and s', then $r \xrightarrow{a} r'$ for some r' with $s'Rr'$.

States s and r are strongly bisimilar, notation $s \leftrightarrows r$, if they are related by some strong bisimulation.

Orthogonal bisimilarity is coarser (or larger) than strong bisimilarity; any strong bisimulation is also an orthogonal bisimulation. We show that for so-called *compact* states strong bisimilarity and orthogonal bisimilarity coincide.

Definition 2.4 A τ-transition is *inert*, if its source and target are orthogonally bisimilar. A state is *compact*, if it has no inert outgoing τ-transitions, and all its successors are compact.

Lemma 2.2 If s and r are compact and $s \leftrightarrows_o r$, then $s \leftrightarrows r$.

Proof. Suppose that R is an orthogonal bisimulation that relates s and r. Assume that there does not exists a smaller orthogonal bisimulation that relates s and r. R relates only compact processes. It is straightforward to show that R is a strong bisimulation. □

Corollary 2.1 If all successors of s and r are compact and $s \leftrightarrows_{ro} r$, then $s \leftrightarrows r$.

We also compare orthogonal bisimilarity with *branching* bisimilarity [19]. Branching bisimilarity is the finest (smallest, i.e., least identifying) of the process equivalences described in [15]. Orthogonal bisimilarity is finer than branching bisimilarity, and hence finer than the equivalences in [15].

Definition 2.5 Let $\Longrightarrow$ be the reflexive transitive closure of $\xrightarrow{\tau}$. Consider a transition system (S, L_τ, T) with some valuation V. A binary relation R on S is a *branching bisimulation*, if it is symmetric, and whenever sRr, then

1. there is an r' with $r \Longrightarrow r'$ and $V(s) = V(r')$, and
2. if $s \xrightarrow{a} s'$ for some a and s', then either
   (a) $a = \tau$ and $s'Rr$, or
   (b) $r \Longrightarrow r'' \xrightarrow{a} r'$ for some r'', r' with sRr'' and $s'Rr'$.

States s and r are branching bisimilar, notation $s \leftrightarrows_b r$, if they are related by some branching bisimulation.

It is straightforward to prove that any orthogonal bisimulation is a branching bisimulation. The root condition for branching bisimulations is the same as the root condition for orthogonal bisimulations.

Proposition 2.1 It holds that $\leftrightarrows \subseteq \underline{\leftrightarrows}_o \subseteq \underline{\leftrightarrows}_b$ and $\leftrightarrows \subseteq \underline{\leftrightarrows}_{ro} \subseteq \underline{\leftrightarrows}_{rb}$, for any transition system over L_τ, and any valuation.

Example 2.2 Consider the transition system

$$s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} s_3.$$

Observe that $s_0 \underline{\leftrightarrows}_{ro} s_1$, but *not* $s_1 \underline{\leftrightarrows}_{ro} s_2$, while $s_1 \underline{\leftrightarrows}_{rb} s_2$.

Divergence

A state s_0 has τ-divergence if it has an *infinite* path

$$s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \cdots .$$

Orthogonal bisimilarity does not always distinguish between states that have τ-divergence and states that have not. E.g., consider the transition system

$$\overset{\tau}{\curvearrowright}\ s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} s_3.$$

Observe that s_0 and s_1 are (rooted) orthogonally bisimilar, while s_0 has τ-divergence and s_1 has not. However, infinite τ-traces do not always collapse under (rooted) orthogonal bisimilarity, an example being

$$\overset{\tau}{\curvearrowright}\ s_0 \xrightarrow{a} s_3 \xleftarrow{a} s_2 \xleftarrow{\tau} s_1,$$

where $s_0 \not\underline{\leftrightarrows}_o s_1$ and $s_0 \not\underline{\leftrightarrows}_o s_2$. This implies that τ-divergence is a context-dependent phenomenon, and that from a semantic point of view, orthogonal bisimilarity is not optimal. For this reason we define a non-collapsing version for which τ-divergence is an invariant.

Definition 2.6 An orthogonal bisimulation R is *divergence sensitive*, if whenever sRr and s has τ-divergence, then r has τ-divergence.

States s and r are divergence sensitive orthogonally bisimilar, notation $s \underline{\leftrightarrows}_{dso} r$, if they are related by a divergence sensitive orthogonal bisimulation.

States s and r are *rooted* divergence sensitive orthogonally bisimilar, notation $s \underline{\leftrightarrows}_{rdso} r$, if they are related by a divergence sensitive orthogonal bisimulation that is rooted between s and r.

Of course, divergence sensitive orthogonal bisimilarity is strictly finer than orthogonal bisimilarity as such, and the same holds for the rooted versions.

3 Modal Characterization

Assume a fixed set P of proposition letters and a set L of labels with $\tau \notin L$. The set $\mathcal{L}$ of formulas ϕ is defined by the following BNF grammar.

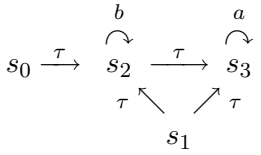
$$\phi ::= \ p \mid \langle\tau\rangle \mid \langle a\rangle\phi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathrm{U} \phi \quad (a \in L,\ p \in P)$$

We abbreviate the formula $\langle\tau\rangle \wedge \neg\langle\tau\rangle$ as $\bot$, and write $\top$ for $\neg\bot$, $\langle a\rangle$ for $\langle a\rangle\top$, and $\mathrm{F}\phi$ for $\top \mathrm{U} \phi$. We adopt the binding convention that $\langle a\rangle$, $\neg$ and F bind stronger than U, which binds stronger than $\wedge$.

Consider a transition system over L_τ with a valuation function V. Truth of a formula in a state s is defined inductively as follows.

- $s \models p$, if $p \in V(s)$.
- $s \models \langle\tau\rangle$, if $s \xrightarrow{\tau}$.
- $s \models \langle a\rangle\phi$, if $s \xrightarrow{a} s'$ and $s' \models \phi$ for some s'.
- $s \models \neg\phi$, if not $s \models \phi$.
- $s \models \phi \wedge \psi$, if $s \models \phi$ and $s \models \psi$.
- $s \models \phi \mathrm{U} \psi$, if, for some $n \geq 0$, there is a $s_0 \cdots s_n \in \tau\text{-}\mathit{paths}(s)$ such that $s_i \models \phi$ for all $i < n$ and $s_n \models \psi$.

Example 3.1 Consider the transition system below. All states satisfy the formula $\mathrm{F}\langle b\rangle \mathrm{U}\langle a\rangle$.



States s_0 and s_1 can reach the same states by τ-steps, namely s_2 and s_3. It holds that s_1 satisfies $\neg\langle b\rangle \mathrm{U}\langle a\rangle$, while s_0 does not. Observe that it is not possible to find a distinguishing formula using only future formulas $\mathrm{F}\phi$ instead of until formulas $\phi \mathrm{U} \psi$.

States s and r are *$\mathcal{L}$-equivalent*, notation $s \sim r$, if for all $\phi \in \mathcal{L}$ it holds that $s \models \phi$ if, and only if, $r \models \phi$.

Theorem 3.1 Consider a transition system over L_τ with any valuation. For all states s and r, it holds that $s \leftrightarrows_o r$ implies $s \sim r$.

Proof. Straightforward by induction on the structure of formulas (proof uses Lemma 2.1).

A transition system is finitely branching in label a, if all states have finitely many a-successors. A transition system is τ-path-image-finite if for all states s there are finitely many states s' with a path $s \cdots s' \in \tau\text{-}\mathit{paths}(s)$.

Lemma 3.1 If R is an orthogonal bisimulation with sRr and $s \xrightarrow{\tau} s'$, then there is a $r_0 \cdots r_n \in \tau\text{-}paths(r)$, for some $n \geq 0$, such that sRr_i for all $i < n$ and $s'Rr_n$, and for all $i, j \leq n$, if $i \neq j$ then $r_i \neq r_j$.

Theorem 3.2 Consider a transition system (S, L_τ, T) with valuation V that is τ-path-image-finite and finitely branching in every label. For all $s, r \in S$, it holds that $s \sim r$ implies $s \leftrightarrows_o r$.

Proof. We show that $\sim$ is an orthogonal bisimulation. Take any s, r with $s \sim r$. Clearly $V(s) = V(r)$.

1. If $s \xrightarrow{a} s'$ with $a \neq \tau$, then since $s \models \langle a \rangle$, also $r \models \langle a \rangle$. So, using that r is finitely branching in a, for some $n \geq 0$, r has a-successors $r_0, \ldots, r_n$. We have to show that, for some $i \leq n$, $s' \sim r_i$. Suppose that, for all $i \leq n$, $s' \not\sim r_i$. Then there is, for every $i \leq n$, a formula ϕ_i, such that $s' \models \phi_i$ and $r_i \not\models \phi_i$. Let $\phi = \langle a \rangle(\phi_0 \wedge \cdots \wedge \phi_n)$. We see that $s \models \phi$, whereas $r \not\models \phi$, which contradicts the assumption $s \sim r$. So $r \xrightarrow{a} r'$ for some r' with $s' \sim r'$.

2. If $s \xrightarrow{\tau} s'$ and $s' \sim s$, then $s' \sim r$ since $\sim$ is transitive, and $r \xrightarrow{\tau}$, since $s \models \langle \tau \rangle$ and hence $r \models \langle \tau \rangle$.

   So suppose that $s \xrightarrow{\tau} s'$ and $s' \not\sim s$. We must show that r can match this τ-step to s' appropriately. Suppose, to the contrary, that it cannot (1), i.e. that there is no $r_0 \cdots r_n \in \tau\text{-}paths(r)$ with $n > 0$ and $\forall i < n(s \sim r_i)$ and $s' \sim r_n$ and for all $i, j \leq n$, if $i \neq j$ then $r_i \neq r_j$. This last condition is justified by Lemma 3.1.

   Let $C \subseteq \tau\text{-}paths(r)$ be the set of sequences $r_0 \cdots r_n$ such that

   $$n \geq 0,\ \forall i \leq n(s \sim r_i),\ \forall i, j \leq n(r_i \neq r_j \vee i = j).$$

   The set C is finite because r is τ-path-image finite. It is nonempty because $r \in C$.

   By assumption (1), we see that for all $r \cdots r' \in C$ it holds that there is no r'' such that $r' \xrightarrow{\tau} r''$ and $s' \sim r''$.

   We define the set C' of extensions of paths in C as follows.

   $$C' = \{r \cdots r'r'' \mid r \cdots r' \in C,\ r' \xrightarrow{\tau} r'',\ r'' \not\sim s\}$$

   The set C' is finite because C is finite and the transition system is finitely branching in τ.

   Let χ be a formula such that $s' \models \chi$ and $s \not\models \chi$. Such a formula χ exists, because $s \not\sim s'$.

   It is straightforward to check that C' must be nonempty, since if it were empty then $r \not\models \mathrm{F}\chi$, whereas $s \models \mathrm{F}\chi$.

   So write $C' = \{\rho_0, \ldots, \rho_k\}$ for some $k \geq 0$. For all $\rho_i = r \cdots r_i \in C'$ it holds that $r_i \not\sim s$ and $r_i \not\sim s'$, and hence that there are formulas ϕ_i, ψ_i such that $s \models \phi_i$, $s' \models \psi_i$, $r_i \not\models \phi_i$ and $r_i \not\models \psi_i$. Let $\phi = \phi_0 \wedge \cdots \wedge \phi_k$ and $\psi = \psi_0 \wedge \cdots \wedge \psi_k$. Then $s \models \phi$, $s' \models \psi$ and for all $i \leq k$, $r_i \not\models \phi$ and $r_i \not\models \psi$.

We see directly that $s \models \phi \mathrm{U}(\psi \wedge \chi)$. We show that $r \not\models \phi \mathrm{U}(\psi \wedge \chi)$, which contradicts the assumption that $s \sim r$. Suppose that $r \models \phi \mathrm{U}(\psi \wedge \chi)$, i.e. that there is a τ-path $r_0 \cdots r_n$ with $r = r_0$ and $n \geq 0$, such that $r_n \models \psi \wedge \chi$ and $r_i \models \phi$ for all $i < n$ (2). We make the following observations:

(a) $n > 0$ because $r \not\models \chi$.

(b) $\forall i < n (r_i \sim s)$. Suppose not, then assume that j is the smallest $j < n$ with $r_j \not\sim s$. Then $r_0 \cdots r_j \in C'$ and so $r_j \not\models \phi$. Contradiction (2).

(c) $r_n \not\sim s$, since $s \not\models \chi$.

From these observations, it follows that $r_0 \cdots r_n \in C'$. Hence, $r_n \not\models \psi$. Contradiction. □

A formula for τ-divergence

We extend $\mathcal{L}$ with a special divergence formula Δ; let $\mathcal{L}_\Delta = \mathcal{L} \cup \{\Delta\}$. The satisfaction relation is defined as before, with $s \models \Delta$, if s has τ-divergence.

States s and r are $\mathcal{L}_\Delta$-*equivalent*, notation $s \sim_\Delta r$, if for all $\phi \in \mathcal{L}_\Delta$ it holds that $s \models \phi$ if, and only if, $r \models \phi$.

Theorem 3.3 Consider a transition system over L_τ with any valuation. For all states s and r, it holds that $s \leftrightarrows_{dso} r$ implies $s \sim_\Delta r$.

