Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT
Theoretical Computer Science 309 (2003) 313–355

Theoretical Computer Science

www.elsevier.com/locate/tcs

Branching time and orthogonal bisimulation equivalence

Jan A. Bergstra[a,b], Alban Ponse[a,1], Mark B. van der Zwaag[c,*,1,2]

[a] *Programming Research Group, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, Netherlands*

[b] *Department of Philosophy, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, Netherlands*

[c] *Department of Computer Science, University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, Netherlands*

Received 19 July 2001; received in revised form 21 February 2003; accepted 25 March 2003
Communicated by M. Wirsing

Abstract

We propose a refinement of branching bisimulation equivalence that we call orthogonal bisimulation equivalence. Typically, internal activity (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 operators. 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.

Keywords: Orthogonal bisimulation; Branching time; Process algebra; Silent step; Labelled transition system

1. Introduction

In concurrency theory, Milner's observation equivalence as discussed in the setting of Calculus of Communicating Systems (CCS [24], cf. [26,27]) is a standard example of

* Corresponding author.
E-mail address: mbz@cs.kun.nl (M.B. van der Zwaag).

[1] This paper was written during a period that Ponse and van der Zwaag worked at CWI, Amsterdam. Both authors are thankful for the pleasant working conditions in that period.

[2] Supported by the Netherlands Organisation for Scientific Research (NWO).

0304-3975/$ - see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/S0304-3975(03)00277-9

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 for hiding actions that are assumed not to be observable or interesting for some other reason. In the process algebraic approaches based on Algebra of Communicating Processes (ACP [8], for an overview see [5,15]), observation equivalence is called τ-*bisimulation equivalence* [9], 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. As such, abstraction plays a central role in process algebraic verifications. Furthermore, adding abstraction and finite guarded recursion to ACP yields universal expressive power: each recursive process graph can be expressed up to τ-bisimilarity (see [2]).

A popular and relatively new semantics that deals with abstraction, proposed by van Glabbeek and Weijland [21], is *branching bisimulation equivalence* (see also [22]). Branching bisimulation equivalence is a refinement of semantics such as observation equivalence, delay bisimulation equivalence [25] 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 [22]: "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 [24]) is claimed to be at the very heart of abstraction (see [22]). This axiom expresses that the observational contents of the silent step τ in 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 (see [19,22]; we return to this point in Section 11).

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

- Internal activity, that is, the performance of τ-steps, may be compressed, but not completely discarded.
- Operators that act on 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 establishes a dichotomy between *concrete processes* [4,20], that is, 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, questions about the relation between concrete processes and those that contain τ-steps, for instance about verification or expressiveness issues, should be reconsidered. 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, more identifying) semantics mentioned above.

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 process algebra: first eliminate all occurrences of the priority operator, and then apply abstraction to arrive at a concise characterization of the external behavior. That the priority operator is not fully compatible with any known semantics that deals with abstraction,[3] 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, so that $x\tau y$ 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 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).

We now consider the case of *divergence*, that is, 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 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 to 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 be repeated consecutively only 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,

[3] In the literature, several solutions for this problem have been proposed, but none of these are totally satisfactory and generally accepted; we return to this issue in our conclusions in Section 11.

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 [22]).

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 for the description of infinite processes, and we briefly discuss fairness in the present setting. Section 8 is on expressiveness modulo orthogonal bisimilarity. Section 9 contains an example on expressiveness. Finally, in Section 10 we describe as an example the specification and verification of a PAR protocol [32] in orthogonal bisimulation equivalence. The paper ends with some remarks and conclusions in Section 11.

Note. In earlier work [34,35], orthogonal bisimilarity was defined using a constant $\imath$ instead of τ. We now consider this use of the symbol $\imath$ obsolete.

2. Definition of the equivalence

We introduce transition systems and some auxiliary notions, and after that orthogonal bisimulation equivalence. We designate its place in the lattice of process equivalences by relating it to strong bisimulation equivalence and branching bisimulation equivalence. Finally, we define a variant that is sensitive with respect to diverging silent (τ) behavior.

We start with the standard definition of a (labelled) transition system over a set L of labels as a triple (S,L,T), where S is a nonempty set of states and $T\subseteq S\times L\times S$ is a transition relation. A transition (s,a,s') is usually written as $s\xrightarrow{a}s'$; state s in this transition is referred to as its source and state s' as its target, or as an (a-)successor of s. We write $s\xrightarrow{a}$ if s has an outgoing a-transition.

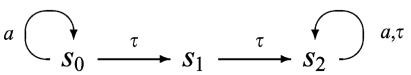
A transition system with *termination* is a transition system together with a predicate $\surd$ on its states; a state s with $\surd s$ is called a termination state. A transition system with termination has *pure termination*, or shortly, is *pure*, if it has a single termination state that has no outgoing transitions. In this case we write $\surd$ to denote the single termination state.

The special label τ represents a silent action: the execution of τ is not observable. The silent action is used for the modelling of internal communications. For a transition system with τ in its set of labels, and for a state s, we define the set of finite τ-paths starting in s as the set τ-*paths*(s) that consists of all sequences $s_0\ldots s_n$ of states with $s_0=s$, $n\geqslant 0$, and $s_i\xrightarrow{\tau}s_{i+1}$ for all $i<n$. For a label set L, that may or may not contain τ, we write L_τ for the set $L\cup\{\tau\}$.

We are now ready to define orthogonal bisimulation equivalence of states.

Definition 1. Consider a transition system (S, L_τ, T) with termination. A binary relation R on S is an *orthogonal bisimulation*, if it is symmetric, and whenever sRr, then
(1) if $\surd s$, then $\surd r$;
(2) if $s \xrightarrow{a} s'$ for some s' and $a \neq \tau$, 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 there is a path $r_0 \ldots r_n \in \tau\text{-}paths(r)$ with $n \geqslant 0$ 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.

For example, the states in the transition system below are orthogonally bisimilar if, and only if, $a = \tau$.



An important observation is that when two states are orthogonally bisimilar and in one a certain action is enabled, then the other can perform this action as well, and this is true for all actions including τ.

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 transition and termination relations. Finally, if the two systems have pure termination, then we identify their termination states.

Below we prove that orthogonal bisimilarity is indeed an equivalence relation. For this proof we use the following lemma, that says that if two states are orthogonally bisimilar, and one has a τ-path of length n, then this path is matched by a τ-path in the other state that consists of n consecutive τ-paths, where each of its intermediate states can be related to an appropriate state in the original path:

Lemma 2. *If R is an orthogonal bisimulation with sRr, and there is a path $s_0 \ldots s_n$ in τ-paths(s), for some $n \geqslant 0$, then there is, for every $i \leqslant n$, an $m_i \geqslant 0$, such that r has a τ-path with $r_0^0 = r$ and $m_n = 0$ and*
(1) *for all $i < n$, $r_i^0 \ldots r_i^{m_i} \in \tau\text{-}paths(r_i^0)$ and $r_i^{m_i} = r_{i+1}^0$,*
(2) *for all $i \leqslant n$, if $j < m_i$ or $j = 0$, then $r_i^j Rs_i$.*

Proof. Straightforward by induction on n.

Theorem 3. *Orthogonal bisimilarity is an equivalence relation.*

Proof. Consider a transition system with termination. Orthogonal bisimilarity is easily shown to be reflexive and symmetric. We show that it is transitive: assuming that $s_1 R' s_2$ and $s_2 R'' s_3$ for orthogonal bisimulations R' and R'', we show that the symmetric relation

$$R = \{(s,r),(r,s) \mid \text{exists } t \text{ such that } sR'tR''r\}$$

is an orthogonal bisimulation, and thereby that $s_1 \leftrightarrows_o s_3$. Take any pair (s,r) from R. By definition of R, there is a state t such that either $sR't$ and $tR''r$, or $rR't$ and $tR''s$. Assume the former; the latter case is symmetric.

First, observe that if s is a termination state then also t and thus r are termination states. Next, if s can do an a-step with $a \neq \tau$ then it is easy to verify that r matches this transition appropriately. So, assume that $s \xrightarrow{\tau} s'$. It is straightforward to verify that $r \xrightarrow{\tau}$. Since $sR't$, the state t matches the τ-step to s' in zero or more transitions: for some $n \geqslant 0$, there is a sequence $t_0 \ldots t_n$ in $\tau\text{-}paths(t)$ such that $sR't_i$ for all $i < n$ and $s'R't_n$. The proof is finished using Lemma 2. □

2.1. Strong bisimulation

We compare orthogonal bisimulation equivalence with strong bisimulation equivalence [30] that is defined as follows. Consider a transition system (S,L,T) with termination. A binary relation R on S is a *strong bisimulation*, if it is symmetric, and whenever sRr, then

(1) if $\surd s$, then $\surd r$;

(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. 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 4. *If s and r are compact, then $s \leftrightarrows_o r$ implies $s \leftrightarrows r$.*

Proof. We show that the relation

$$R = \{(s,t) \mid s \leftrightarrows_o r \text{ and } s,r \text{ compact}\}$$

is a strong bisimulation. Clearly, it is symmetric. Take states s and r with sRr. By definition of R there exists an orthogonal bisimulation R' that relates s and r. We distinguish the following cases. First, if $\surd s$ then it must be that $\surd r$, because R' is an orthogonal bisimulation. Second, if s can do an a-step for some action $a \neq \tau$, then this step is matched directly by an a-step in r, also because R' is an orthogonal bisimulation. Finally, if s has a τ-step to s', then we know that there is a path $r_0 \ldots r_n$ in $\tau\text{-}paths(r)$ such that $sR'r_i$ for $i < n$ and $s'R'r_n$. It suffices to show that it must be that $n = 1$. If $n = 0$, then $s' \leftrightarrows_o r_0$. Since $s \leftrightarrows_o r_0$ and orthogonal bisimilarity is an equivalence relation, we find that s and its successor s' are orthogonally bisimilar, which contradicts the assumption that s is compact. If $n > 1$, then r and its successor r_1 are orthogonally bisimilar, which contradicts the assumption that r is compact. This finishes the proof. □

2.2. Branching bisimulation

We now turn to branching bisimilarity [22]. This equivalence is the finest (smallest, least identifying) of the process equivalences described in [17]. Orthogonal bisimilarity is finer than branching bisimilarity, and hence finer than the equivalences in [17].

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

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

States s and r are branching bisimilar, notation $s \leftrightarrows_{\mathrm{b}} r$, if they are related by some branching bisimulation.

It is straightforward to prove that any orthogonal bisimulation is a branching bisimulation.

2.3. Rootedness

Orthogonal bisimilarity is not a congruence with respect to the operation for alternative composition in process algebra, as can be seen from the following basic example (see Section 4 for the semantics of process terms): the terms τ and $\tau\tau$ are orthogonally bisimilar, while the terms $a + \tau$ and $a + \tau\tau$ with $a \neq \tau$ are not. As for branching bisimilarity, this problem can be overcome by imposing the *root condition* defined below. It turns out that *rooted* orthogonal bisimilarity is a congruence with respect to the process algebraic operators (we come back to this point in Sections 4 and 6).

An orthogonal (branching) bisimulation R is *rooted* between states s and r, if sRr and, for all $a \in L_\tau$,

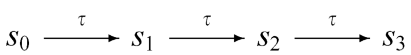
(1) if $s \xrightarrow{a} s'$ for some s', then $r \xrightarrow{a} r'$ for some r' with $s'Rr'$;
(2) if $r \xrightarrow{a} r'$ for some r', then $s \xrightarrow{a} s'$ for some s' with $s'Rr'$.

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

Using Theorem 3 it is straightforward to verify that rooted orthogonal bisimilarity is an equivalence relation.

Proposition 5. $\leftrightarrows \subseteq \leftrightarrows_{\mathrm{o}} \subseteq \leftrightarrows_{\mathrm{b}}$ *and* $\leftrightarrows \subseteq \leftrightarrows_{\mathrm{ro}} \subseteq \leftrightarrows_{\mathrm{rb}}$, *for any transition system with termination.*

For example, the states s_0 and s_1 in the transition system below are rooted orthogonally bisimilar to each other but not to s_2, while s_1 and s_2 are rooted branching bisimilar.

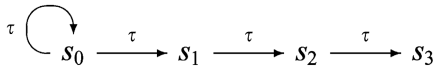


The following lemma is an easy corollary of Lemma 4:

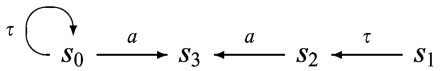
Lemma 6. *If all successors of s,r are compact, then $s \leftrightarrows_{\mathrm{ro}} r$ implies $s \leftrightarrows r$.*

2.4. Divergence

A state s has τ-divergence if there is an infinite τ-path starting in s, that is, if there are states s_i with $s = s_0$ and $s_i \xrightarrow{\tau} s_{i+1}$ for all $i \in \mathbb{N}$. Orthogonal bisimilarity does not always distinguish between states that have τ-divergence and states that have not. For example, the states s_0 and s_1 in the transition system below 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



where $s_0 \not\leftrightarrows_{\mathrm{o}} s_1$ and $s_0 \not\leftrightarrows_{\mathrm{o}} s_2$. 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 noncollapsing version for which τ-divergence is an invariant: 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 $s \leftrightarrows_{\mathrm{dso}} r$, if they are related by a divergence sensitive orthogonal bisimulation. States s and r are *rooted* divergence sensitive orthogonally bisimilar, notation $s \leftrightarrows_{\mathrm{rdso}} r$, 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 is true for the rooted versions.

3. Modal characterization

We present a modal logic that characterizes orthogonal bisimulation equivalence: states in finitely branching transition systems are orthogonally bisimilar exactly if they satisfy the same formulas. This logic can be seen as a variation of Hennessey–Milner Logic [23] which characterizes strong bisimulation equivalence for finitely branching processes. See [31,13] for further details concerning modal logics and their relation to concrete processes; see [29] for a modal characterization of branching bisimulation equivalence.

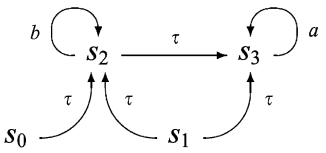


Fig. 1. A transition system.

The primitives of the logic are as follows: transition labels act as existential modal operators, and it has negation, conjunction, and an until operator. Furthermore, there is a termination predicate and a τ-enabledness predicate.

Given a fixed set L of labels not containing τ, the set $\mathscr{L}$ of formulas ϕ is defined by the following grammar:

$$\phi ::= \surd \mid \tau \mid a\phi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \,\mathrm{U}\, \phi,$$

where a ranges over L. We abbreviate the formula $\tau \wedge \neg\tau$ as $\bot$. Furthermore, we write $\top$ for $\neg\bot$, a for $a\top$, and $\mathrm{F}\phi$ for $\top \,\mathrm{U}\, \phi$.

Consider a transition system over L_τ with termination. Truth of a formula in a state s is defined inductively by

- $s \models \surd$, if $\surd s$,
- $s \models \tau$, if $s \xrightarrow{\tau}$,
- $s \models a\phi$, 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 $s \models \psi$, and
- $s \models \phi \,\mathrm{U}\, \psi$, if, for some $n \geqslant 0$, there is a $s_0 \ldots s_n \in \tau\text{-}paths(s)$ such that $s_i \models \phi$ for all $i < n$ and $s_n \models \psi$.

States s and r are $\mathscr{L}$-equivalent, notation $s \sim r$, if, for all formulas ϕ in $\mathscr{L}$, $s \models \phi$ if and only if $r \models \phi$.

Consider for example the transition system in Fig. 1. Every state in this picture satisfies the formula $(\mathrm{F}b)\,\mathrm{U}\,a$. Also, observe that states s_0 and s_1 can reach the same states by τ-steps, namely s_2 and s_3. But while s_1 satisfies $(\neg b)\,\mathrm{U}\,a$, this is not true for s_0. Observe that it is not possible to find a distinguishing formula for s_0 and s_1 using the until operator U only in its restricted form as the future operator F.

Theorem 7. *Consider a transition system over L_τ with termination. For all states s and r, $s \leftrightarrows_{\mathrm{o}} r$ implies $s \sim r$.*

Proof. By induction on the structure of formulas (using Lemma 2). □

In the other direction, the characterization is less general: we have to restrict to transition systems that are finitely branching and τ-path-image-finite. 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 $s \ldots s' \in \tau\text{-}paths(s)$.

We use the following lemma that is easy to prove:

Lemma 8. *If R is an orthogonal bisimulation with sRr and $s \xrightarrow{\tau} s'$, then there is a path $r_0 \dots r_n \in \tau\text{-}paths(r)$ with $n \geqslant 0$ such that sRr_i for all $i < n$ and $s'Rr_n$, and $r_i \neq r_j$ for all distinct $i, j \leqslant n$.*

Theorem 9. *Consider a transition system over L_τ with termination that is τ-path-image-finite and finitely branching in every label. For all states s and r, $s \sim r$ implies $s \leftrightarrows_o r$.*

Proof. We show that $\sim$ is an orthogonal bisimulation. Take any s, r with $s \sim r$. We find directly that $\surd s$ if and only if $\surd r$. There are two cases.

First, consider the case where state s can do a concrete action step: let $s \xrightarrow{a} s'$ with $a \neq \tau$. Since $s \models a\top$, also $r \models a\top$. So, using that r is finitely branching in a, for some $n \geqslant 0$, r has a-successors $r_0, \dots, r_n$. We have to show that, for some $i \leqslant n$, $s' \sim r_i$. Suppose that, for all $i \leqslant n$, $s' \not\sim r_i$. Then there is, for every $i \leqslant n$, a formula ϕ_i, such that $s' \models \phi_i$ and $r_i \not\models \phi_i$. Let $\phi = a(\phi_0 \wedge \dots \wedge \phi_n)$. 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 $s' \sim r'$, which was to be demonstrated.