Proof. Trivial extension of the proof of Theorem 3.1.

Theorem 3.4 Consider a transition system over L_τ, with any valuation, that is τ-path-image-finite and finitely branching in every label. For all states s and r, it holds that $s \sim_\Delta r$ implies $s \leftrightarrows_{dso} r$.

Proof. Trivial extension of the proof of Theorem 3.2.

4 Process Algebra

We use *process algebra* because it provides an elegant notation for transition systems, and allows for axiomatic reasoning. The axiom system $\mathrm{ACP}(A, \gamma)$ [7] consists of the axioms in Table 1. The signature is determined by a finite set of constants A, the elements of which are called actions, and by a binary partial, commutative and associative function γ on A. The function γ defines synchronous communication between actions. We write a, b for arbitrary elements of A, and $\gamma(a, b) \downarrow$ if $\gamma(a, b)$ is defined.

The signature has a constant $\delta \notin A$ (deadlock). Furthermore, the signature has binary operators $+$ (alternative composition), $\cdot$ (sequential composition), $\|$ (parallel composition, merge), $\lfloor\!\lfloor$ (left merge) and $|$ (communication merge). It has a unary operator ∂_H (encapsulation) for every $H \subseteq A$. We write A_δ to denote the set $A \cup \{\delta\}$. We use infix notation for all binary operators, and adopt the binding convention that $+$ binds weakest and $\cdot$ binds strongest. We suppress $\cdot$, writing xy for $x \cdot y$.

Table 1: The axioms of $\mathrm{ACP}(A,\gamma)$, $a,b \in A_\delta$ and $H \subseteq A$.

(A1)	$x+y$	$=$	$y+x$	
(A2)	$x+(y+z)$	$=$	$(x+y)+z$	
(A3)	$x+x$	$=$	x	
(A4)	$(x+y)z$	$=$	$xz+yz$	
(A5)	$(xy)z$	$=$	$x(yz)$	
(A6)	$x+\delta$	$=$	x	
(A7)	δx	$=$	δ	
(CM1)	$x \mid\mid y$	$=$	$(x \between y + y \between x) + x \mid y$	
(CM2)	$a \between x$	$=$	ax	
(CM3)	$ax \between y$	$=$	$a(x \mid\mid y)$	
(CM4)	$(x+y) \between z$	$=$	$x \between z + y \between z$	
(CM5)	$ax \mid b$	$=$	$(a \mid b)x$	
(CM6)	$a \mid bx$	$=$	$(a \mid b)x$	
(CM7)	$ax \mid by$	$=$	$(a \mid b)(x \mid\mid y)$	
(CM8)	$(x+y) \mid z$	$=$	$x \mid z + y \mid z$	
(CM9)	$x \mid (y+z)$	$=$	$x \mid y + x \mid z$	
(CF1)	$a \mid b$	$=$	$\gamma(a,b)$	if $\gamma(a,b)\downarrow$
(CF2)	$a \mid b$	$=$	δ	otherwise
(D1)	$\partial_H(a)$	$=$	a	if $a \notin H$
(D2)	$\partial_H(a)$	$=$	δ	if $a \in H$
(D3)	$\partial_H(x+y)$	$=$	$\partial_H(x)+\partial_H(y)$	
(D4)	$\partial_H(xy)$	$=$	$\partial_H(x)\partial_H(y)$	

Table 2: Compression axioms, $a \in A_{\delta\tau}$ and $I \subseteq A$.

(O1)	$x\tau\tau$	$=$	$x\tau$	
(O2)	$x\tau(y+z)$	$=$	$x(y+z)$	if $\tau y = \tau\tau y,\ \tau z = \tau\tau z$
(O3)	$x(\tau(y+z)+z)$	$=$	$x(y+z)$	if $\tau y = \tau\tau y$
(TI1)	$\tau_I(a)$	$=$	a	if $a \notin I$
(TI2)	$\tau_I(a)$	$=$	τ	if $a \in I$
(TI3)	$\tau_I(x+y)$	$=$	$\tau_I(x)+\tau_I(y)$	
(TI4)	$\tau_I(xy)$	$=$	$\tau_I(x)\tau_I(y)$	

Subsystems of $\mathrm{ACP}(A,\gamma)$ are $\mathrm{BPA}(A)$, which consists of axioms A1−A5, and has sequential and alternative composition as operators, and $\mathrm{BPA}_\delta(A)$, the extension of $\mathrm{BPA}(A)$ with the deadlock process, axiomatized by A6 and A7. If E is any of these axiom systems, we write $\mathbb{T}(E)$ for its set of closed terms.

Semantics. We give a structural operational semantics for any of the theories presented. We define transition systems where closed terms are states. These transition systems have a special termination state; it is assumed that the valuation is such that only this special state has the termination marking $\surd$. Therefore, we use the symbol $\surd$ to denote the termination state.

Let E be one of $\mathrm{BPA}(A)$, $\mathrm{BPA}_\delta(A)$ and $\mathrm{ACP}(A,\gamma)$. Then $\mathrm{TS}(E)$ is the transition system

$$(\mathbb{T}(E)\cup\{\surd\},A,T),$$

where $\surd\notin\mathbb{T}(E)$ and the transition relation T is generated by the transition rules in Table 4. The transition rules are such that $\surd$ has no outgoing transitions. In this case we say that the transition system has *pure termination*, or shortly, that it is *pure*.

Strong bisimilarity is a congruence with respect to all operators defined. All theories presented so far are sound and complete with respect to strong bisimilarity. These are standard results, cf. e.g. [12].

The silent step

The axiom system $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$ consists of $\mathrm{ACP}(A,\gamma)$ and the axioms in Table 2. Its signature is obtained by extending the signature of $\mathrm{ACP}(A,\gamma)$ with the constant $\tau\notin A_\delta$, and with a unary operator τ_I for every $I\subseteq A$. Let $A_\tau=A\cup\{\tau\}$ and $A_{\delta\tau}=A_\delta\cup\{\tau\}$. In the axioms of Table 1, we now let a,b range over $A_{\delta\tau}$. In the semantics, we take A_τ as the set of transition labels. The subsystems $\mathrm{BPA}_\tau^{\mathrm{orth}}(A)$ and $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ are the respective extensions of $\mathrm{BPA}(A)$ and $\mathrm{BPA}_\delta(A)$ with τ and the axioms O1−O3. It is straightforward to verify that the axioms in Table 2 are sound with respect to rooted orthogonal bisimilarity.

The conditions in the axioms O2 and O3 are of the form $\tau x=\tau\tau x$. Such a condition holds for x, if, and only if, the process x does not equal deadlock and all initial actions of x equal τ.

Theorem 4.1 Rooted orthogonal bisimilarity is a congruence with respect to all operators of $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$.

Proof. See Appendix A.

Proposition 4.1 If t is a closed $\mathrm{BPA}_\tau^{\mathrm{orth}}(A)$ term that is built from τ's only, i.e. for all subterms $a\in A_\tau$ of t it holds that $a=\tau$, then t is derivably equal to exactly one of τ, $\tau\tau$ and $\tau\tau+\tau$.

Proof. Straightforward by induction on the structure of t. An interesting case is

$$\tau(\tau\tau+\tau)\overset{(\mathrm{A3})}{=}\tau(\tau(\tau+\tau)+\tau)\overset{(\mathrm{O3})}{=}\tau(\tau+\tau)\overset{(\mathrm{A3})}{=}\tau\tau.$$

Branching bisimilarity. Rooted branching bisimilarity is axiomatized by the axioms B1 and B2, see Table 3. In Section 2, we have seen that rooted branching bisimilarity is a coarser equivalence than rooted orthogonal bisimilarity. This is reflected in the strength of the axioms: it is straightforward to show that

$$\mathrm{B1}+\mathrm{B2}\vdash \mathrm{O1}{-}\mathrm{O3} \quad \text{and} \quad \mathrm{B1}+\mathrm{O3}\vdash \mathrm{B2}.$$

Table 3: Branching bisimilarity axioms.

(B1)	$x\tau$	$=$	x
(B2)	$x(\tau(y+z)+z)$	$=$	$x(y+z)$

5 Completeness

We prove that $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ is complete with respect to rooted orthogonal bisimilarity, i.e. any two rooted orthogonally bisimilar closed terms are derivably equal. The proof is based on Lemma 2.2 and the completeness of $\mathrm{BPA}_{\delta}(A)$. The completeness of $\mathrm{BPA}_{\tau}^{\mathrm{orth}}(A)$ can be proved similarly; this proof is omitted. We state that $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ is a conservative extension of $\mathrm{BPA}_{\tau}^{\mathrm{orth}}(A)$.

Via the completeness of $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$, it follows that $\mathrm{ACP}_{\tau}^{\mathrm{orth}}(A,\gamma)$ is complete: every closed $\mathrm{ACP}_{\tau}^{\mathrm{orth}}(A,\gamma)$ term is derivably equal to a closed $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ term. This elimination result is standard for ACP, and carries over to its orthogonal variant directly, as the special status of τ as an action does not interfere with the elimination. We state that $\mathrm{ACP}_{\tau}^{\mathrm{orth}}(A,\gamma)$ is a conservative extension of $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$.

Definition 5.1 $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ *basic terms* are defined inductively as follows.

1. The elements of $A_{\delta\tau}$ are basic terms.
2. If $a\in A_\tau$, and t is a basic term, then $a\cdot t$ is a basic term.
3. If t and u are basic terms, then $t+u$ is a basic term.

If $I=\{i_1,\ldots,i_n\}$ is a finite index set, then we write $\sum_{i\in I}t_i$ for the process term $t_{i_1}+\cdots+t_{i_n}$. As $+$ is associative, we do not write parentheses. We use the convention that $\sum_{i\in\emptyset}t_i=\delta$. Every basic term can, modulo axioms A1, A2 and A6, be written as

$$\sum_{i\in I}a_it_i+\sum_{j\in J}a_j,$$

where I,J are finite index sets such that for all $i\in I$ and $j\in J$, t_i is a basic term and $a_i,a_j\in A_\tau$.

Lemma 5.1 Every closed $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ term is derivably equal to a basic term.

Table 4: Transition rules.

$$a \xrightarrow{a} \surd \qquad \frac{x \xrightarrow{a} \surd}{xy \xrightarrow{a} y} \qquad \frac{x \xrightarrow{a} x'}{xy \xrightarrow{a} x'y} \qquad \frac{x \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd \quad y + x \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x' \quad y + x \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} \surd \quad a \notin H}{\partial_H(x) \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{a} y \quad a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(y)}$$

$$\frac{x \xrightarrow{a} \surd}{x \between y \xrightarrow{a} y \quad x \,||\, y \xrightarrow{a} y \quad y \,||\, x \xrightarrow{a} y}$$

$$\frac{x \xrightarrow{a} x'}{x \between y \xrightarrow{a} x' \,||\, y \quad x \,||\, y \xrightarrow{a} x' \,||\, y \quad y \,||\, x \xrightarrow{a} y \,||\, x'}$$

$$\frac{x \xrightarrow{a} \surd \quad y \xrightarrow{b} \surd \quad \gamma(a,b) = c}{x \,||\, y \xrightarrow{c} \surd \quad x \mid y \xrightarrow{c} \surd} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y' \quad \gamma(a,b) = c}{x \,||\, y \xrightarrow{c} x' \,||\, y' \quad x \mid y \xrightarrow{c} x' \,||\, y'}$$

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} \surd \quad \gamma(a,b) = c}{x \,||\, y \xrightarrow{c} x' \quad x \mid y \xrightarrow{c} x' \quad y \,||\, x \xrightarrow{c} x' \quad y \mid x \xrightarrow{c} x'}$$

$$\frac{x \xrightarrow{a} \surd \quad a \notin I}{\tau_I(x) \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{a} \surd \quad a \in I}{\tau_I(x) \xrightarrow{\tau} \surd} \qquad \frac{x \xrightarrow{a} y \quad a \notin H}{\tau_I(x) \xrightarrow{a} \tau_I(y)} \qquad \frac{x \xrightarrow{a} y \quad a \in H}{\tau_I(x) \xrightarrow{\tau} \tau_I(y)}$$

Proof. Straightforward.

Lemma 5.2 If $t = \sum_{i \in I} \tau t_i$ for some nonempty finite set I, then $\tau t = \tau\tau t$.

Proof. Using induction on $|I|$ and axioms O1 and O2.

Lemma 5.3 If $t = \sum_{i \in I} \tau t_i + t'$ for some nonempty finite index set I, with t_i compact and $t \leftrightarroweq_o t_i$ for all $i \in I$, then $t = \tau t_i + t'$ for any $i \in I$.

Proof. Take any $i, j \in I$. Since $\leftrightarroweq_o$ is an equivalence, we have $t_i \leftrightarroweq_o t_j$. Since t_i, t_j are compact, we have by Lemma 2.2 that $t_i \leftrightarroweq t_j$. By the completeness of BPA$_\delta$ we get $t_i = t_j$. The required identity follows by axiom A3. □

Lemma 5.4 Every closed $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ term is derivably equal to a basic term that has only compact successors.