Second, we consider the case where state s can do a silent step: let $s \xrightarrow{\tau} s'$ for some state s'. If $s' \sim s$ then $s' \sim r$ since $\sim$ is transitive, and $r \xrightarrow{\tau}$, since $s \models \tau$ and hence $r \models \tau$. So suppose that $s' \not\sim s$. We must show that r can match this τ-step to s' appropriately. Suppose, to the contrary, that it cannot (†), that is, that there is no $r_0 \dots r_n \in \tau\text{-}paths(r)$ with $n > 0$ and $s \sim r_i$, for all $i < n$, and $s' \sim r_n$ and, for all $i, j \leqslant n$, if $i \neq j$ then $r_i \neq r_j$. This last condition is justified by Lemma 8.

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

$$n \geqslant 0, \quad \forall i \leqslant n(s \sim r_i), \quad \text{and} \quad \forall i, j \leqslant n(r_i \neq r_j \vee i = j).$$

The set C is finite because r is τ-path-image finite. It is nonempty because $r \in C$. By assumption (†), we see that, for all $r \dots r' \in C$, 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:

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

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 $C' = \{\rho_0, \dots, \rho_k\}$ for some $k \geqslant 0$. For all $\rho_i = r \dots r_i \in C'$ we have 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 $\phi = \phi_0 \wedge \dots \wedge \phi_k$ and $\psi = \psi_0 \wedge \dots \wedge \psi_k$. Then $s \models \phi$, $s' \models \psi$ and for all $i \leqslant k$, $r_i \not\models \phi$ 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 $r \models \phi \,\mathrm{U}\, (\psi \wedge \chi)$, that is, that there is a τ-path $r_0 \ldots r_n$ with $r = r_0$ and $n \geqslant 0$, such that $r_n \models \psi \wedge \chi$ and $r_i \models \phi$ for all $i < n$ (‡). We make the following observations:

- $n > 0$, because $r \not\models \chi$.
- $r_i \sim s$ for all $i < n$. Suppose not, then assume that j is the smallest $j < n$ with $r_j \not\sim s$. Then $r_0 \ldots r_j \in C'$ and so $r_j \not\models \phi$. Contradiction (‡).
- $r_n \not\sim s$, since $s \not\models \chi$.

From these observations, it follows that $r_0 \ldots r_n \in C'$. Hence $r_n \not\models \psi$, which yields the required contradiction. □

We end this section with the remark that a modal logic characterizing divergence sensitive orthogonal bisimulation equivalence is obtained easily by extending the logic with a predicate that is satisfied by a state if and only if it has τ-divergence. The proofs for the corresponding counterparts of Theorems 7 and 9 are trivial extensions of the proofs of these.

4. Process algebra

We use process algebra because it provides an elegant notation for transition systems, and allows for axiomatic reasoning. We begin by presenting the axiom system without abstraction. The axiom system ACP(A, γ) [8] 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 actions.

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 renaming operator ∂_H (encapsulation) for every set $H \subseteq A$. 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 $x \cdot y$.

Subsystems of ACP(A, γ) are BPA(A) (basic process algebra), which consists of axioms A1–A5, and has sequential and alternative composition as operators, and BPA$_\delta(A)$, the extension of BPA(A) with the deadlock process, axiomatized by axioms A6 and A7. If E is any of these axiom systems, then we write CT(E) for its set of closed terms.

We give an operational semantics for the presented axiom systems; we define transition systems with pure termination where closed terms are states: let E be one of the presented axioms systems, parametrized with action set A, then TS(E) is the transition system

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

where $\surd$ is a fresh symbol and the transition relation T is generated by the transition rules in Table 3. The transition rules are such that the termination state $\surd$ has no

Table 1
The axioms of ACP(A,γ); $a,b\in A_\delta$ 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)	$\delta x=\delta$	
(CM1)	$x\|\|y=(x\mathbin{\lfloor\!\lfloor} y+y\mathbin{\lfloor\!\lfloor} x)+x\mid y$	
(CM2)	$a\mathbin{\lfloor\!\lfloor} x=ax$	
(CM3)	$ax\mathbin{\lfloor\!\lfloor} y=a(x\|\|y)$	
(CM4)	$(x+y)\mathbin{\lfloor\!\lfloor} z=x\mathbin{\lfloor\!\lfloor} z+y\mathbin{\lfloor\!\lfloor} z$	
(CM5)	$ax\mid b=(a\mid b)x$	
(CM6)	$a\mid bx=(a\mid b)x$	
(CM7)	$ax\mid by=(a\mid b)(x\|\|y)$	
(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 $\gamma(a,b)$ defined
(CF2)	$a\mid b=\delta$	otherwise
(D1)	$\partial_H(a)=a$	if $a\notin H$
(D2)	$\partial_H(a)=\delta$	if $a\in H$
(D3)	$\partial_H(x+y)=\partial_H(x)+\partial_H(y)$	
(D4)	$\partial_H(xy)=\partial_H(x)\partial_H(y)$	

outgoing transitions; hence, the transition system is pure (has pure termination). 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; see, for example, [15].

We proceed now to extend these axiom systems with the constant τ for the silent step and with axioms characterizing orthogonal bisimulation equivalence. The signature for the axiom system $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$ is obtained by extending the signature of ACP(A,γ) with the fresh constant τ and with a unary renaming operator τ_I for every set $I\subseteq A$. Let $A_\tau=A\cup\{\tau\}$ and $A_{\delta\tau}=A_\delta\cup\{\tau\}$. Its axioms are listed in Tables 1 and 2, and we now let a and b range over $A_{\delta\tau}$ in the axioms of Table 1. The conditions in the *compression* axioms O1–O3 are of the form $\tau x=\tau\tau x$. Such a condition is true for x if and only if the process x does not equal deadlock and all initial actions of x equal τ. In the operational semantics, we take A_τ as the set of transition labels; the silent action is simply executed like the other actions (see Table 3).

The subsystems $\mathrm{BPA}_\tau^{\mathrm{orth}}(A)$ and $\mathrm{BPA}_{\delta\tau}^{\mathrm{orth}}(A)$ are the extensions of BPA(A) and BPA$_\delta(A)$ with τ and the compression axioms O1–O3. It is straightforward to verify that the axioms in Table 2 are sound with respect to rooted orthogonal bisimilarity; the proofs can be found in Section A.1.

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)	$\tau_I(a) = \tau$	if $a \in I$
(TI3)	$\tau_I(x+y) = \tau_I(x) + \tau_I(y)$	
(TI4)	$\tau_I(xy) = \tau_I(x)\tau_I(y)$	

Table 3
Transition rules

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

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

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

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

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

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

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

Theorem 10. *Rooted orthogonal bisimilarity is a congruence with respect to all operators of* $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$.

Proof. See the appendix (Section A.2).

We end this section with two separate remarks.

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

Table 4
Branching bisimulation axioms

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

This proposition can be proved straightforwardly by induction on the structure of terms; for example, we derive using axioms A3 and O3:

$$\tau(\tau\tau+\tau) = \tau(\tau(\tau+\tau)+\tau) = \tau(\tau+\tau) = \tau\tau.$$

Second, rooted *branching* bisimilarity is axiomatized by axioms B1 and B2, see Table 4. 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

$$\mathrm{B1}+\mathrm{B2} \vdash \mathrm{O1}+\mathrm{O2}+\mathrm{O3} \quad \text{and} \quad \mathrm{B1}+\mathrm{O3} \vdash \mathrm{B2}.$$

5. Completeness

We prove completeness of the axiom system $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$, that is, we prove that any two rooted orthogonally bisimilar closed terms are derivably equal. The proof is based on Lemma 6 and the completeness of $\mathrm{BPA}_{\delta}(A)$ with respect to strong bisimulation: we show that terms are derivably equal to terms with only compact successors, and for these terms strong bisimilarity coincides with rooted orthogonal bisimilarity. The completeness of $\mathrm{BPA}^{\mathrm{orth}}_{\tau}(A)$ (without deadlock) can be proved similarly; this proof is omitted. We state that $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$ is a conservative extension of $\mathrm{BPA}^{\mathrm{orth}}_{\tau}(A)$.

The completeness of $\mathrm{ACP}^{\mathrm{orth}}_{\tau}(A,\gamma)$ follows as an easy corollary from the completeness of $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$, since the operations for parallelism can easily be eliminated from terms: every closed $\mathrm{ACP}^{\mathrm{orth}}_{\tau}(A,\gamma)$ term is derivably equal to a closed $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$ 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 $\mathrm{ACP}^{\mathrm{orth}}_{\tau}(A,\gamma)$ is a conservative extension of $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$.

In the completeness proof we assume that terms are written as *basic terms*, that are defined inductively as follows.

Definition 11. Let A be the set of action symbols. Then:
(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.

We use the notation $\sum_i t_i$ to describe an alternative composition of processes t_i, where the parameter i ranges over some finite set of indices. (Recall that alternative composition is commutative and associative.) 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

$$\sum_i a_i t_i + \sum_j a_j,$$

where the t_i are basic terms and $a_i, a_j \in A_\tau$.

Lemma 12. *Every closed* $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$ *term is derivably equal to a basic term.*

Proof. Standard and thus omitted. □

Lemma 13. *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 14. *If* $t = \sum_{i\in I} \tau t_i + t'$ *for some nonempty finite index set* I, *with* t_i *compact and* $t \leftrightarrows_{\mathrm{o}} t_i$ *for all* i *in* I, *then* $t = \tau t_i + t'$ *for any* i *in* I.

Proof. Take any i and j from I. Since orthogonal bisimilarity is an equivalence, we have $t_i \leftrightarrows_{\mathrm{o}} t_j$. Since t_i, t_j are compact, we have by Lemma 4 that $t_i \leftrightarrows t_j$. By the completeness of BPA_δ with respect to strong bisimilarity we get $t_i = t_j$. The required identity follows by axiom A3.

Lemma 15. *Every closed* $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$ *term is derivably equal to a basic term that has only compact successors.*

Proof. Take any closed term t. By Lemma 12 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 $\sum_{i\in I} \tau u_i + u^{\mathrm{c}}$, where

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

$u \leftrightarrows_{\mathrm{o}} u_i$ for all i in I; $a_j \in A_\tau$ and $u \not\leftrightarrows_{\mathrm{o}} u_j$ 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 from I (if $I = \emptyset$, then u itself is compact). By Lemma 14, we find $u = \tau u_i + u^{\mathrm{c}}$.

We know that u_i is compact and that $u \leftrightarrows_{\mathrm{o}} u_i$. 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 $u_j \leftrightarrows_{\mathrm{o}} u'_j$. By Lemma 4 and the completeness of BPA_δ, 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 $u_i \not\leftrightarrows_o u_l$. We assume that L is nonempty; if it is not, then infer from $u \leftrightarrows_o u_i$ 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 13 and axiom O3, we find that $au = au_i$, where the right-hand side has compact successors. □

Theorem 16. *The system* $\mathrm{BPA}^{\mathrm{orth}}_{\delta\tau}(A)$ *is complete with respect to rooted orthogonal bisimilarity, that is, any two closed terms that are rooted orthogonally bisimilar, are derivably equal.*

Proof. Take any two rooted orthogonally bisimilar closed terms. By soundness and by Lemma 15 we may assume that all successors of these terms are compact. By Lemma 6 we have that they are strongly bisimilar. Derivability follows from the fact that $\mathrm{BPA}_\delta(A)$ is complete with respect to strong bisimilarity. □

Corollary 17. *The system* $\mathrm{ACP}^{\mathrm{orth}}_\tau(A,\gamma)$ *is complete with respect to rooted orthogonal bisimilarity.*

6. Priorities

We extend the axiom system $\mathrm{ACP}^{\mathrm{orth}}_\tau(A,\gamma)$ with the priority operator θ, introduced in ACP in [1] (for an overview of the use of priorities in process algebra, see [14]). Parameter of this operator is a partial ordering $\leqslant$ on the set A_τ of actions (we write $a<b$ or $b>a$ if $a \leqslant b$ and $a \not\equiv b$). If, for example, the priority ordering is given by $a>b$ and $a>c$, then action a has priority over b and over c. In this case we find that $\theta(a+b)=a$ and $\theta(b+c)=b+c$. Our approach is fully general in that the occurrence of τ in the ordering is completely unrestricted. 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 10.

The transition rules for the priority operator are in Table 5. For the axiomatization of the priority operator, we need the auxiliary operator $\lhd$. A process $x \lhd y$ behaves

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; $a,b\in A_{\delta\tau}$

(P1)	$a\lhd b=a$ if $a\not\leq b$
(P2)	$a\lhd b=\delta$ 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)	$\theta(x+y)=\theta(x)\lhd y+\theta(y)\lhd x$

as the part of x that has initial actions that do not have an initial action with higher priority in y. The axioms are in Table 6.

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

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

For another example, let the priority ordering be given by $c<b$. Consider terms $t\equiv a(\tau(b+c)+c)$ and $u\equiv a(b+c)$. These processes are rooted branching bisimilar, and hence identified by all process equivalences in [17]. Observe that none of these equivalences identifies $\theta(t)=a(\tau b+c)$ and $\theta(u)=ab$. We conclude that, in the setting with τ, the priority operator is not a congruence for the abstract process equivalences in [17]. Also observe the following: process t evolves into the process $\tau(b+c)+c$ by the execution of action a. The latter 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 Section A.2 it is proved that rooted orthogonal bisimilarity is a congruence with respect to the priority operator in the setting with τ. Soundness of the priority axioms with respect to this equivalence follows from their soundness with respect to strong bisimulation equivalence and the fact that orthogonally bisimilar processes have the same initial actions (for all actions including τ). We state without proof 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 the priority operator can be eliminated from terms, which is easy to verify.

Note. Various ways for dealing with the priority operators in abstract semantics have been proposed. A first, classical approach is to eliminate all priority operators *before* applying abstraction. Another approach was advocated by Bol and Groote [12], 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: assume that I is a set of internal actions, all of which have the same priority as τ. Then we have that τ_I and θ commute modulo orthogonal bisimilarity:

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

which is the strongest commutation result that can be expected.

7. Recursion operators 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 recursion *operators* [6,10]. As the most basic of these we consider the binary Kleene star operator $*$, defined by

$$x^* y = x(x^* y) + y.$$

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 BPA, axioms for the $*$ are BKS1–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 [16] it is shown that $\mathrm{BPA}^*(A)$ axiomatizes bisimilarity over that signature. The system $\mathrm{ACP}^*(A,\gamma)$ is defined by adding the axioms BKS1–BKS4. In the setting with τ and the binary Kleene star, the system $\mathrm{BPA}_\tau^{\mathrm{orth}*}$ is defined by extending $\mathrm{BPA}^*(A)$ with the axioms O1–O3 (see Table 2) and axioms O4 and O5 given in Table 7. Note that these last two axioms are easily proved valid in orthogonal bisimulation equivalence. Finally, the system $\mathrm{ACP}_\tau^{\mathrm{orth}*}(A,\gamma)$ 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 the binary Kleene star: only finite

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)	$x((\tau + \tau\tau)^*y) = x((\tau\tau)^*y)$

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

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

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

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

state processes are definable. This restriction can be relaxed by adding the *push-down* operator ($, see [7,10]), defined by the axiom

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

We write $E^{\$}$ for the inclusion of the push-down axiom in axiom system E. The transition rules are as expected, and given in Table 8.

With the push-down operator also nonregular processes can be defined. A typical example is the term R given by

$$R = (succ(succ^{\$}\, pred) + zero)^* exit.$$

This term can be recognized as a definition of a register, modelling a memory location for a natural number with unbounded capacity and restricted access by a successor action, a predecessor action, a zero test action, and an exit or termination action. A graphical representation of the process R is given in Fig. 2. Observe that a register holding value n is modelled by the process

$$R(n) = (succ^{\$}\, pred)^n R.$$

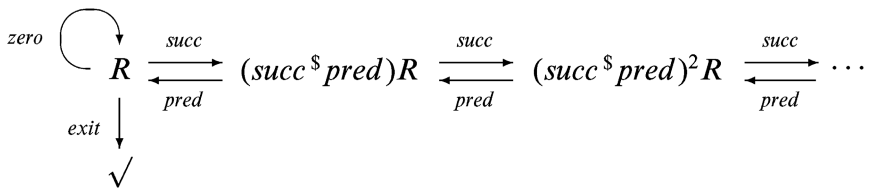


Fig. 2. The register process.

In ACP$^{*\$}$ registers can synchronize with terms representing register machine programs. As an example, let

$$H = \{a, a' \mid a = succ,\ pred,\ zero,\ exit\}$$

and $\gamma(a, a') = i$ for $a = succ,\ pred,\ zero,\ exit$. Then termination of a register holding value n can be described by

$$\partial_H((pred')^* exit' \cdot x \parallel R(n)) = i^{n+1} \cdot \partial_H(x).$$

For indexed registers R_1 and R_2, transfer of the value of R_1 to R_2 is described by

$$\partial_H((pred_1' \cdot succ_2')^* zero_1' \cdot x \parallel R_1(n) \parallel R_2(m)),$$

where the communication function and the set H are appropriately adjusted to the indexing of the register actions. It is not difficult to derive that this term is equal to

$$i^{2n+1} \cdot \partial_H(x \parallel R_1(0) \parallel R_2(n+m)).$$

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. For example, 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 [7,10], but we shall not use it.