Proof. Take any closed term t. By Lemma 5.1 we may assume that t is a basic term. We apply induction on the structure of t. If $t \equiv \delta$, then it has no successors. If $t \in A_\tau$, then its only successor is $\surd$, which is compact. If $t \equiv t' + t''$, then the proof is immediate using the induction hypothesis.

So assume that $t \equiv at'$. We have by induction hypothesis that $t' = u$ for some basic term u with compact successors. The term u has a compact part and an inert part; the term u is, modulo A1, A2 and A6 of the form $\sum_{i \in I} \tau u_i + u^c$, where

$$u^c = \sum_{j \in J} a_j u_j + \sum_{k \in K} a_k;$$

$u \leftrightarroweq_o u_i$ for all $i \in I$; $a_j \in A_\tau$ and $u \not\leftrightarroweq_o u_j$ for all $j \in J$; $a_k \in A_\tau$ for all $k \in K$.

The processes u_i and u_j are compact. We show that au is derivably equal to a term with compact successors. Take any $i \in I$ (if $I = \emptyset$, then u itself is compact). By Lemma 5.3, it holds that $u = \tau u_i + u^c$.

We know that u_i is compact and that $u \leftrightarroweq_o u_i$. From these two facts, it is straightforward to verify that u_i must be a summation consisting of the following summands.

1. For every $k \in K$, one or more summands a_k. By axiom A3, we may assume that there is exactly one summand a_k for every $k \in K$.

2. For every $j \in J$, one or more summands $a_j u'_j$ with $u_j \leftrightarroweq_o u'_j$. By Lemma 2.2 and the completeness of BPA$_\delta$, we have that $u'_j = u_j$ for all u'_j. By these identities and by axiom A3, we may assume that there is exactly one summand $a_j u_j$ for every $j \in J$.

3. For every l in some finite index set L, a summand τu_l, with $u_i \not\leftrightarroweq_o u_l$. We assume that L is nonempty; if it is not, then infer from $u \leftrightarroweq_o u_i$ and the fact that u has a τ-transition (to u_i), that there must be a $j \in J$ with $a_j = \tau$. In this case use axiom A3 to double a summand $a_j u'_j$ with such a j, thereby producing a summand τu_l.

Finally, we get that $u_i = \sum_{l \in L} \tau u_l + u^c$.

Combining $u = \tau u_i + u^c$, Lemma 5.2 and axiom O3, we find that $au = au_i$, where the right-hand side has compact successors. □

Theorem 5.1 The axiom system $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$ is complete with respect to $\leftrightarrows_{ro}$, i.e. if t and u are closed terms and $t \leftrightarrows_{ro} u$, then $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau} \vdash t = u$.

Proof. Assume that $t \leftrightarrows_{ro} u$. By Lemma 5.4 and soundness we may assume that all successors of t and u are compact. By Corollary 2.1 we have that they are strongly bisimilar. BPA_{δ} is complete with respect to strong bisimilarity. □

Corollary 5.1 The axiom system $\mathrm{ACP}^{\mathrm{orth}}_{\tau}(A, \gamma)$ is complete with respect to $\leftrightarrows_{ro}$.

6 Priorities

We extend $\mathrm{ACP}^{\mathrm{orth}}_{\tau}(A,\gamma)$ with the priority operator θ, introduced in ACP in [1]. Parameter of θ is a partial ordering $\leq$ on A_τ(we write $a < b$ or $b > a$ if $a \leq b$ and $a \not\equiv b$). If, e.g., the priority ordering is given by $a > b$ and $a > c$, then a has priority over b and over c. In this case, we have that $\theta(a + b) = a$ and $\theta(b + c) = b + c$. The priority operator can be used to model interrupts in a distributed system; it is used as such in the specification of a PAR protocol in Section 9.

The transition rules for θ are in Table 5. For the axiomatization, we need the auxiliary operator $\lhd$ ("unless"). The axioms are in Table 6. A process $p \lhd q$ behaves as the part of p that has initial actions that do not have an initial action with higher priority in q.

We give an example derivation. Suppose that $a > b$.

$$\begin{aligned} \theta(ax + by) &= \theta(ax) \lhd by + \theta(by) \lhd ax \\ &= (a \lhd b) \cdot \theta(x) + (b \lhd a) \cdot \theta(y) \\ &= a \cdot \theta(x) + \delta \\ &= a \cdot \theta(x) \end{aligned}$$

The following example shows that, in the setting with τ, the priority operator is not a congruence for the abstract process equivalences in [15].

Example 6.1 Let the priority ordering by given by $c < b$. Consider $t \equiv a(\tau(b + c) + c)$ and $u \equiv a(b + c)$. These processes are rooted branching bisimilar, and hence identified by all process equivalences in [15]. Observe that none of these equivalences identifies $\theta(t) = a(\tau b + c)$ and $\theta(u) = ab$.

In this example, the process t performs a and reaches the process $\tau(b + c) + c$. This process has a *direct* option to execute c, and a *blind* option to execute b; the τ is hiding the option for b. In orthogonal bisimulation equivalence, a nondirect option can never become direct: orthogonally bisimilar processes have exactly the same direct options.

In Appendix A, it is proved that rooted orthogonal bisimilarity is a congruence with respect to the priority operator in the setting with τ. We state without proof that the priority axioms are sound and that $\mathrm{ACP}^{\mathrm{orth}}_{\tau\theta}(A, \gamma)$ is a conservative extension of $\mathrm{ACP}^{\mathrm{orth}}_{\tau}(A, \gamma)$. Completeness follows from the fact that θ can be eliminated from terms, which is easy to verify.

Table 5: Transition rules for the priority operator.

$$\frac{x \xrightarrow{a} \surd \quad \neg\exists b > a.\ x \xrightarrow{b}}{\theta(x) \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{a} x' \quad \neg\exists b > a.\ x \xrightarrow{b}}{\theta(x) \xrightarrow{a} \theta(x')}$$

$$\frac{x \xrightarrow{a} \surd \quad \neg\exists b > a.\ y \xrightarrow{b}}{x \lhd y \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{a} x' \quad \neg\exists b > a.\ y \xrightarrow{b}}{x \lhd y \xrightarrow{a} x'}$$

Table 6: Priority axioms, $a, b \in A_\tau$.

(P1)	$a \lhd b$	$=$	a	if $a \not< b$
(P2)	$a \lhd b$	$=$	δ	if $a < b$
(P3)	$x \lhd yz$	$=$	$x \lhd y$	
(P4)	$x \lhd (y + z)$	$=$	$(x \lhd y) \lhd z$	
(P5)	$xy \lhd z$	$=$	$(x \lhd z)y$	
(P6)	$(x + y) \lhd z$	$=$	$x \lhd z + y \lhd z$	
(TH1)	$\theta(a)$	$=$	a	
(TH2)	$\theta(xy)$	$=$	$\theta(x)\theta(y)$	
(TH3)	$\theta(x + y)$	$=$	$\theta(x) \lhd y + \theta(y) \lhd x$	

Note. Various ways for dealing with the priority operators in abstract semantics were proposed. A first, classical approach is to eliminate all priority operations *before* applying abstraction. Another approach was advocated by Bol and Groote in [11], where the unless operator is equipped with a "look-ahead" facility for τ-steps. Both these approaches are not fully general, in the sense that they do not admit that τ (freely) enters the priority ordering. Although it may in some cases be questionable whether τ should be given a priority, this is not in any technical sense problematic. This last fact can be characterized as follows: write $a \equiv_{pr} b$ if a and b have the same priority in some give priority ordering. Now assume I is a set of internal actions, and $\tau \equiv_{pr} i$ for all $i \in I$. Then we have that τ_I and θ commute modulo orthogonal bisimilarity, i.e.,

$$\theta \circ \tau_I(x) = \tau_I \circ \theta(x),$$

which is the strongest commutation result that can be expected.

7 Binary Kleene Star, Push-Down and Fairness

In process algebra, potentially infinite behaviors are usually characterized by means of recursive equations. As an example, the equation

$$X = aX$$

characterizes the process that can perform an infinite sequence of a-steps only, and so do the equations $Y = aYb$ and $Z = aaZ$ (and many more). Recently, a different approach to the specification of such behaviors attracted attention, namely the use of recursive *operations* [6, 9]. As the most basic of these we consider the *binary Kleene star* operator $(*)$, defined by

BKS1 $\quad x^*y = x(x^*y) + y.$

For example, $a^*\delta$ expresses the process mentioned above, and so does $(aa)^*\delta$. We adopt the convention that $\cdot$ and $*$ bind equally strong.

In the setting of $\mathrm{BPA}(A)$, axioms for the binary Kleene star are BKS1, BKS2 and BKS3 from Table 7. If E is any of the axiom systems discussed in the previous sections, we write E^* for its extension with the appropriate axioms on the binary Kleene star. In [13] it is shown that $\mathrm{BPA}^*(A)$ axiomatizes bisimilarity over that signature. The system $\mathrm{ACP}^*(A,\gamma)$ is defined by adding BKS1 − BKS4. In the setting with τ and $*$, the system $\mathrm{BPA}_\tau^{\mathrm{orth}*}$ is defined by extending $\mathrm{BPA}^*(A)$ with the axioms O1−O3 (see Table 2) and the axioms O4 and O5 given in Table 7 below. (Note that these last two axioms are easily proved valid in orthogonal bisimulation equivalence.) Finally, the system $\mathrm{ACP}^{\mathrm{orth}*}(A,\gamma)$ is defined by adding all axioms from Table 7 to $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$.

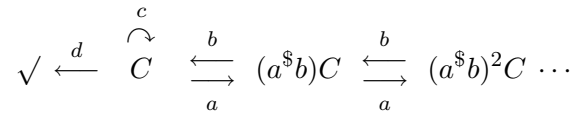
The transition rules for $*$ are as expected, and given in Table 8. Observe that each closed term over one of the systems with $*$ has finitely many substates, where substates are those terms that can be reached by transitions. This reveals the limited expressiveness of the above-mentioned systems with $*$: only finite state processes are definable. This restriction can be relaxed by adding the *push-down* operation (\$, see [9]), defined by

PD $\quad x^{\$}y = x((x^{\$}y)(x^{\$}y)) + y.$

We adopt the convention that $\cdot$ and \$ bind equally strong. Transition rules for \$ are as expected, and given in Table 8. We write $E^{\$}$ for the inclusion of axiom PD in axiom system E. With \$ also non-regular processes can de defined. A typical example is

$$(a(a^{\$}b) + c)^*d.$$

This term can be recognized as a definition of a *counter* or *register*, modelling a memory location for a natural number with unbounded capacity and restricted access by the actions: a ("add one"), b ("subtract one"), c ("test zero"), and d ("clear location"). Abbreviating $(a(a^{\$}b) + c)^*d$ to C, a graphical representation of C is given below.



In $\mathrm{ACP}^{*\$}(A,\gamma)$, registers can synchronize with terms representing register machine programs. As an example, let $H = \{x, \overline{x} \mid x = a, b, c, d\}$ and $\gamma(x, \overline{x}) = i$ for $x = a, b, c, d$. Then termination of register C in case it holds value 2, characterized by the state $(a^{\$}b)^2C$, can be described by $\partial_H((a^{\$}b)^2C \,||\, \overline{b}^*\overline{d}) = i^3$, and emptying register C can be described by $\partial_H((a^{\$}b)^2C \,||\, \overline{b}^*\overline{c})$. For indexed registers C_1 and C_2, transfer of the value of C_1 to C_2 is

'programmed' by $(\overline{b}_1\overline{a}_2)^*\overline{c}_1$. In the following section we return to the issue of expressivity, and we shall use register machine computation in a style as suggested above.

In settings with δ, there are no finite equational axiomatizations of the binary Kleene star operator. Therefore we provide the following adaptation of RSP, the Recursive Specification Principle.

RSP^* if $x = yx + z$ and $\partial_A(y) = \delta$, then $x = y^*z$

Here the second condition acts as a "guardedness restriction"; it excludes terms with an initial τ action. E.g., we cannot infer $\tau\tau a = \tau^*\delta$, although $\tau\tau a = \tau\tau\tau a + \delta$ is valid. For the push-down operator there is a similar adaptation of RSP [9, 10], but we shall not use it.

Divergence. Due to the character and common use of τ, one may want to abstract from infinite sequences or loops consisting only of τ steps. Depending on the kind of process semantics one wants to use, different solutions have been found. In the case of rooted branching bisimulation equivalence, a particular solution is provided by

FIR_1^{b} $\tau(\tau^*x) = \tau x$,

where FIR abbreviates Fair Iteration Rule. In the setting of rooted orthogonal bisimulation equivalence, we have the 'fairness axioms' given in Table 9. In Section 9 we provide a protocol verification in which fairness is used.

If we consider processes modulo rooted divergence sensitive orthogonal bisimilarity, then of course the axioms OFIR1 and OFIR2 are no longer valid.

Table 7: The binary Kleene star axioms.