Fairness. 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$_1^{\mathrm{b}}$) $\tau(\tau^* x) = \tau x$,

where FIR abbreviates Fair Iteration Rule. In the setting of rooted orthogonal bisimulation equivalence, we have the 'fairness axioms' given in Table 9. (If we consider

Table 9
Fairness axioms

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

processes modulo rooted divergence sensitive orthogonal bisimilarity, then of course axioms OFIR1 and OFIR2 are no longer valid.) In Section 10 we provide a protocol verification in which fairness is used.

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? To handle these questions we restrict to transition systems that have pure termination, or shortly, that are pure: 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 $\mathscr{T}$ up to some behavioral equivalence $\sim$ in axiom system E comes down to showing that for each state s in $\mathscr{T}$ different from $\surd$ there is a term t over E satisfying $s \sim t$.

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 $x=x\tau$.

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 $\mathscr{T}=(S,L,T)$, its *sequential τ-saturation* $\mathscr{T}_\tau$ is defined by (S_τ,L_τ,T_τ) where

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

Clearly, each pure transition system $\mathscr{T}=(S,L,T)$ is branching bisimilar to its sequential τ-saturation (which is recursive if $\mathscr{T}$ is). Therefore, the question how to express the sequential τ-saturation of pure transition systems up to rooted (divergence sensitive) orthogonal bisimulation equivalence is a relevant one in our setting: this is as close as we can get to the orthogonal world of concrete processes.

We view binary Kleene star and push-down as a modern alternative to the so-called 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 $\mathrm{ACP}_\tau^{\mathrm{orth}*}(A,\gamma)$, 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 18. *For each finite pure transition system $\mathscr{T}$ with finite label set L not containing τ, there is a finite extension A of L such that $\mathscr{T}_\tau$ can be expressed up to rooted divergence sensitive orthogonal bisimulation equivalence in* $\mathrm{ACP}_\tau^{\mathrm{orth}}(A,\gamma)$, *using only handshaking over $A\backslash L$, and either $*$ or* \$.

Proof. Assume that $\mathscr{T}$ has states $\{\surd, X_1, \ldots, X_n\}$ for some $n>0$. Then, for every j with $0<j\leqslant n$, 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, $\mathscr{T}_\tau$ 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 $\mathscr{T}_\tau$. Let A be the extension of L with the following $2n+3$ fresh actions:

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

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 lth process to start, and
r_l : $(l > 0)$ read instruction to start the lth process.

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

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

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

$$G = \left(\sum_{k=1}^{n} r_k F_k\right)^* s_0, \quad K = \left(\sum_{k=1}^{n} r_k s_k\right)^* r_0.$$

We derive

$$\begin{aligned}\partial_H(F_j G \,\|\, K) &= \partial_H\left(\left(\sum_{k=1}^{n} \alpha_{j,k} s_k G + \beta_j G\right) \,\|\, K\right)\\ &= \sum_{k=1}^{n} \alpha_{j,k} \cdot \partial_H(s_k G \,\|\, K) + \beta_j \cdot \partial_H(G \,\|\, K)\\ &= \sum_{k=1}^{n} \alpha_{j,k} \cdot i \cdot \partial_H(G \,\|\, s_k K) + \beta_j \cdot i\\ &= \sum_{k=1}^{n} \alpha_{j,k} \cdot i \cdot i \cdot \partial_H(F_k G \,\|\, K) + \beta_j \cdot i.\end{aligned}$$

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

$$X_j \leftrightarrows_{\mathrm{ordso}} \tau_{\{i\}} \circ \partial_H(F_j G \,\|\, K)$$

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

In the case of \$, consider process terms

$$M = \left(\sum_{k=1}^{n} r_k F_k\right)^{\$} s_0, \quad N = \left(\sum_{j=1}^{n} r_k s_k\right)^{\$} r_0.$$

Then

$$X_j \leftrightarrows_{\mathrm{ordso}} \tau_{\{i\}} \circ \partial_H(F_j M \,\|\, N)$$

for each $j = 1,\ldots,n$. This can be shown along the same lines, using a denumerable infinity of copies of the transitions of $\mathscr{T}_\tau$: let l range over the naturals and consider

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

Clearly, $X_j \leftrightarrows_{\mathrm{rdso}} Y_j(l)$ for each state X_j of $\mathscr{T}_\tau$ and each value of l. So it suffices to show that also

$$\tau_{\{i\}} \circ \partial_H(F_j M \,\|\, N) \leftrightarrows_{\mathrm{ordso}} Y_j(0).$$

We show this by first omitting the $\tau_{\{i\}}$-application:

$$\begin{aligned}\partial_H(F_j\cdot M^{l+1}\,\|\,N^{l+1}) &= \sum_{k=1}^{n}\alpha_{j,k}\cdot\partial_H(s_kM^{l+1}\,\|\,N^{l+1})+\beta_j\cdot\partial_H(M^{l+1}\,\|\,N^{l+1})\\ &= \sum_{k=1}^{n}\alpha_{j,k}\cdot i\cdot\partial_H(M^{l+1}\,\|\,s_kN^{l+2})+\beta_j\cdot i^{l+1}\\ &= \sum_{k=1}^{n}\alpha_{j,k}\cdot i^2\cdot\partial_H(F_k\cdot M^{l+2}\,\|\,N^{l+2})+\beta_j\cdot i^{l+1}.\end{aligned}$$

Hence, applying $\tau_{\{i\}}$ and axiom O1, we find for each l that

$$\begin{aligned}&\tau_{\{i\}}\circ\partial_H(F_j\cdot M^{l+1}\,\|\,N^{l+1})\\ &\quad=\sum_{k=1}^{n}\alpha_{j,k}\tau\cdot\tau_{\{i\}}\circ\partial_H(F_k\cdot M^{l+2}\,\|\,N^{l+2})+\beta_j\tau,\end{aligned}$$

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

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 *nonregular*, 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 $\mathrm{ACP}_\tau^{\mathrm{orths}\$}(A,\gamma)$ 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 [10] (also recorded in [7]).

We sketch a proof of the expressibility of pure recursive transition systems with bounded fan-out. An example is given in Section 9. This proof is based on a characterization of register machine computations (see, for example, [28]) in process algebra (a detailed explanation can be found in [11]). Recall the straightforward representation of registers presented in the previous section. 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 that computes f using three registers. Then there is a context $\mathrm{Con}[\_](n)$ where n refers to a register value such that

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

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, $\mathrm{Con}[\_](n)$ can be extended to $\mathrm{Con}[\_](n_1,\ldots,n_k)$ for the computation of $k>1$ computable functions in a

sequential fashion:

$$\mathrm{Con}[P_1 \cdots P_k x](n,0,\ldots,0) \leftrightarrows_{\mathrm{o}} i^{g'(n)} \cdot \mathrm{Con}[x](f_1(n),\ldots,f_k(n))$$

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

Now let $\mathscr{T}=(S,L,T)$ be a recursive pure transition system with S a set of naturals containing 0 and fan-out bounded by m. The state 0 is the termination state (so $\surd=0$).[4] With the above implementation scheme at hand, it is not hard to express the sequential τ-saturation of $\mathscr{T}$ 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 (that is, no outgoing transitions and either successful termination or deadlock). If a state has at least one outgoing transition, then its menu is a list $a_1,\ldots,a_k$, with $1\leqslant k\leqslant m$, of labels of its outgoing transitions. The ordering of these labels is arbitrary but fixed: for every multiset of labels there is at most one menu. Fix an enumeration of these menus, such that menu number 0 stands for successful termination and 1 stands for deadlock.

Let furthermore the transition relation T be characterized by $(m+1)$-tuples fetching all outgoing transitions (at most m): a state s yields the map

$$(s,0,\ldots,0)\mapsto(s_1,\ldots,s_{m+1}),$$

where $s_j=0$ for $1\leqslant j\leqslant m+1$ if $s=0$, and otherwise

- s_{m+1} gives the menu number of s, and
- s_j for $1\leqslant j\leqslant m$ is the target associated with source s and the jth label of menu s_{m+1} if such a transition is present, and 0 otherwise.

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

$$\mathrm{Con}[Px](s,0,\ldots,0)\xrightarrow{i}\!\!\!\!\rightarrow \mathrm{Con}[x](s_1,\ldots,s_{m+1}),$$

where $\xrightarrow{i}\!\!\!\!\rightarrow$ is the transitive closure of $\xrightarrow{i}$.