(BKS1)	x^*y	$=$	$x(x^*y) + y$
(BKS2)	$x^*(yz)$	$=$	$(x^*y)z$
(BKS3)	$(x+y)^*z$	$=$	$x^*(y((x+y)^*z) + z)$
(BKS4)	$\partial_H(x^*y)$	$=$	$\partial_H(x)^*\partial_H(y)$
(BKS5)	$\tau_I(x^*y)$	$=$	$\tau_I(x)^*\tau_I(y)$
(O4)	$x((\tau\tau)^*y)$	$=$	$x(\tau^*y)$ if $\tau y = \tau\tau y$
(O5)	$x((\tau + \tau\tau)^*y)$	$=$	$x((\tau\tau)^*y)$

8 Expressiveness

In this section we consider some basic expressiveness questions: which sort of transition systems can be expressed in which of the axiom systems discussed before? In order to handle these questions we restrict to transition systems that are pure, i.e., transitions systems with a (single) termination state $\surd$ not having outgoing transitions, and with at least one other state (different from $\surd$, see Section 4). Expressing a pure transition system $\mathcal{T}$ up to some behavioral equivalence $\sim$ in axiom system E comes down to showing that for each state s in $\mathcal{T}$ different from $\surd$ there is a term t over E satisfying $s \sim t$.

Table 8: Transition rules for binary Kleene star and push-down.

$$\frac{x \xrightarrow{a} \surd}{x^*y \xrightarrow{a} x^*y} \qquad \frac{x \xrightarrow{a} x'}{x^*y \xrightarrow{a} x'(x^*y)} \qquad \frac{y \xrightarrow{a} \surd}{x^*y \xrightarrow{a} \surd} \qquad \frac{y \xrightarrow{a} y'}{x^*y \xrightarrow{a} y'}$$

$$\frac{x \xrightarrow{a} \surd}{x^{\$}y \xrightarrow{a} (x^{\$}y)(x^{\$}y)} \qquad \frac{x \xrightarrow{a} x'}{x^{\$}y \xrightarrow{a} x'((x^{\$}y)(x^{\$}y))} \qquad \frac{y \xrightarrow{a} \surd}{x^{\$}y \xrightarrow{a} \surd} \qquad \frac{y \xrightarrow{a} y'}{x^{\$}y \xrightarrow{a} y'}$$

Table 9: Fairness axioms.

(OFIR1)	$x(\tau^*(y+\tau z))$	$=$	$x(y+\tau z)$
(OFIR2)	$x(\tau^*(y+\tau))$	$=$	$x(y+\tau)$

In [2], Baeten, Bergstra and Klop proved the following basic expressiveness result: each recursive pure transition system (or 'process graph') can be expressed up to rooted τ-bisimilarity in ACP with abstraction and finite, guarded recursive specifications. Furthermore, these authors showed that abstraction is necessary for this result. Here a *recursive* transition system is one that has a recursive set of states, a finite set of labels, and a transition relation that can be characterized by a recursive function (describing for each state its finite number of transitions in terms of an appropriate encoding). The proof of this expressiveness result carries over to branching bisimulation equivalence, but not to any of the orthogonal bisimulation equivalences defined in this paper. The main reason for this mismatch is the role of the law $x = x\tau$. In order to study expressiveness questions in the setting of orthogonal bisimilarity, it therefore seems reasonable to enrich transition systems with τ's in the following way: given a transition system $\mathcal{T} = (S, L, T)$, its *sequential τ-saturation* $\mathcal{T}_\tau$ is defined by (S_τ, L_τ, T_τ) where

- $S_\tau = \{s, s_\tau \mid s \in S\}$ (and $s \in S \Rightarrow s_\tau \notin S$),
- $L_\tau = L \cup \{\tau\}$,
- $T_\tau = \{s \xrightarrow{a} t_\tau, t_\tau \xrightarrow{\tau} t \mid s \xrightarrow{a} t \in T\}$.

We view binary Kleene star and push-down as a modern alternative to "finite guarded recursive specifications" as used in the expressiveness result in [2]. First, we prove in detail that we can express the sequential τ-saturation of any finite pure transition system with labels in $L \subseteq A$ up to rooted divergence sensitive orthogonal bisimulation equivalence in $\mathrm{ACP}^{\mathrm{orth}*}(A, \gamma)$, provided A is sufficiently large. Next, we argue that any recursive pure transition system with finite label set $L \subseteq A$ and bounded fan-out can be expressed in $\mathrm{ACP}_\tau^{\mathrm{orth}*\$}(A, \gamma)$ up to rooted orthogonal bisimulation equivalence, for a suitable, finite set A of actions.

Theorem 8.1 For each finite pure transition system $\mathcal{T}$ with finite label set L not containing τ, there is a finite extension A of L such that $\mathcal{T}_\tau$ can be expressed up to rooted divergence sensitive orthogonal bisimulation equivalence in $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$, using only handshaking over $A \setminus L$, and either $*$ or \$.

Proof. Assume that $\mathcal{T}$ has states $\{\surd, X_1, ..., X_n\}$ for some $n > 0$. Then for every j with $0 < j \leq n$, X_j can be characterized by

$$X_j = \sum_{k=1}^{n} \alpha_{j,k} X_k + \beta_j$$

with $\alpha_{j,k}$ and β_j finite sums of actions or δ in the following way: for each transition $X_j \xrightarrow{a} X_k$ there is a summand a in $\alpha_{j,k}$ and for each transition $X_j \xrightarrow{b} \surd$ there is a summand b in β_j, and conversely, each summand of $\alpha_{j,k}$ and β_j is associated with a transition. If there are no transitions with source X_j and target X_k ($\surd$), then $\alpha_{j,k}$ (β_j, respectively) equals δ. As a consequence, $\mathcal{T}_\tau$ can be characterized by $X_j = \sum_{k=1}^{n} \alpha_{j,k}\tau X_k + \beta_j \tau$.

We define process terms that mimick the transitions of $\mathcal{T}_\tau$. Let A be the extension of L with the following $2n+3$ fresh actions:

$$i, \text{ and } r_l, s_l \qquad (l = 0, 1, ..., n).$$

Let $\gamma(r_l, s_l) = i$ be the only communications defined (handshaking). As to provide some intuition, these actions model the following behavior:

- s_0: order termination,
- r_0: receive the order to terminate,
- s_l: ($l > 0$) instruct the l-th process to start, and
- r_l: ($l > 0$) read instruction to start the l-th process.

Let $H = \{r_l, s_l \mid l = 0, 1, ..., n\}$, and for $j = 1, ..., n$,

$$F_j = \sum_{k=1}^{n} \alpha_{j,k} s_k + \beta_j.$$

In the case of $*$, consider the following process terms:

$$\begin{array}{lcl} Q & = & (\sum_{k=1}^{n} r_k F_k)^* s_0, \\ M & = & (\sum_{k=1}^{n} r_k s_k)^* r_0. \end{array}$$

We derive:

$$\begin{array}{lcl} \partial_H(F_j Q \,||\, M) & = & \partial_H([(\sum_{k=1}^{n} \alpha_{j,k} s_k) Q + \beta_j Q] \,||\, M) \\ & = & \sum_{k=1}^{n} \alpha_{j,k} \cdot \partial_H(s_k \cdot Q \,||\, M) + \beta_j \cdot \partial_H(Q \,||\, M) \\ & = & \sum_{k=1}^{n} \alpha_{j,k} \cdot i \cdot \partial_H(Q \,||\, s_k \cdot M) + \beta_j \cdot i \\ & = & \sum_{k=1}^{n} \alpha_{j,k} \cdot i \cdot i \cdot \partial_H(F_k \cdot Q \,||\, M) + \beta_j \cdot i. \end{array}$$

Consequently, $\tau_{\{i\}} \circ \partial_H(F_j Q \,||\, M)$ satisfies the identities for X_j ($j = 1, ..., n$) up to rooted divergence sensitive orthogonal bisimilarity. Hence, $\mathcal{T}_\tau$ can be expressed in $\mathrm{ACP}^{\mathrm{orth}*}(A,\gamma)$: for each state X_j of $\mathcal{T}_\tau$,

$$X_j \leftrightarrows_{rdso} \tau_{\{i\}} \circ \partial_H(F_j Q \,||\, M)$$

in $\mathrm{TS}(\mathrm{ACP}^{\mathrm{orth}*}(A,\gamma)) \cup \mathcal{T}_\tau$ (with single termination state $\surd$).

In the case of \$, consider process terms

$$\begin{array}{lcl} K & = & (\sum_{k=1}^{n} r_k F_k)^{\$} s_0, \\ N & = & (\sum_{j=1}^{n} r_k s_k)^{\$} r_0. \end{array}$$

Then $X_j \leftrightarrows_{rdso} \tau_{\{i\}} \circ \partial_H(F_j K \,||\, N)$ for each $j = 1, ..., n$. This can be shown along the same lines, using a denumerable infinity of copies of the transitions of $\mathcal{T}_\tau$: let l range over $\mathbb{N}$ (the naturals) and consider

$$Y_j(l) = \sum_{k=1}^{n} \alpha_{j,k} \tau Y_k(l+1) + \beta_j \tau.$$

Clearly, $X_j \leftrightarrows_{rdso} Y_j(l)$ for each state X_j of $\mathcal{T}_\tau$ and each value of l. So it suffices to show that also $\tau_{\{i\}} \circ \partial_H(F_j K \,||\, N) \leftrightarrows_{rdso} Y_j(0)$. We show this by first omitting the $\tau_{\{i\}}$-application:

$$\begin{array}{rcl} \partial_H(F_j \cdot K^{k+1} \,||\, N^{k+1}) & = & \sum_{k=1}^{n} \alpha_{j,k} \cdot \partial_H(s_k K^{k+1} \,||\, N^{k+1}) + \beta_j \cdot \partial_H(K^{k+1} \,||\, N^{k+1}) \\ & = & \sum_{k=1}^{n} \alpha_{j,k} \cdot i \cdot \partial_H(K^{k+1} \,||\, s_k N^{k+2}) + \beta_j \cdot i^{k+1} \\ & = & \sum_{k=1}^{n} \alpha_{j,k} \cdot i^2 \cdot \partial_H(F_k \cdot K^{k+2} \,||\, N^{k+2}) + \beta_j \cdot i^{k+1}. \end{array}$$

Hence, applying $\tau_{\{i\}}$ and axiom $x\tau\tau = x\tau$ (O1), we find for each k

$$\tau_{\{i\}} \circ \partial_H(F_j \cdot K^{k+1} \,||\, N^{k+1}) = \sum_{k=1}^{n} \alpha_{j,k} \tau \cdot \tau_{\{i\}} \circ \partial_H(F_k \cdot K^{k+2} \,||\, N^{k+2}) + \beta_j \tau,$$

which shows that $\tau_{\{i\}} \circ \partial_H(F_j \cdot K^{k+1} \,||\, N^{k+1}) \leftrightarrows_{rdso} Y_j(k)$. □

The above result shows that each *regular* process can be defined modulo sequential τ-saturation and rooted divergence sensitive orthogonal bisimilarity in $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$, provided we adopt (at least) one of $*$ and \$, and A is sufficiently large (but finite). For *non-regular*, computable processes (that is, processes that can be characterized by a recursive pure transition system) we have the following expressiveness result: the sequential τ-saturation of a recursive pure transition system with (finite) label set $L \subseteq A$ and bounded fan-out can be expressed in $\mathrm{ACP}_\tau^{\mathrm{orth}*\$}(A,\gamma)$ and $\mathrm{ACP}_\tau^{\mathrm{orth}\$}(A,\gamma)$ up to rooted divergence sensitive orthogonal bisimulation equivalence, provided A is sufficiently large. For example, one can express the sequential τ-saturation of a stack over a finite data type using the approach in [9].

We sketch a proof of the expressibility of pure recursive transition systems with bounded fan-out. This proof is based on a characterization of register machine computations (see, e.g., [24]) in process algebra (a detailed explanation can be found in [10]). As has been shown in the previous section, registers have a straightforward representation in $\mathrm{BPA}^{*\$}(A)$. Furthermore, each register machine *program* has a straightforward representation in $\mathrm{BPA}^{*}(A)$. It easily follows that each (unary) recursive function f can be 'implemented' in $\mathrm{ACP}^{*\$}(A,\gamma)$ in the following sense: let P represent in $\mathrm{BPA}^{*}(A)$ a register machine program P_f that computes f using three registers, and write $C(0)$ for $(a(a^{\$}b) + c)^* d$ (a register containing value 0) and $C(n+1)$ for $(a^{\$}b)C(n)$. Then there is a context $\mathsf{Con}[\_](n)$ where n refers to the register value $C(n)$ such that

$$\mathsf{Con}[Px](n)] \leftrightarrows \begin{cases} i^{g(n)} \cdot \mathsf{Con}[x](f(n)) & \text{if } f(n) \text{ is defined,} \\ i^* \delta & \text{otherwise.} \end{cases}$$

Here g is a computable function defined on the domain of f, and the i-steps result from communications between the registers and the program. Furthermore, $\mathsf{Con}[\_](n)$ can be extended to $\mathsf{Con}[\_](n_1, ..., n_k)$ in order to compute $k > 1$ computable functions in a sequential fashion:

$$\mathsf{Con}[P_1...P_k x](n, 0, ..., 0) \leftrightarrows i^{g'(n)} \cdot \mathsf{Con}[x](f_1(n), ..., f_k(n))$$

if each f_i is defined on n, and computed by register machine program P_i.

Now let $\mathcal{T} = (S, L, T)$ be a recursive pure transition system with $S \subseteq \mathbb{N}$ and fan-out bounded by N. With the above implementation scheme at hand, it is not hard to express the sequential τ-saturation of $\mathcal{T}$ up to rooted divergence sensitive orthogonal bisimilarity in $\mathrm{ACP}_\tau^{\mathrm{orth}*\$}(A, \gamma)$. A possible approach is the following. Given some state, let its *menu* be a characterization of the labels of all its outgoing transitions or its termination status (i.e., no outgoing transitions and either $\surd$ or deadlock). Fix an enumeration of these menus. Let furthermore the transition relation T be characterized by $(N+1)$-tuples fetching all outgoing transitions (at most N): a state s yields the map

$$(s, 0, ..., 0) \mapsto (s_1, ..., s_{N+1})$$

where

- s_{N+1} gives the menu value of s,
- s_i for $i = 1, ..., N$ is the target associated with source s and the i-th label of menu s_{N+1} if such a transition is present, and 0 otherwise.

By the recursiveness of $\mathcal{T}$, the above $N + 1$ functions that define $\mapsto$ can be computed by some register program P, thus

$$\mathsf{Con}[Px](s, 0, ..., 0) \xrightarrow{i}\!\!\!\!\rightarrow \mathsf{Con}[x](s_1, ..., s_{N+1}),$$

where $\_ \xrightarrow{i}\!\!\!\!\rightarrow \_$ abbreviates the transitive closure of $\_ \xrightarrow{i} \_$. Furthermore, assuming that we use menu value 0 for successful termination and 1 for deadlock, it is straightforward to define a process term Q that interprets the menu value s_{N+1} in the following way:

$$\mathsf{Con}[Qx](s_1, ..., s_N, s_{N+1}) \xrightarrow{i}\!\!\!\!\rightarrow \begin{cases} \mathsf{Con}[\delta x](s_1, ..., s_N, 0) & \text{if } s_{N+1} = 1, \\ \mathsf{Con}[(\sum_{j \in J} a_j E_j)x](s_1, ..., s_N, 0) & \text{if } s_{N+1} > 1. \end{cases}$$

Here the a_j's are prescribed by the menu value and E_j is a process that transfers s_j to the first position, and empties all other registers. It follows that the full computation of all transitions from s is captured by

$$\mathsf{Con}[P((QP)^* \overline{d}_{N+1} R)](s, 0, ..., 0),$$

where $\overline{d}_{N+1}$ synchronizes with the termination action d_{N+1} of the menu register in case it holds value 0 and R terminates all remaining processes. Finally, applying $\tau_{\{i\}}$ we can express $\mathcal{T}_\tau$ up to rooted divergence sensitive orthogonal bisimilarity: for each $s \in S \subseteq S_\tau$ it follows that if s has transitions $s \xrightarrow{a_j} (s_j)_\tau \xrightarrow{\tau} s_j$ for $j \in J$, then

$$s \leftrightarrows_{rdso} \tau_{\{i\}}(\mathsf{Con}[(\textstyle\sum_{j \in J} a_j E_j) P((QP)^* \overline{d}_{N+1} R)](s_{j1}, s_{j2}, ..., s_{jN}, 0))$$

in the combined transition system. In the case that s has no transitions, $s \leftrightarrows_{rdso} \delta$. In Appendix B we provide some specific examples. Following the proof of Theorem 8.1 above, it is straightforward how this approach should be adapted to $\mathrm{ACP}_\tau^{\mathrm{orth}\$}(A,\gamma)$ (thus, without $*$, cf. the related results in [9, 10]). The above can be summarized as follows:

Theorem 8.2 For each recursive pure transition system $\mathcal{T}$ with finite label set L not containing τ and bounded fan-out, there is a finite extension A of L such that $\mathcal{T}_\tau$ can be expressed up to rooted divergence sensitive orthogonal bisimulation equivalence in $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$, using only handshaking over $A \setminus L$, and either \$, or both $*$ and \$.

We note that for each term over $\mathrm{ACP}_\tau^{\mathrm{orth}*\$}(A,\gamma)$ or $\mathrm{ACP}_\tau^{\mathrm{orth}\$}(A,\gamma)$, its fan-out and that of all its substates is bounded by its complexity. This implies that a stronger expresiveness result is not possible. An essential unbounded fan-out (i.e., each bisimilar system also has an unbouded fan-out) is not expressible by a (finitary) process term.

9 Verification of a PAR Protocol

We prove the correctness of a Positive Acknowledgement and Retransmission (PAR) protocol [26]. The architecture of the protocol is depicted in Figure 1. We refer to [27] for an earlier verification in process algebra.

The sender S reads a datum from the environment, and sends this datum, accompanied by a bit, to the receiver R via channel K. The receiver has just one (thus *positive*) acknowledgement for the arrival of a frame. The receiver sends its acknowledgements via channel L to S. The channels are unreliable; upon receiving a datum the channel K can do one of three things: it can pass on the datum, it can loose the information but send an error message instead, and it can fail to do anything. The channel L either passes on the acknowledgement, or fails to do anything. A dummy internal action i is added to make the choice between these options non-deterministic.

The sender and the receiver act in response to received data only. This poses a problem, because, when either of the channels K and L fails to act upon receiving a message, the system will be waiting for nothing to happen, i.e. it will deadlock. In order to avoid this, the *time-out* action to may occur, that is supposed to reactivate the system if it threatens to deadlock. In [26], this time-out is issued by a timer, that is started by the sender at the moment it sends a frame to K. Here, we model the time-out interruption by placing the system in the scope of a priority operator and giving the action to lower priority than any other action; it occurs if no other activity is possible. After a time-out, the sender retransmits the last message.

Specification. We assume a finite data set D and a set of frames $F = D \times \{0,1\}$. The components are specified as follows.

$$\begin{aligned} S &= (S_0 \cdot S_1)^* \delta \\ S_n &= \textstyle\sum_{d\in D} r(d) \cdot s_0(dn) \cdot S_{dn} \\ S_{dn} &= (\mathsf{to} \cdot s_0(dn))^* r_3 \end{aligned}$$

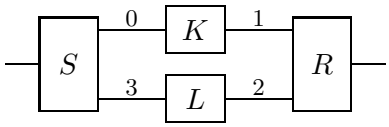


Figure 1: Architecture of the protocol.

$$\begin{aligned} K &= (\textstyle\sum_{f\in F} r_0(f)\cdot K_f)^*\delta \\ K_f &= i\cdot s_1(f) + i\cdot s_1(\bot) + i \end{aligned}$$

$$L = (r_2\cdot(i\cdot s_3 + i))^*\delta$$

$$\begin{aligned} R &= (R_0\cdot R_1)^*\delta \\ R_n &= (r_1(\bot) + \textstyle\sum_{d\in D} r_1(d(1-n))\cdot s_2)^* \sum_{d\in D} r_1(dn)\cdot s(d)\cdot s_2 \end{aligned}$$

The s_k and r_k actions, with $k \leq 3$, are the send and receive actions over the internal port k. We let $\gamma(s_k, r_k) = \gamma(r_k, s_k) = c_k$ for $k \in \{2, 3\}$ and

$$\gamma(s_k(m), r_k(m)) = \gamma(r_k(m), s_k(m)) = c_k(m)$$

for all messages m and $k \leq 1$, and γ undefined otherwise. The action c_2 models the passing of an acknowledgment from R to L. The action c_3 is the passing of an acknowledgement from L to S. The action to is performed only if no other actions are enabled; the partial priority ordering $<$ is defined by $\mathsf{to} < a$ for all actions a other than to.

The set H consists of all the send and receive actions over the internal ports. The set I consists of to, i and all the communications. The complete system is defined as $P = \tau_I(p)$, where

$$p = \theta(\partial_H(S \,||\, K \,||\, R \,||\, L)).$$

Verification. Notation: $[\_]$ for $\theta(\partial_H(\_))$. We use the following abbreviations.

$$\begin{aligned} p_d^0 &= [S_{d0}S_1S \,||\, K_{d0}K \,||\, R \,||\, L] \\ p_d^1 &= [S_{d0}S_1S \,||\, K \,||\, R \,||\, L] \\ p_d^2 &= [S_{d0}S_1S \,||\, K \,||\, R_1R \,||\, (i\cdot s_3 + i)L] \\ p_d^3 &= [S_{d0}S_1S \,||\, K \,||\, R_1R \,||\, L] \\ p_d^4 &= [S_{d0}S_1S \,||\, K_{d0}K \,||\, R_1R \,||\, L] \\ q &= [S_1S \,||\, K \,||\, R_1R \,||\, L] \end{aligned}$$

Driving the composed operator $\theta\circ\partial_H$ inwards we leave out all summands that are blocked by either the encapsulation or the priority operator. In particular this means that, in the presence of alternatives, summands starting with a to action are renamed to δ. The linearization is illustrated in Figure 2.

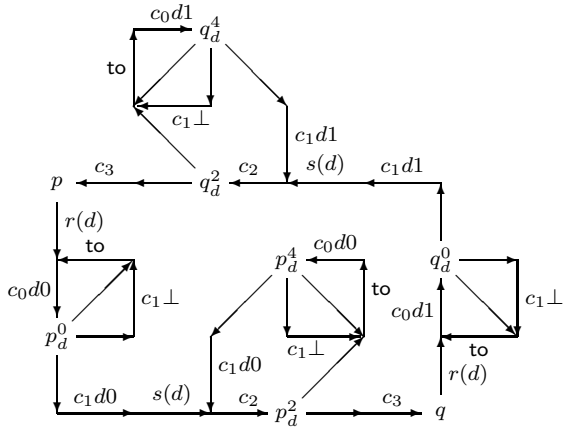


Figure 2: Illustration of the process p. We left out the labels of i-transitions.

$$
\begin{aligned}
p &= \textstyle\sum_{d\in D} r(d)\cdot[s_0(d0)\cdot S_{d0}S_1S \,||\, K \,||\, R \,||\, L] \\
&= \textstyle\sum_{d\in D} r(d)\cdot c_0(d0)\cdot p_d^0
\end{aligned}
$$

$$
\begin{aligned}
p_d^0 &= i\cdot[S_{d0}S_1S \,||\, s_1(d0)\cdot K \,||\, R \,||\, L] \;+ \\
&\quad\; i\cdot[S_{d0}S_1S \,||\, s_1(\bot)\cdot K \,||\, R \,||\, L] \;+\; i\cdot p_d^1 \\
&= i\cdot c_1(d0)\cdot[S_{d0}S_1S \,||\, K \,||\, s(d)\cdot s_2\cdot R_1R \,||\, L] \;+\; (i\cdot c_1(\bot)+i)\cdot p_d^1 \\
&= i\cdot c_1(d0)\cdot s(d)\cdot[S_{d0}S_1S \,||\, K \,||\, s_2\cdot R_1R \,||\, L] \;+\; (i\cdot c_1(\bot)+i)\cdot p_d^1 \\
&= i\cdot c_1(d0)\cdot s(d)\cdot c_2\cdot p_d^2 + (i\cdot c_1(\bot)+i)\cdot p_d^1
\end{aligned}
$$

$$
\begin{aligned}
p_d^1 &= \mathsf{to}\cdot[s_0(d0)\cdot S_{d0}S_1S \,||\, K \,||\, R \,||\, L] \\
&= \mathsf{to}\cdot c_0(d0)\cdot p_d^0
\end{aligned}
$$

$$
\begin{aligned}
p_d^2 &= i\cdot[S_{d0}S_1S \,||\, K \,||\, R_1R \,||\, s_3\cdot L] \;+\; i\cdot p_d^3 \\
&= i\cdot c_3\cdot q \;+\; i\cdot p_d^3
\end{aligned}
$$

$$
\begin{aligned}
p_d^3 &= \mathsf{to}\cdot[s_0(d0)\cdot S_{d0}S_1S \,||\, K \,||\, R_1R \,||\, L] \\
&= \mathsf{to}\cdot c_0(d0)\cdot p_d^4
\end{aligned}
$$

$$
\begin{aligned}
p_d^4 &= i\cdot[S_{d0}S_1S \,||\, s_1(d0)\cdot K \,||\, R_1R \,||\, L] \;+ \\
&\quad\; i\cdot[S_{d0}S_1S \,||\, s_1(\bot)\cdot K \,||\, R_1R \,||\, L] \;+\; i\cdot p_d^3 \\
&= i\cdot c_1(d0)\cdot[S_{d0}S_1S \,||\, K \,||\, s_2\cdot R_1R \,||\, L] \;+\; (i\cdot c_1(\bot)+i)\cdot p_d^3 \\
&= i\cdot c_1(d0)\cdot c_2\cdot p_d^2 \;+\; (i\cdot c_1(\bot)+i)\cdot p_d^3
\end{aligned}
$$

Using RSP* we derive from the equations for p:

$$\begin{aligned} p_d^0 &= ((i \cdot c_1(\bot) + i) \cdot \mathsf{to} \cdot c_0(d0))^* i \cdot c_1(d0) \cdot s(d) \cdot c_2 \cdot p_d^2 \\ p_d^2 &= \left(i \cdot \mathsf{to} \cdot c_0(d0) \cdot \tilde{p}_d^4\right)^* i \cdot c_3 \cdot q \\ \tilde{p}_d^4 &= ((i \cdot c_1(\bot) + i) \cdot \mathsf{to} \cdot c_0(d0))^* i \cdot c_1(d0) \cdot c_2 \end{aligned}$$