Furthermore, it is straightforward to define a process term M that interprets the menu s_{m+1} in the following way:

$$\mathrm{Con}[Mx](s_1,\ldots,s_m,s_{m+1})\xrightarrow{i}\!\!\!\!\rightarrow \begin{cases} \mathrm{Con}[\delta x](s_1,\ldots,s_m,0) & \text{if } s_{m+1}=1,\\ \mathrm{Con}\left[\sum_{j=1}^{k} a_j F_j x\right](s_1,\ldots,s_m,0) & \text{if } s_{m+1}>1. \end{cases}$$

(In case $s_{m+1}=0$, the process M blocks.) Here the a_j's and k are prescribed by the menu and F_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

$$\mathrm{Con}[PQ](s,0,\ldots,0),$$

[4] *Note*: In [11] the number 0 was reserved for the deadlock state. Here the choice to let 0 stand for successful termination allows a more elegant presentation.

where

$$Q = (MP)^* exit'_{m+1} E.$$

The exit action synchronizes with the termination action $exit_{m+1}$ of the menu register in case it holds value 0, and E terminates all remaining processes. Finally, applying $\tau_{\{i\}}$ we can express $\mathscr{T}_\tau$ up to rooted divergence sensitive orthogonal bisimilarity: for each $s \in S \subseteq S_\tau \backslash \{\surd\}$ it follows that if s has menu $a_1, \ldots, a_k$ then it has transitions

$$s \xrightarrow{a_j} (s_j)_\tau \xrightarrow{\tau} s_j$$

for $1 \leqslant j \leqslant k$, and

$$s \leftrightarrows_{\text{o rdso}} \tau_{\{i\}} \left(\text{Con} \left[\sum_{j=1}^{k} a_j F_j PQ \right] (s_1, \ldots, s_k, \ldots, s_m, 0) \right)$$

(where $s_l = 0$ for $k < l \leqslant m$) in the combined transition system. In the case that s has no transitions, it holds that $s \leftrightarrows_{\text{rdso}} \delta$. In the next section we provide an example.

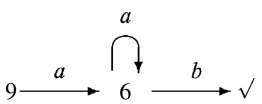
Following the proof of Theorem 18 above, it is straightforward how this approach should be adapted to $\text{ACP}_\tau^{\text{orth\$}}(A, \gamma)$ (thus, without $*$, cf. the related results in [11]). The above can be summarized as follows:

Theorem 19. *For each recursive pure transition system $\mathscr{T}$ with finite label set L not containing τ and bounded fan-out, there is a finite extension A of L such that $\mathscr{T}_\tau$ can be expressed up to rooted divergence sensitive orthogonal bisimulation equivalence in* $\text{ACP}_\tau^{\text{orth}}(A, \gamma)$, *using only handshaking over $A \backslash L$, and either* \$, *or both* $*$ *and* \$.

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

9. Expressiveness: illustration

In this section, we give an example of the expression of (the sequential τ-saturation of) transition systems using register-machine based processes, as presented in Section 8. We consider the case of recursive pure transition systems over label set $\{a, b\}$, and with fan-out of at most two. The states are naturals and 0 is the termination state (so $\surd = 0$). As an example we shall find a process algebraic expression for the state 9 in the sequential τ-saturation of the transition system (1).



(1)

This τ-saturation is given below (2).

$$9 \xrightarrow{a} 6_\tau \underset{a}{\overset{\tau}{\rightleftarrows}} 6 \xrightarrow{b} \surd_\tau \xrightarrow{\tau} \surd \tag{2}$$

Clearly, it is not difficult to find an expression for state 9; for example, using the binary Kleene star operator, we express state 9 in (1) as $a(a^*b)$, and in (2) as the process term

$$a\tau((a\tau)^*b\tau).$$

Here, we shall give another (more complex) expression for this state following the procedure outlined in Section 8.

We start with a menu enumeration for pure recursive transition systems over $\{a,b\}$ and with fan-out bounded by two:

0	for successful termination,	4	for a,a,
1	for deadlock,	5	for a,b,
2	for a,	6	for b,b.
3	for b,		

For example, state 6 in transition system (1) has one outgoing a-transition and one outgoing b-transition; hence it has menu number 5.

We can characterize the transition relation of a particular transition system by a mapping on 3-tuples of naturals as follows: a state s yields the map

$$(s,0,0) \mapsto (s_1,s_2,s_3),$$

where s_3 gives the menu number of state s, and s_j for $j=1,2$ is the target state associated with source s and the jth label of menu s_3 if such a transition is present, and otherwise 0.

For example, in the case of transition system (1), we find that the transitions are given by

$$(9,0,0) \mapsto (6,0,2),$$
$$(6,0,0) \mapsto (6,0,5).$$

(Recall that 0 is the termination state.)

Let P be such that it models the computation of this mapping, that is, if $(s,0,0) \mapsto (s_1,s_2,s_3)$, then

$$\mathrm{Con}[Px](s,0,0) \xrightarrow{i} \mathrm{Con}[x](s_1,s_2,s_3).$$

We define the menu interpretation process M partly in a graphical manner in Fig. 3. The processes F_j in the definition of M are the processes that clean up the registers after the execution of an action; they write the value of the newly reached state in the

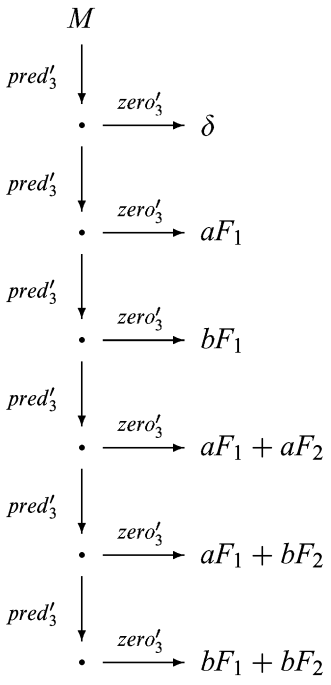


Fig. 3. The menu process M.

first register and empty all the others. In this case they are given by

$$F_1 = pred'_2{}^* zero'_2,$$
$$F_2 = (pred'_1{}^* zero'_1)((pred'_2 succ'_1)^* zero'_2).$$

We see that the process F_1 just empties the second register. The process F_2 first empties the first register and then transfers the contents of the second register to the first register. Observe that the third register, the menu number, has already been emptied.

We write Q for $(MP)^* exit'_3 E$. Now, state 9 in (2) is expressed by

$$\tau_{\{i\}}(\mathrm{Con}[aF_1PQ](6,0,0))$$

(see Fig. 4), that is,

$$9 \leftrightarrows_{\mathrm{ordso}} \tau_{\{i\}}(\mathrm{Con}[aF_1PQ](6,0,0)),$$

as the only difference is in the length of τ-sequences.

10. Verification of a PAR protocol

In this section, we consider the Positive Acknowledgment and Retransmission (PAR) protocol [32]. This protocol describes the transmission of data from a sender process

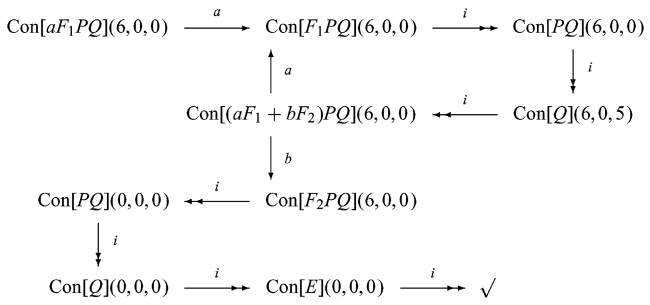


Fig. 4. Transition system.

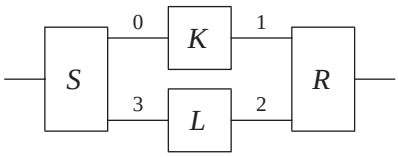


Fig. 5. Architecture of the protocol.

S to a receiver process R over unreliable channels, such that the external behavior corresponds to that of a one-place buffer. We prove the correctness of this protocol, that is, we first give the description of an implementation and then apply abstraction. Next, we compress the τ-actions and show that the resulting term indeed matches the behavior of a one-place buffer.

The architecture of the protocol is depicted in Fig. 5. We refer to [33] 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) acknowledgment for the arrival of a frame. The receiver sends its acknowledgments 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 acknowledgment, or fails to do anything. A dummy internal action i is added to make the choice between these options nondeterministic.

The sender and the receiver act in response to received data only. This poses a problem, because, when one of the channels K and L fails to act upon receiving a message, the system will be waiting for nothing to happen, that is, it will deadlock. To avoid this, a time-out action may occur, that is supposed to reactivate the system if it threatens to deadlock. In [32], 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 time-out action lower priority than any other action; it occurs if no other activity is possible. After a time-out, the sender retransmits the last message.

Table 10
Specification

$S=(S_0\cdot S_1)^*\delta$
$S_n=\sum_{d\in D} r(d)\cdot s_0(dn)\cdot S_{dn}$
$S_{dn}=(time\_out\cdot s_0(dn))^* r_3$
$K=(\sum_{f\in F} r_0(f)\cdot K_f)^*\delta$
$K_f=i\cdot s_1(f)+i\cdot s_1(\bot)+i$
$L=(r_2\cdot(i\cdot s_3+i))^*\delta$
$R=(R_0\cdot R_1)^*\delta$
$R_n=(r_1(\bot)+\sum_{d\in D} r_1(d(1-n))\cdot s_2)^*\sum_{d\in D} r_1(dn)\cdot s(d)\cdot s_2$

10.1. Specification

We assume a finite data set D and a set of frames $F=D\times\{0,1\}$. The components are specified in Table 10.