In a similar way we can derive

$$\begin{aligned} q &= \textstyle\sum_{d \in D} r(d) \cdot c_0(d1) \cdot q_d^0 \\ q_d^0 &= ((i \cdot c_1(\bot) + i) \cdot \mathsf{to} \cdot c_0(d1))^* i \cdot c_1(d1) \cdot s(d) \cdot c_2 \cdot q_d^2 \\ q_d^2 &= \left(i \cdot \mathsf{to} \cdot c_0(d1) \cdot \tilde{q}_e^4\right)^* i \cdot c_3 \cdot p \\ \tilde{q}_d^4 &= ((i \cdot c_1(\bot) + i) \cdot \mathsf{to} \cdot c_0(d1))^* i \cdot c_1(d1) \cdot c_2 \end{aligned}$$

Using substitution we eliminate all process abbreviations from the equation for p, yielding terms t_p and t_q such that $p = t_p \cdot q$ and $q = t_q \cdot p$. We derive by pressing the operator τ_I inwards and compressing τ's

$$\tau_I(t_p) = \tau_I(t_q) = \textstyle\sum_{d \in D} r(d) \cdot \tau(\tau^* \tau)\tau \cdot s(d) \cdot \tau((\tau\tau(\tau^*\tau)\tau)^*\tau)\tau.$$

Using OFIR2 we derive that

$$\tau_I(t_p) = \textstyle\sum_{d \in D} r(d) \cdot \tau \cdot s(d) \cdot \tau.$$

Since P equals $\tau_I(t_p) \cdot \tau_I(t_p) \cdot P$, we find by RSP* that

$$P = \left(\textstyle\sum_{d \in D} r(d) \cdot \tau \cdot s(d) \cdot \tau\right)^* \delta.$$

Remark 9.1 In [28], a verification of the Concurrent Alternating Bit Protocol (CABP) can be found. The CABP has parallel internal activity; in any state the system can do some internal step. This is reflected in the outcome of the verification. The CABP was proved rooted orthogonally bisimilar to the process

$$(\tau^* \textstyle\sum_{d \in D} r(d) \cdot \tau \cdot (\tau^* s(d)) \cdot \tau)^* \delta.$$

10 Conclusions

In this paper we introduced orthogonal bisimulation equivalence and proved a number of elementary results. In this final section we comment on its position in behavioral semantics.

Orthogonal bisimulation equivalence admits compression, but is not as abstract as the common behavioral equivalences that deal with abstraction. In particular, it is a finer equivalence than branching bisimilarity [19]. Van Glabbeek and Weijland, the founders of branching bisimulation equivalence, remark that "we know of no useful operator for which some abstract equivalence in the linear time-branching time spectrum is a congruence, but rooted branching bisimulation is not." [19, p. 594], and provide many more arguments in favor of branching bisimulation equivalence. Furthermore, branching bisimulation equivalence is argued to be optimal in the following sense: it is the *coarsest* behavioral equivalence that respects the

branching structure of processes[2] (see [16]), and it is the *finest* congruence possible for a common repertoire of process algebra operations (supporting the interleaving hypothesis) that is *abstract* in the sense that it satisfies at least $a\tau x = ax$ (cf. [14, 19]).

To the best of our knowledge, orthogonal bisimilarity is the first behavioral semantics that is a congruence for the priority operator, and that takes the nature of abstraction into account (up to compression). In comparison with branching bisimulation equivalence, a typical property of orthogonal bisimulation equivalence is that it refutes the axiom $x\tau = x$ (or $a\tau x = ax$ in a setting with only action prefix), while it validates the weakened version $x\tau\tau = x\tau$. This property simply represents another perspective on silent activity, acknowledging its presence, but not its structure. An immediate consequence of $x\tau = x$ not being sound is that divergence is preserved more often than in branching bisimilarity. This may play a role in the area of protocol specification and verification, where τ-cycles usually result from the abstraction of the occurrence and recovery of an undesirable event, and fairness is assumed. In Section 9 it is shown that such events in our modeling of the PAR protocol can indeed be discarded after abstraction, as all exits of the occurring τ-cycles happen to start with a τ-step. However, this is not always the case in the daily practice of protocol specification in process algebra (cf. Remark 9.1). For this reason, the divergence sensitive variant of orthogonal bisimulation equivalence is distinguished (which simply never discards τ-cycles while it respects the branching structure of a process in the same sense).

In this paper, we attempted to show that orthogonal bisimulation equivalence is a refinement of branching bisimulation equivalence that is interesting in its own right. Although it is not an 'abstract equivalence' in the sense described above, it certainly provides another look at abstraction in process algebra, and may be of use in situations where compression or priorities play a role.

Future Work. We did not yet analyze the complexity of (divergence sensitive) orthogonal bisimilarity in finite state transition systems. Furthermore, the 'branching structure of a process' as defined in [16] depends on a notion of *observable content* of the traces of that process; it might well be that the "compression content" of a trace, i.e., the traces of the process from which all second and consecutive τ's are removed, leads to a characterization of orthogonal bisimilarity along the same lines as in [16]. Finally, 'orthogonal versions' for other behavioral semantics still have to be formulated, characterized and interrelated. For example, how should orthogonal ready equivalence or failure equivalence be defined, and is it a congruence for process algebra with priorities? We note that it does not make sense to consider orthogonal versions of behavioral equivalences that identify more than ready trace or failure trace equivalence: it is well-known (see, e.g., [5]) that the priority operator is not compatible with failure or ready semantics.[3]

[2] This notion is formally defined in [16].

[3] Suppose $A = \{a, b, c, d, e, f\}$ and $f < b < d$ and $P = a(bc + d) + a(be + f)$, $Q = a(be + d) + a(bc + f)$, then $P \equiv_R Q$ (so $P \equiv_F Q$), but $\theta(P) \not\equiv_F \theta(Q)$.

References

[1] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, IX:127–168, 1986.

[2] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science*, 51(1/2):129–176, 1987.

[3] J.C.M. Baeten and R.J. van Glabbeek. Another look at abstraction in process algebra (extended abstract). In Th. Ottmann, editor, *Proceedings 14th International Colloquium on Automata, Languages and Programming (ICALP'87),* Karlsruhe, volume 267 of *Lecture Notes in Computer Science*, pages 84–94. Springer-Verlag, 1987.

[4] J.C.M. Baeten and C. Verhoef. Concrete process algebra. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 4, Syntactical Methods*, chapter 2, pages 149–268. Oxford University Press, 1995.

[5] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[6] J.A. Bergstra, I. Bethke, and A. Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37(4):243–258, 1994.

[7] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1–3):109–137, 1984.

[8] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, May 1985.

[9] J.A. Bergstra and A. Ponse. Two recursive generalizations of iteration in process algebra. Technical Report P9808, Programming Research Group, University of Amsterdam, 1998. Extended version to appear in *Theoretical Computer Science* under the title *Non-regular iterators in process algebra*.

[10] J.A. Bergstra and A. Ponse. Register-machine based processes. Available at `www.science.uva.nl/~alban/publist/Summerschool.ps.Z` (an extended version is submitted for publication), 1999.

[11] R.N. Bol and J.F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, 1996.

[12] W.J. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. Springer-Verlag, 2000.

[13] W.J. Fokkink and H. Zantema. Basic process algebra with iteration: completeness of its equational axioms. *The Computer Journal*, 37(4):259–267, 1994.

[14] R.J. van Glabbeek. A complete axiomatization for branching bisimulation congruence of finite-state behaviours. In A.M. Borzyszkowski and S. Sokolowski, editors, *Proceedings Mathematical Foundations of Computer Science 1993 (MFCS)*, Lecture Notes in

Computer Science, pages 473–484, Gdansk, Poland, August/September 1993. Springer-Verlag.

[15] R.J. van Glabbeek. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves. Manuscript. Preliminary version available by ftp at `ftp://boole.stanford.edu/pub/spectrum.ps.gz`, 1993. Extended abstract in E. Best, editor: Proceedings *CONCUR'93*, 4^{th} International Conference on Concurrency Theory, Hildesheim, Germany, August 1993, LNCS 715, Springer, pp. 66–81.

[16] R.J. van Glabbeek. What is branching time semantics and why to use it? In M. Nielsen, editor, *The Concurrency Column*, pages 190–198. *Bulletin of the EATCS* 53, 1994. Also available as Report STAN-CS-93-1486, Stanford University, 1993, and by ftp at `ftp://Boole.stanford.edu/pub/branching.ps.gz`.

[17] 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. Elsevier, To appear in 2001. Available at `http://boole.stanford.edu/pub/spectrum1.ps.gz`.

[18] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89,* Proceedings of the IFIP 11th World Computer Congress, San Fransisco 1989, pages 613–618. North-Holland, 1989. Full version in *Journal of the ACM* 43(3):555–600, 1996.

[19] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[20] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[21] R. Milner. Modal characterisation of observable machine behaviour. In G. Astesiano and C. Bohm, editors, *Proceedings CAAP 81*, volume 112 of *Lecture Notes in Computer Science*, pages 25–34. Springer-Verlag, 1981.

[22] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 28(3):267–310, 1983.

[23] R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[24] M.L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Englewood Cliffs N.J., 1967.

[25] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[26] A.S. Tanenbaum. *Computer Networks* (second edition). Prentice-Hall, Englewood Cliffs N.J., 1989.

[27] F.W. Vaandrager. Two simple protocols. In J.C.M. Baeten, editor, *Applications of Process Algebra*, number 17 in Cambridge Tracts in Theoretical Computer Science, pages 23–45. Cambridge University Press, 1990.

[28] M.B. van der Zwaag. Some verifications in process algebra with iota. Report P9806, Programming Research Group, University of Amsterdam, 1998.

[29] M.B. van der Zwaag. A verification in process algebra with iota. In J.F. Groote, S.P. Luttik, and J.J. van Wamel, editors, *Proceedings of the Third International Workshop on Formal Methods for Industrial Critical Systems*, pages 347–368, Amsterdam, The Netherlands, May 1998. CWI.

A Congruence Proofs

We prove that rooted orthogonal bisimilarity is a congruence relation with respect to all operators of $\mathrm{ACP}_{\tau\theta}^{\mathrm{orth}}$. The proof is both straightforward and elaborate. Consider the transition system of closed $\mathrm{ACP}_{\tau\theta}^{\mathrm{orth}}(A,\gamma)$ terms for some arbitrary A, γ. We let t,u,v,w range over closed process terms and x,y,z range over states (the state set consists of the closed terms and the termination state $\surd$).

Two useful properties of orthogonal bisimulations: if R is an orthogonal bisimulation with xRy, then $x=\surd$ if, and only if, $y=\surd$, and, for $a\in A_\tau$, $x\xrightarrow{a}$ if, and only if, $y\xrightarrow{a}$.

Consider (for $i=1,2$) process terms t_i and u_i such that $t_i \leftrightarrows_{ro} u_i$. We prove that $t_1\diamond t_2 \leftrightarrows_{ro} u_1\diamond u_2$ for all $\diamond\in\{\cdot,\|,\between,|\}$, and that $\dagger(t_1)\leftrightarrows_{ro}\dagger(u_1)$ for all $\dagger\in\{\partial_H,\tau_I,\theta\}$ with arbitrary $I,H\subseteq A$ and priority ordering $<$ on A_τ. After this proof, we give a separate proof for the alternative composition.

There is an orthogonal bisimulation R_i, that is rooted between t_i and u_i (for $i=1,2$). Let $R=R_1\cup R_2\cup R'$, where R' is defined as follows.

$$R'=\left\{(t\diamond v,u\diamond w),(\dagger(t),\dagger(u))\;\middle|\;\begin{array}{l}(t,u)\in R_1\setminus\{(\surd,\surd)\},\ (v,w)\in R_2\setminus\{(\surd,\surd)\},\\ \diamond\in\{\cdot,\|,\between,|\},\ \dagger\in\{\partial_H,\tau_I,\theta\}\end{array}\right\}$$

We show that R is an orthogonal bisimulation that satisfies the appropriate root conditions.

Symmetry. The set R is symmetric. Take any $(x,y)\in R$. If, for $i=1,2$, $(x,y)\in R_i$, then also $(y,x)\in R$, since R_i is symmetric. If $(x,y)\in R'$, then we make the following case distinction.

- If $x\equiv t\diamond v$ and $y\equiv u\diamond w$ for some $\diamond$, then, by definition of R', it holds that tR_1u and vR_2w. Since R_1 and R_2 are symmetric, it holds that uR_1t and wR_2v. Hence, by definition of R', it follows that $(y,x)\in R$.
- If $x\equiv\dagger(t)$ for some $\dagger$, then use the definition of R' and the symmetry of R_1.

Concrete action steps. Take any $(x,y)\in R'$ and assume that $x\xrightarrow{a}x'$ for some a and x' with $a\neq\tau$. We have to show that $y\xrightarrow{a}y'$ for some y' with $x'Ry'$. We make a case distinction on the form of x.

- If $x \equiv t \cdot v$, then, by definition of R', $y \equiv u \cdot w$ such that tR_1u and vR_2w. We see that either $t \xrightarrow{a} \surd$ and $x' = v$, or $t \xrightarrow{a} t'$ and $x' = t' \cdot v$. In the first case also $u \xrightarrow{a} \surd$ since tR_1u. Hence $y \xrightarrow{a} w$. Since vR_2w, also vRw, ok. In the second case, we find that $u \xrightarrow{a} u'$ for some u' with $t'R_1u'$. Then $y \xrightarrow{a} u' \cdot w$. We get $t' \cdot vRu' \cdot w$ using the definition of R'.

- If $x \equiv t||v$, then, by definition of R', $y \equiv u||w$ such that tR_1u and vR_2w. We distinguish 8 possibilities:

  1. $t \xrightarrow{a} \surd$ and $x' = v$. In this case also $u \xrightarrow{a} \surd$, and hence $y \xrightarrow{a} w$, ok.
  2. $t \xrightarrow{a} t'$ and $x' = t' \,||\, v$. In this case $u \xrightarrow{a} u'$ with $t'R_1u'$. Then $y \xrightarrow{a} u' \,||\, w$. Using the definition of R', we get $x'Ru' \,||\, w$.
  3. $v \xrightarrow{a} \surd$ and $x' = t$. Like case 1.
  4. $v \xrightarrow{a} v'$ and $x' = t \,||\, v'$. Like case 2.
  5. $t \xrightarrow{b} \surd$ and $v \xrightarrow{c} \surd$ and $\gamma(b,c) = a$ and $x' = \surd$. In this case, it holds that $u \xrightarrow{b} \surd$ and $w \xrightarrow{c} \surd$. Hence $y \xrightarrow{a} \surd$. It holds that $\surd R \surd$, since $\surd R_1 \surd$ (and $\surd R_2 \surd$).
  6. $t \xrightarrow{b} t'$ and $v \xrightarrow{c} \surd$ and $\gamma(b,c) = a$ and $x' = t'$. Straightforward.
  7. $t \xrightarrow{b} \surd$ and $v \xrightarrow{c} v'$ and $\gamma(b,c) = a$ and $x' = v'$. Straightforward.
  8. $t \xrightarrow{b} t'$ and $v \xrightarrow{c} v'$ and $\gamma(b,c) = a$ and $x' = t' \,||\, v'$. Straightforward.

- If $x \equiv t \between v$, then, by definition of R', $y \equiv u \between w$ such that tR_1u and vR_2w. We distinguish 2 possibilities:

  1. $t \xrightarrow{a} \surd$ and $x' = v$. In this case also $u \xrightarrow{a} \surd$, and hence $y \xrightarrow{a} w$, ok.
  2. $t \xrightarrow{a} t'$ and $x' = t' \,||\, v$. In this case $u \xrightarrow{a} u'$ with $t'R_1u'$. Then $y \xrightarrow{a} u' \,||\, w$. Using the definition of R', we get $x'Ru' \,||\, w$.

- If $x \equiv t \,|\, v$, then, by definition of R', $y \equiv u \,|\, w$ such that tR_1u and vR_2w. We distinguish 4 possibilities:

  1. $t \xrightarrow{b} \surd$ and $v \xrightarrow{c} \surd$ and $\gamma(b,c) = a$ and $x' = \surd$. In this case, it holds that $u \xrightarrow{b} \surd$ and $w \xrightarrow{c} \surd$. Hence $y \xrightarrow{a} \surd$. It holds that $\surd R \surd$, since $\surd R_1 \surd$ (and $\surd R_2 \surd$).
  2. $t \xrightarrow{b} t'$ and $v \xrightarrow{c} \surd$ and $\gamma(b,c) = a$ and $x' = t'$. Straightforward.
  3. $t \xrightarrow{b} \surd$ and $v \xrightarrow{c} v'$ and $\gamma(b,c) = a$ and $x' = v'$. Straightforward.
  4. $t \xrightarrow{b} t'$ and $v \xrightarrow{c} v'$ and $\gamma(b,c) = a$ and $x' = t' \,||\, v'$. Straightforward.

- If $x \equiv \partial_H(t)$, then, by definition of R', $y \equiv \partial_H(u)$ such that tR_1u. If $x' = \surd$, then $t \xrightarrow{a} \surd$ and $a \notin H$. Then also $u \xrightarrow{a} \surd$, and $y \xrightarrow{a} \surd$, ok. If $x' \neq \surd$, then $x' = \partial_H(t')$, for some t' with $t \xrightarrow{a} t'$ and $a \notin H$. Then $u \xrightarrow{a} u'$ for some u' with $t'R_1u'$. So $y \xrightarrow{a} \partial_H(u')$. Using the definition of R', we get $\partial_H(t')R\partial_H(u')$.

- If $x \equiv \tau_I(t)$, then we proceed as in the previous case.

- If $x \equiv \theta(t)$, then, by definition of R', $y \equiv \theta(u)$ such that tR_1u. If $x' = \surd$, then $t \xrightarrow{a} \surd$ and there is no $b > a$ with $t \xrightarrow{b}$. Since tR_1u, also $u \xrightarrow{a} \surd$ and there is no $b > a$ with $u \xrightarrow{b}$. Hence $y \xrightarrow{a} \surd$.

  If $x' = \surd$, then $x' \equiv \theta(t')$, for some t' with $t \xrightarrow{a} t'$ and there is no $b > a$ with $t \xrightarrow{b}$. Since tR_1u, it holds that $u \xrightarrow{a} u'$ for some u' with $t'R_1u'$ and there is no $b > a$ with $u \xrightarrow{b}$. Hence $y \xrightarrow{a} \theta(u')$ and $x'R'\theta(u')$.

Silent steps. Take any $(x, y) \in R'$ and assume that $x \xrightarrow{\tau} x'$ for some x'. We have to show that $y \xrightarrow{\tau}$ and that, for some $n \geq 0$, there are y_k such that $y_0 \cdots y_n \in \tau\text{-}paths(y)$ and $x'Ry_n$ and xRy_k for all $i < n$.

We make a case distinction on the form of x.

- If $x \equiv t \cdot v$, then, by definition of R', $y \equiv u \cdot w$ such that tR_1u and vR_2w. We see that either $t \xrightarrow{\tau} \surd$ and $x' = v$, or $t \xrightarrow{\tau} t'$ and $x' = t' \cdot v$. In the first case, since tR_1u, $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n > 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $u_n = \surd$ and tR_1u_i for all $i < n$. Then $(u_0 \cdot w) \cdots (u_{n-1} \cdot w)w \in \tau\text{-}paths(y)$. We have $x'R_2w$, and $xR'u_i \cdot w$ for $i < n$ by definition of R'.

  In the second case, we find that, since tR_1u, $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n \geq 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $t'R_1u_n$ and tR_1u_i for all $i < n$. Then $(u_0 \cdot w) \cdots (u_n \cdot w) \in \tau\text{-}paths(y)$. We have $x'R_2u_n \cdot w$, and $xR'u_i \cdot w$ for $i < n$ by definition of R'.

- If $x \equiv t||v$, then, by definition of R', $y \equiv u||w$ such that tR_1u and vR_2w. We distinguish 4 possibilities:

  1. $t \xrightarrow{\tau} \surd$ and $x' = v$. In this case $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n > 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $u_n = \surd$ and tR_1u_i for all $i < n$. Then also $(u_0 \,||\, w) \cdots (u_{n-1} \,||\, w)w \in \tau\text{-}paths(y)$, and, by definition of R', $xR'u_i \,||\, w$ for all $i < n$.
  2. $t \xrightarrow{\tau} t'$ and $x' = t' \,||\, v$. In this case $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n \geq 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $t'R_1u_n$ and tR_1u_i for all $i < n$. Then also $(u_0 \,||\, w) \cdots (u_n \,||\, w) \in \tau\text{-}paths(y)$, and, by definition of R', $xR'u_i \,||\, w$, for all $i < n$, and $x'R'u_n \,||\, w$.
  3. $v \xrightarrow{\tau} \surd$ and $x' = t$. Like case 1.
  4. $v \xrightarrow{\tau} v'$ and $x' = t \,||\, v'$. Like case 2.

- If $x \equiv t \between v$, then, by definition of R', $y \equiv u \between w$ such that tR_1u and vR_2w. We distinguish 2 possibilities:

  1. $t \xrightarrow{\tau} \surd$ and $x' = v$. In this case $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n > 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $u_n = \surd$ and tR_1u_i for all $i < n$. Then also $(u_0 \between w)(u_1 \,||\, w) \cdots (u_{n-1} \,||\, w)w \in \tau\text{-}paths(y)$, and, by definition of R', $xR'u_i \,||\, w$ for all $0 < i < n$.

2. $t \xrightarrow{\tau} t'$ and $x' = t' \,||\, v$. In this case $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n \geq 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $t'R_1u_n$ and tR_1u_i for all $i < n$. Then also $(u_0 \between w)(u_1 \,||\, w) \cdots (u_n \,||\, w) \in \tau\text{-}paths(y)$, and, by definition of R', $xR'u_i \,||\, w$, for all $0 < i < n$, and $x'R'u_n \,||\, w$.

- If $x \equiv t \,|\, v$, then x cannot have an outgoing τ-step.

- If $x \equiv \partial_H(t)$, then, by definition of R', $y \equiv \partial_H(u)$ such that tR_1u. We see that $t \xrightarrow{\tau}$ and hence $u \xrightarrow{\tau}$ and $y \xrightarrow{\tau}$.

  If $x' = \surd$, then $t \xrightarrow{\tau} \surd$. Furthermore, for some $n > 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $u_n = \surd$ and tR_1u_i for all $i < n$. Then also $\partial_H(u_0) \cdots \partial_H(u_{n-1})\surd \in \tau\text{-}paths(y)$ and $xR'\partial_H(u_i)$ for all $i < n$.

  If $x' \neq \surd$, then $x' = \partial_H(t')$, for some t' with $t \xrightarrow{\tau} t'$. Furthermore, for some $n \geq 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $t'R_1u_n$ and tR_1u_i for all $i < n$. Then also $\partial_H(u_0) \cdots \partial_H(u_n) \in \tau\text{-}paths(y)$ and $x'R'\partial_H(u_n)$ and $xR'\partial_H(u_i)$ for all $i < n$.

- If $x \equiv \tau_I(t)$, then, by definition of R', $y \equiv \tau_I(u)$ such that tR_1u. We distinguish 4 possibilities:

  1. $t \xrightarrow{\tau} \surd$ and $x' = \surd$. We see that $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n > 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $u_n = \surd$ and tR_1u_i for all $i < n$. Then also $\tau_I(u_0) \cdots \tau_I(u_{n-1})\surd \in \tau\text{-}paths(y)$ and $xR'\tau_I(u_i)$ for all $i < n$.
  2. $t \xrightarrow{\tau} t'$ and $x' = \tau_I(t')$. We see that $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Furthermore, for some $n \geq 0$, there are u_i such that $u_0 \cdots u_n \in \tau\text{-}paths(u)$ and $t'R_1u_n$ and tR_1u_i for all $i < n$. Then also $\tau_I(u_0) \cdots \tau_I(u_n) \in \tau\text{-}paths(y)$ and $x'R'\tau_I(u_n)$ and $xR'\tau_I(u_i)$ for all $i < n$.
  3. $t \xrightarrow{a} \surd$ and $a \in I$ and $x' = \surd$. In this case, $u \xrightarrow{a} \surd$ and hence $y \xrightarrow{\tau} \surd$.
  4. $t \xrightarrow{a} t'$ and $a \in I$ and $x' = \tau_I(t')$. In this case, $u \xrightarrow{a} u'$ for some u' with $t'R_1u'$. So $y \xrightarrow{\tau} \tau_I(u')$. By definition of R', we have $x'R'\tau_I(u')$.

- If $x \equiv \theta(t)$, then, by definition of R', $y \equiv \theta(u)$ such that tR_1u.

  If $x' = \surd$, then $t \xrightarrow{\tau} \surd$, and there is no $a > \tau$ with $t \xrightarrow{a}$. Since tR_1u, we have that for some $n > 0$, there are u_i such that $u_0 \ldots u_n \in \tau\text{-}paths(u)$, $u_n = \surd$, and tR_1u_i for $i < n$. Since, for $i < n$, tR_1u_i, it holds that there is no $a > \tau$ with $u_i \xrightarrow{a}$. Hence $\theta(u_0) \ldots \theta(u_{n-1})u_n \in \tau\text{-}paths(y)$. We have that $xR\theta(u_i)$ by definition of R'.

  If $x' \neq \surd$, then $t \xrightarrow{\tau} t'$ for some t' with $x' \equiv \theta(t')$, and there is no $a > \tau$ with $t \xrightarrow{a}$. Since tR_1u, we have that for some $n \geq 0$, there are u_i such that $u_0 \ldots u_n \in \tau\text{-}paths(u)$, $t'R_1u_n$, and tR_1u_i for $i < n$. So $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Since, for $i < n$, tR_1u_i, it holds that there is no $a > \tau$ with $u_i \xrightarrow{a}$. Hence $\theta(u_0) \ldots \theta(u_n) \in \tau\text{-}paths(y)$. We have that $xR\theta(u_i)$ and $x'R\theta(u_n)$ by definition of R'.

Rootedness. We have to show that R is rooted between $t_1 \diamond t_2$ and $u_1 \diamond u_2$, and between $\dagger(t_1)$ and $\dagger(u_1)$, for all $\diamond \in \{\cdot, ||, \between, |\}$ and $\dagger \in \{\partial_H, \tau_I, \theta\}$.