The s_k and r_k actions, with $k\leqslant 3$, 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 $k=2,3$, and

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

for all messages m and $k=0$, 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 acknowledgment from L to S. The time-out action is performed only if no other actions are enabled; the partial priority ordering is defined by

$$time\_out < a$$

for all actions a other than the time-out action.

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

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

10.2. Verification

Notation: $[\_]$ for $\theta(\partial_H(\_))$. We use the abbreviations given in Table 11. Driving the composed operator $\theta\circ\partial_H$ inwards we leave out all summands that are blocked by the encapsulation or the priority operator. In particular this means that, in the presence of alternatives, summands starting with a time-out action are renamed to δ. The derivation of the linearization may be found in Table 12. The linearization is illustrated in Fig. 6.

Table 11
Abbreviations

$$X_d^0 = [S_{d0}S_1S \parallel K_{d0}K \parallel R \parallel L]$$

$$X_d^1 = [S_{d0}S_1S \parallel K \parallel R \parallel L]$$

$$X_d^2 = [S_{d0}S_1S \parallel K \parallel R_1R \parallel (i \cdot s_3 + i)L]$$

$$X_d^3 = [S_{d0}S_1S \parallel K \parallel R_1R \parallel L]$$

$$X_d^4 = [S_{d0}S_1S \parallel K_{d0}K \parallel R_1R \parallel L]$$

$$Y = [S_1S \parallel K \parallel R_1R \parallel L]$$

Table 12
Linearization

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

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

$$\begin{aligned}
X_d^1 &= \mathit{time_out} \cdot [s_0(d0) \cdot S_{d0}S_1S \parallel K \parallel R \parallel L] \\
&= \mathit{time_out} \cdot c_0(d0) \cdot X_d^0
\end{aligned}$$

$$\begin{aligned}
X_d^2 &= i \cdot [S_{d0}S_1S \parallel K \parallel R_1R \parallel s_3 \cdot L] + i \cdot X_d^3 \\
&= i \cdot c_3 \cdot Y + i \cdot X_d^3
\end{aligned}$$

$$\begin{aligned}
X_d^3 &= \mathit{time_out} \cdot [s_0(d0) \cdot S_{d0}S_1S \parallel K \parallel R_1R \parallel L] \\
&= \mathit{time_out} \cdot c_0(d0) \cdot X_d^4
\end{aligned}$$

$$\begin{aligned}
X_d^4 &= i \cdot [S_{d0}S_1S \parallel s_1(d0) \cdot K \parallel R_1R \parallel L] \\
&\quad + i \cdot [S_{d0}S_1S \parallel s_1(\perp) \cdot K \parallel R_1R \parallel L] + i \cdot X_d^3 \\
&= i \cdot c_1(d0) \cdot [S_{d0}S_1S \parallel K \parallel s_2 \cdot R_1R \parallel L] + (i \cdot c_1(\perp) + i) \cdot X_d^3 \\
&= i \cdot c_1(d0) \cdot c_2 \cdot X_d^2 + (i \cdot c_1(\perp) + i) \cdot X_d^3
\end{aligned}$$

Using RSP* we derive from the equations for X derived in Table 12:

$$\begin{aligned}
X_d^0 &= ((i \cdot c_1(\perp) + i) \cdot \mathit{time_out} \cdot c_0(d0))^* i \cdot c_1(d0) \cdot s(d) \cdot c_2 \cdot X_d^2, \\
X_d^2 &= (i \cdot \mathit{time_out} \cdot c_0(d0) \cdot \widetilde{X}_d^4)^* i \cdot c_3 \cdot Y, \\
\widetilde{X}_d^4 &= ((i \cdot c_1(\perp) + i) \cdot \mathit{time_out} \cdot c_0(d0))^* i \cdot c_1(d0) \cdot c_2.
\end{aligned}$$

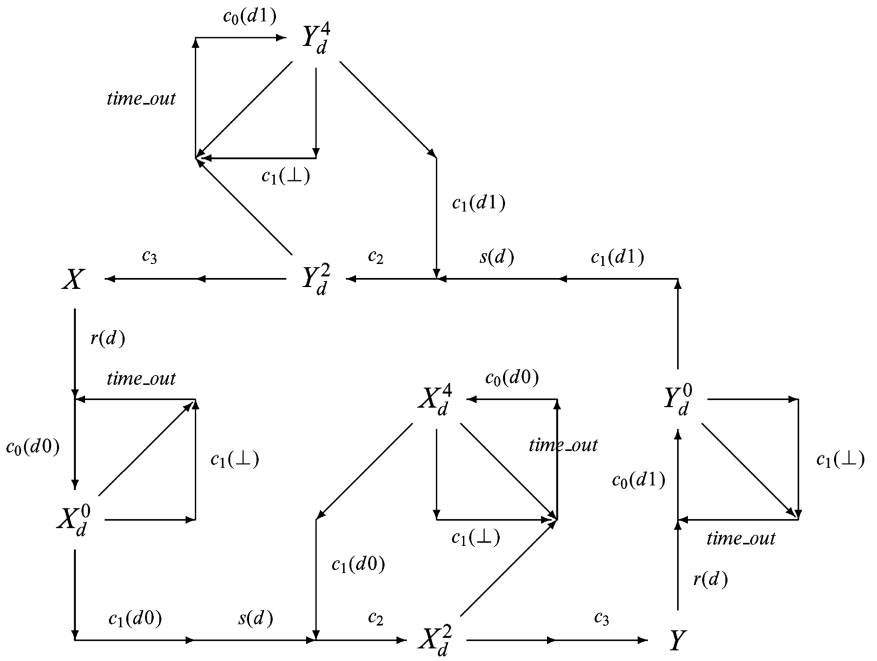


Fig. 6. Illustration of the process X.

In a similar way we can derive

$$
\begin{aligned}
Y &= \sum_{d\in D} r(d)\cdot c_0(d1)\cdot Y_d^0,\\
Y_d^0 &= ((i\cdot c_1(\perp)+i)\cdot time\_out\cdot c_0(d1))^* i\cdot c_1(d1)\cdot s(d)\cdot c_2\cdot Y_d^2,\\
Y_d^2 &= (i\cdot time\_out\cdot c_0(d1)\cdot \tilde{Y}_d^4)^* i\cdot c_3\cdot X,\\
\tilde{Y}_d^4 &= ((i\cdot c_1(\perp)+i)\cdot time\_out\cdot c_0(d1))^* i\cdot c_1(d1)\cdot c_2.
\end{aligned}
$$

Using substitution we eliminate all process abbreviations from the equation for X, yielding terms X' and Y' such that $X = X'Y$ and $Y = Y'X$. We derive by pressing the operator τ_I inwards and compressing τ's (using the axioms from Table 2) that

$$\tau_I(X') = \tau_I(Y') = \sum_{d\in D} r(d)\cdot\tau(\tau^*\tau)\tau\cdot s(d)\cdot\tau((\tau\tau(\tau^*\tau)\tau)^*\tau)\tau.$$

Using axiom OFIR2 we derive that

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

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

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

Clearly, the right-hand term matches the behavior of a one-place buffer.

Remark 20. In [34], a verification of the Concurrent Alternating Bit Protocol (CABP) with respect to orthogonal bisimulation can be found. This protocol 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

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

11. 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 of internal activity, but is not as abstract as the common behavioral equivalences that deal with abstraction. In particular, it is a finer equivalence than branching bisimilarity [22]. 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." [22, 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,[5] and it is the *finest* congruence possible for a common repertoire of process algebra operators (supporting the interleaving hypothesis) that is abstract in the sense that it satisfies at least $a\tau x = ax$ (cf. [18,22]).

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

[5] This notion is formally defined in [19].

recovery of an undesirable event, and fairness is assumed. In Section 10 it is shown that such events in our modelling 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 20). 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 [19] depends on a notion of *observable content* of the traces of that process; it might well be that the "compression content" of traces, that is, 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 [19]. 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, for example, [5]) that the priority operator is not compatible with failure or ready semantics. [6]

Appendix A. Soundness and congruence proofs

In Section A.1 we prove that the compression axioms are sound with respect to rooted orthogonal bisimulation equivalence. In Section A.2 we prove that this equivalence is a congruence with respect to the operators of $\mathrm{ACP}_{\tau\theta}^{\mathrm{orth}}$.

A.1. Soundness proofs

We prove that axioms O1–O3 (see Table 2) are sound with respect to rooted orthogonal bisimulation equivalence for any of the axiom systems that we introduced. We repeat these axioms here for convenience.

$$x\tau\tau = x\tau, \tag{O1}$$

[6] 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 $\theta(P)\not\equiv_F \theta(Q)$.

$$x\tau(y+z) = x(y+z) \qquad \text{if } \tau y = \tau\tau y,\ \tau z = \tau\tau z, \tag{O2}$$

$$x(\tau(y+z)+z) = x(y+z) \quad \text{if } \tau y = \tau\tau y. \tag{O3}$$

The soundness proofs for the TI axioms (also in Table 2) are trivial and therefore omitted here.