- Sequential composition. Assume that $t_1 \cdot t_2 \xrightarrow{\tau} x$. We distinguish two possibilities:

1. $t_1 \xrightarrow{\tau} \surd$ and $x = t_2$. Since R_1 is rooted between t_1 and u_1, it holds that $u_1 \xrightarrow{\tau} \surd$ and hence $u_1 \cdot u_2 \xrightarrow{\tau} u_2$, ok.
2. $t_1 \xrightarrow{\tau} t_1'$ and $x = t_1' \cdot t_2$. Since R_1 is rooted between t_1 and u_1, there must be some u_1' with $u_1 \xrightarrow{\tau} u_1'$ and $t_1' R_1 u_1'$. We see that $u_1 \cdot u_2 \xrightarrow{\tau} u_1' \cdot u_2$ and $x R u_1' \cdot u_2$, ok.

Silent steps of $u_1 \cdot u_2$ are treated symmetrically.

- Merge. Assume that $t_1 \,||\, t_2 \xrightarrow{\tau} x$. We distinguish 4 possibilities:

1. $t_1 \xrightarrow{\tau} \surd$ and $x = t_2$. Since R_1 is rooted between t_1 and u_1, it holds that $u_1 \xrightarrow{\tau} \surd$ and hence $u_1 \,||\, u_2 \xrightarrow{\tau} u_2$, ok.
2. $t_1 \xrightarrow{\tau} t_1'$ and $x = t_1' \,||\, t_2$. Since R_1 is rooted between t_1 and u_1, there must be some u_1' with $u_1 \xrightarrow{\tau} u_1'$ and $t_1' R_1 u_1'$. We see that $u_1 \,||\, u_2 \xrightarrow{\tau} u_1' \,||\, u_2$ and $x R u_1' \,||\, u_2$, ok.
3. $t_2 \xrightarrow{a} \surd$ and $x = t_1$. Like case 1.
4. $t_2 \xrightarrow{a} t_2'$ and $x = t_1 \,||\, t_2'$. Like case 2.

Silent steps of $u_1 \,||\, u_2$ are treated symmetrically.

- Left merge. Assume that $t_1 \mathbin{\underline{||}} t_2 \xrightarrow{\tau} x$. We distinguish 2 possibilities:

1. $t_1 \xrightarrow{\tau} \surd$ and $x = t_2$. Since R_1 is rooted between t_1 and u_1, it holds that $u_1 \xrightarrow{\tau} \surd$ and hence $u_1 \mathbin{\underline{||}} u_2 \xrightarrow{\tau} u_2$, ok.
2. $t_1 \xrightarrow{\tau} t_1'$ and $x = t_1' \,||\, t_2$. Since R_1 is rooted between t_1 and u_1, there must be some u_1' with $u_1 \xrightarrow{\tau} u_1'$ and $t_1' R_1 u_1'$. We see that $u_1 \mathbin{\underline{||}} u_2 \xrightarrow{\tau} u_1' \,||\, u_2$ and $x R u_1' \,||\, u_2$, ok.

Silent steps of $u_1 \mathbin{\underline{||}} u_2$ are treated symmetrically.

- Communication merge. No τ-steps.

- Encapsulation. Assume that $\partial_H(t_1) \xrightarrow{\tau} x$. We distinguish 2 possibilities:

1. $t_1 \xrightarrow{\tau} \surd$ and $x = \surd$. Since R_1 is rooted between t_1 and u_1, it holds that $u_1 \xrightarrow{\tau} \surd$ and hence $\partial_H(u_1) \xrightarrow{\tau} \surd$, ok.
2. $t_1 \xrightarrow{\tau} t_1'$ and $x = \partial_H(t_1')$. Since R_1 is rooted between t_1 and u_1, there must be some u_1' with $u_1 \xrightarrow{\tau} u_1'$ and $t_1' R_1 u_1'$. We see that $\partial_H(u_1) \xrightarrow{\tau} \partial_H(u_1')$ and $x R \partial_H(u_1')$, ok.

Silent steps of $\partial_H(u_1)$ are treated symmetrically.

- Hiding. Assume that $\tau_I(t_1) \xrightarrow{\tau} x$. We distinguish 4 possibilities:

1. $t_1 \xrightarrow{\tau} \surd$ and $x = \surd$. Since R_1 is rooted between t_1 and u_1, it holds that $u_1 \xrightarrow{\tau} \surd$ and hence $\tau_I(u_1) \xrightarrow{\tau} \surd$, ok.
2. $t_1 \xrightarrow{\tau} t_1'$ and $x = \tau_I(t_1')$. Since R_1 is rooted between t_1 and u_1, there must be some u_1' with $u_1 \xrightarrow{\tau} u_1'$ and $t_1' R_1 u_1'$. We see that $\tau_I(u_1) \xrightarrow{\tau} \tau_I(u_1')$ and $x R \tau_I(u_1')$, ok.
3. $t_1 \xrightarrow{a} \surd$ and $a \in I$ and $x = \surd$. Since $t_1 R_1 u_1$, it holds that $u_1 \xrightarrow{a} \surd$ and hence $\tau_I(u_1) \xrightarrow{a} \surd$, ok.

4. $t_1 \xrightarrow{a} t_1'$ and $a \in I$ and $x = \tau_I(t_1')$. Since $t_1 R_1 u_1$, there must be some u_1' with $u_1 \xrightarrow{a} u_1'$ and $t_1' R_1 u_1'$. We see that $\tau_I(u_1) \xrightarrow{\tau} \tau_I(u_1')$ and $x R \tau_I(u_1')$, ok.

Silent steps of $\tau_I(u_1)$ are treated symmetrically.

- Priority. Assume that $\theta(t_1) \xrightarrow{\tau} x$. If $x = \surd$, then it must be that $t_1 \xrightarrow{\tau} \surd$ and there is no $a > \tau$ with $t_1 \xrightarrow{a}$. Since $t_1 R_1 u_1$, it holds that there is no $a > \tau$ with $u_1 \xrightarrow{a}$. Since R_1 is rooted between t_1 and u_1, it holds that $u_1 \xrightarrow{\tau} \surd$.

  If $x \neq \surd$, then it must be that $t_1 \xrightarrow{\tau} t_1'$ for some t_1' with $x = \theta(t_1')$, and there is no $a > \tau$ with $t_1 \xrightarrow{a}$. Since $t_1 R_1 u_1$, it holds that there is no $a > \tau$ with $u_1 \xrightarrow{a}$. Since R_1 is rooted between t_1 and u_1, it holds that $u_1 \xrightarrow{\tau} u_1'$ for some u_1' with $t_1' R_1 u_1'$. By definition of R', we find that $x R \theta(u_1')$.

  Silent steps of $\theta(u_1)$ are treated symmetrically.

Alternative composition. We prove that $\leftrightarrows_{ro}$ is a congruence with respect to the alternative composition. We give a separate proof for this operator, because, contrary to the other operators, we have to use the root condition explicitly.

We show that the relation

$$R = R_1 \cup R_2 \cup \{(t_1 + t_2, u_1 + u_2), (u_1 + u_2, t_1 + t_2)\}$$

is an orthogonal bisimulation that is rooted between $t_1 + t_2$ and $u_1 + u_2$. Clearly, R is symmetric.

- If $t_1 + t_2 \xrightarrow{a} t$ for some a and t with $a \neq \tau$, it must be that $t_i \xrightarrow{a} t$ for some $i \in \{1, 2\}$. Since $t_i R_i u_i$ it holds that $u_i \xrightarrow{a} u$ for some u with $t R_i u$. Then also $u_1 + u_2 \xrightarrow{a} u$ and $t R u$.

- If $t_1 + t_2 \xrightarrow{\tau} t$ for some t, then it must be that $t_i \xrightarrow{\tau} t$ for some $i \in \{1, 2\}$. Since R_i is rooted between t_i and u_i, it holds that $u_i \xrightarrow{\tau} u$ for some u with $t R_i u$. Then also $u_1 + u_2 \xrightarrow{\tau} u$ and $t R u$.

Steps of $u_1 + u_2$ are treated symmetrically. The second case shows that R is rooted between $t_1 + t_2$ and $u_1 + u_2$.

B Expressing Recursive Pure Transition Systems

In this appendix, we spell out an instance of Theorem 8.2 (see Section 8). We consider the case of recursive pure transition systems over label set $L = \{a, b\}$, and with fan-out of at most three. We choose the following menu enumeration:

0	for successful termination,	6	for b, b,
1	for deadlock,	7	for a, a, a,
2	for a,	8	for a, a, b,
3	for b,	9	for a, b, b,
4	for a, a,	10	for b, b, b,
5	for a, b,		

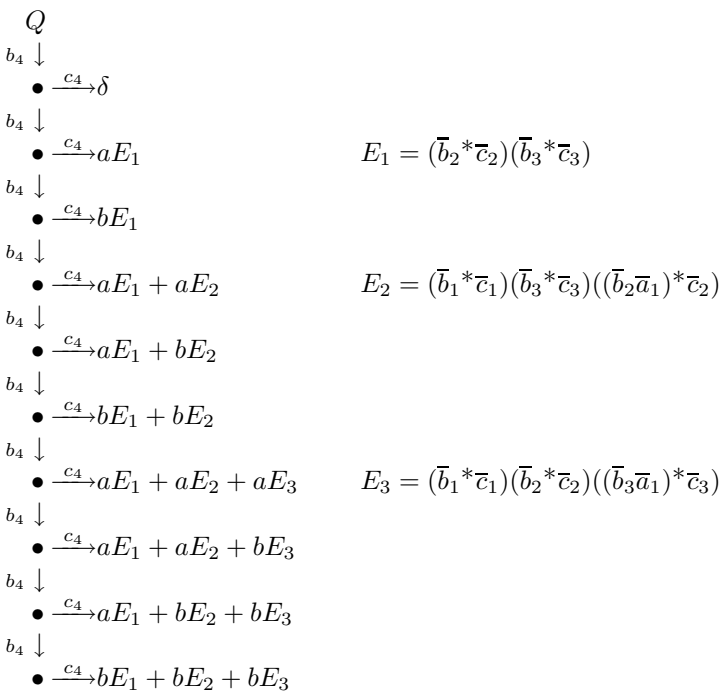


Figure 3: The process Q.

A transition relation is characterized by 4-tuples fetching all outgoing transitions (at most 3): a state s yields the map

$$(s, 0, 0, 0) \mapsto (s_1, ..., s_4)$$

where

- s_4 gives the menu number of s,
- s_i for $i = 1, 2, 3$ is the target associated with source s and the i-th label of menu s_4 if such a transition is present, and otherwise 0.

Let P be such that it models the computation of $\mapsto$, i.e.,

$$\mathsf{Con}[Px](s, 0, 0, 0) \xrightarrow{i} \!\!\!\!\rightarrow \mathsf{Con}[x](s_1, ..., s_4).$$

We define Q partly in a graphical manner in Figure 3. We write $\mathsf{Con}_\tau[x](\vec{s})$ for $\tau_{\{i\}}(\mathsf{Con}[x](\vec{s}))$, and Prog for $(QP)^*\overline{d}_4R$.

As a first example, consider the transition system $5 \xrightarrow{a} \surd$. It is easily seen that

$$\mathsf{Con}_\tau[aE_1P\mathsf{Prog}](0, 0, 0, 0)$$

is rooted divergence sensitive bisimilar with state 5 in the sequential τ-saturation of $5 \xrightarrow{a} \surd$.

Next, consider the transition systems in Figure 4. It is immediately clear that the sequential τ-saturation of the leftmost transition system, here drawn in the middle, has the property that $5 \leftrightarrows_{rdso} \mathsf{Con}_\tau[([aE_1 + aE_2)P\mathsf{Prog}](5, 7, 0, 0)$, as the only difference is in the length of τ-sequences (here drawn by double arrows). Note that we could have taken $(7, 5, 0, 0)$ as initial tuple because there is no ordering in menu 4.

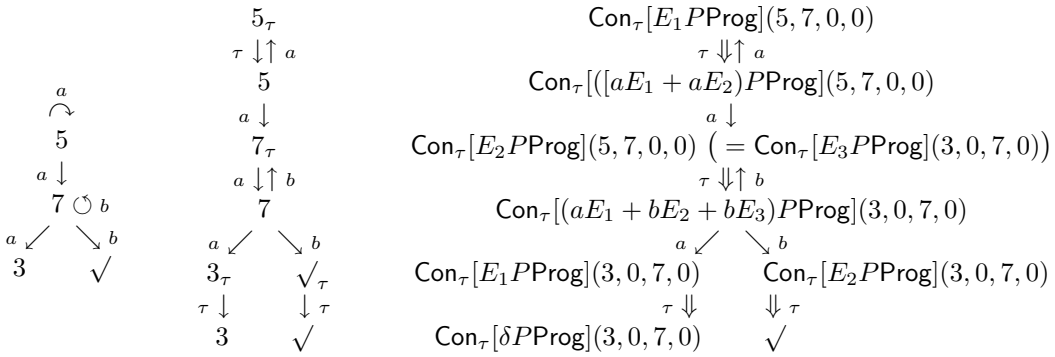


Figure 4: Three transition systems.