Let A be the set of action symbols not containing τ, let $A_\tau = A \cup \{\tau\}$, and write P for the set of closed terms. Let Id be the identity relation on $P \cup \{\surd\}$.

The conditions in axioms O2 and O3 are of the form $\tau x = \tau\tau x$, by which we require that the process x starts with a silent step:

Lemma A.1. *For all $p \in P$, if $\tau p \leftrightarrows_{\mathrm{ro}} \tau\tau p$, then $p \xrightarrow{\tau}$ and, for all $a \in A_\tau$, $p \xrightarrow{a}$ implies $a = \tau$.*

Proof. From the fact that τp and $\tau\tau p$ are *rooted* orthogonally bisimilar, it follows that p and τp are orthogonally bisimilar. Orthogonally bisimilar processes have the same initial actions. □

We prove that the compression axioms O1–O3 are sound with respect to rooted orthogonal bisimulation equivalence using induction on the length of derivations.

Axiom O1: We must show that $p\tau\tau \leftrightarrows_{\mathrm{ro}} p\tau$ for all $p \in P$. The symmetric closure of the relation

$$\{(p'\tau\tau, p'\tau),\ (\tau\tau, \tau),\ (\tau, \tau),\ (\surd, \surd) \mid p' \in P\}$$

is an orthogonal bisimulation that is rooted between $p\tau\tau$ and $p\tau$ for all $p \in P$.

Axiom O2: Let t_i be a term, for $i = 0, 1, 2$, and suppose that $\tau t_i = \tau\tau t_i$ has a sound derivation for $i = 1, 2$ (induction hypothesis). Suppose that from these derivations we derive

$$t_0\tau(t_1 + t_2) = t_0(t_1 + t_2)$$

in one step using axiom O2.

Assume an arbitrary closed interpretation that maps t_i to p_i for $i = 0, 1, 2$. We need to show that

$$p_0\tau(p_1 + p_2) \leftrightarrows_{\mathrm{oro}} p_0(p_1 + p_2).$$

By induction hypothesis we have that $\tau p_i \leftrightarrows_{\mathrm{o\,ro}} \tau\tau p_i$ for $i = 1, 2$.

Let R be the symmetric closure of the relation

$$Id \cup \{(p\tau(p_1 + p_2), p(p_1 + p_2)),\ (\tau(p_1 + p_2), p_1 + p_2) \mid p \in P\}.$$

It is not difficult to show that R is an orthogonal bisimulation; the case to check is the pair containing $\tau(p_1 + p_2)$ and $p_1 + p_2$.

The second process can match the τ-step by the first process with an empty τ-path, and it follows from Lemma A.1 that it can do a τ-step.

If the second process takes a step, say a step from p_1 to p_1', then we know by Lemma A.1 that this must be a τ-step. The first process can match this step with the path

$$\tau(p_1 + p_2) \xrightarrow{\tau} p_1 + p_2 \xrightarrow{\tau} p_1',$$

which clearly meets the requirements.

Finally, R is clearly rooted between $p_0\tau(p_1 + p_2)$ and $p_0(p_1 + p_2)$.

Axiom O3: This case is similar to the previous case. Let t_i be a term, for $i = 0, 1, 2$. Suppose (induction hypothesis) that $\tau t_1 = \tau\tau t_1$ has a sound derivation. Suppose that from this derivation we derive

$$t_0(\tau(t_1 + t_2) + t_2) = t_0(t_1 + t_2)$$

in one step using axiom O3.

Assume an arbitrary closed interpretation that maps t_i to p_i for $i = 0, 1, 2$. We need to show that

$$p_0(\tau(p_1 + p_2) + p_2) \leftrightarrows_{\mathrm{oro}} p_0(p_1 + p_2).$$

By induction hypothesis we have that $\tau p_1 \leftrightarrows_{\mathrm{oro}} \tau\tau p_1$.

Let R be the symmetric closure of the relation

$$Id \cup \{(p(\tau(p_1 + p_2) + p_2), p(p_1 + p_2)),\ (\tau(p_1 + p_2) + p_2, p_1 + p_2) \mid p \in P\}.$$

It is not difficult to show that R is an orthogonal bisimulation; the case to check is the pair containing $\tau(p_1 + p_2) + p_2$ and $p_1 + p_2$.

The second process can match the τ-step to $p_1 + p_2$ by the first process with an empty τ-path, and it follows from Lemma A.1 that it has at least one τ-step. If the first process takes a step from p_2, then the second process can match this step directly with the same step from p_2.

Next we look at steps taken by the second process. We know by Lemma A.1 that a step from p_1, say to p_1', must be a τ-step. The first process can match such a step with the path

$$\tau(p_1 + p_2) + p_2 \xrightarrow{\tau} p_1 + p_2 \xrightarrow{\tau} p_1'.$$

Otherwise, if the second process takes a step from p_2, then the first process can match this step directly with the same step from p_2.

Finally, R is clearly rooted between $p_0(\tau(p_1 + p_2) + p_2)$ and $p_0(p_1 + p_2)$.

A.2. 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 and 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 $t_i\leftrightarrows_{\mathrm{ro}} u_i$. We prove that $t_1\diamond t_2\leftrightarrows_{\mathrm{ro}} u_1\diamond u_2$ for all $\diamond\in\{\cdot,\|,\lfloor\!\lfloor,|\}$, and that $\dagger(t_1)\leftrightarrows_{\mathrm{ro}}\dagger(u_1)$ 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:

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

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

A.2.1. 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 tR_1u and vR_2w by definition of R'. Since R_1 and R_2 are symmetric, we find 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.

A.2.2. 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. 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 $y\xrightarrow{a}w$.
(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, we have $u\xrightarrow{b}\surd$ 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 $y \xrightarrow{a} w$.

(2) $t \xrightarrow{a} t'$ and $x' = t' \between v$. In this case $u \xrightarrow{a} u'$ with $t'R_1u'$. Then $y \xrightarrow{a} u' \between w$. Using the definition of R', we get $x'Ru' \between w$.

- If $x \equiv t \mid v$, then, by definition of R', $y \equiv u \mid w$ 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 we have $u \xrightarrow{b} \surd$ 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 $x' = t' \between v'$. 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 $y \xrightarrow{a} \surd$. 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 t does not have a b-step with $b > a$. It follows from tR_1u, that $u \xrightarrow{a} u'$ for some u' with $t'R_1u'$ and u does not have a b-step with $b > a$. Hence $y \xrightarrow{a} \theta(u')$ and $x'R'\theta(u')$.

A.2.3. 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 $n \geqslant 0$, there are y_k such that $y_0 \cdots y_n$ in τ-$paths(y)$ and $x'Ry_n$ and xRy_k for all $k < n$.

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 τ-$paths(u)$ 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 τ-$paths(y)$. 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 $n \geqslant 0$, there are states u_i such that $u_0 \cdots u_n$ in τ-$paths(u)$ and $t'R_1u_n$ and tR_1u_i for all $i < n$. Then $(u_0 \cdot w) \cdots (u_n \cdot w)$ in τ-$paths(y)$. We have $x'R_2u_n \cdot w$, and $xR'u_i \cdot w$ for $i < n$ by definition of R'.
- If $x \equiv t \between v$, then, by definition of R', $y \equiv u \between w$ 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 τ-$paths(u)$ and $u_n = \surd$ and tR_1u_i for all $i < n$.

Then also $(u_0 \parallel w)\cdots(u_{n-1} \parallel w)w$ in τ-$paths(y)$, and, by definition of R', $xR'u_i \parallel w$ for all $i<n$.

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

(3) $v \xrightarrow{\tau} \surd$ and $x' = t$. Like case (1).

(4) $v \xrightarrow{\tau} v'$ and $x' = t \parallel v'$. 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 τ-$paths(u)$ and $u_n = \surd$ and tR_1u_i for all $i<n$. Then also $(u_0 \between w)(u_1 \parallel w)\cdots(u_{n-1} \parallel w)w$ in τ-$paths(y)$, and, by definition of R', $xR'u_i \parallel w$ for all $0<i<n$.

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

- If $x \equiv t \mid v$, 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 τ-$paths(u)$ 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 τ-$paths(y)$ 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 $n \geqslant 0$, there are u_i such that $u_0 \cdots u_n$ in τ-$paths(u)$ 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 τ-$paths(y)$ 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 τ-$paths(u)$ 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 τ-$paths(y)$ 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 $n \geqslant 0$, there are u_i such that $u_0 \cdots u_n$ in τ-$paths(u)$ 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 τ-$paths(y)$ 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 τ-$paths(u)$, $u_n = \surd$, and tR_1u_i for $i<n$. Since, for $i<n$, tR_1u_i, it follows that there is no $a>\tau$ with $u_i \xrightarrow{a}$.

Hence $\theta(u_0)\dots\theta(u_{n-1})\ u_n$ in τ-$paths(y)$. 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 $n \geqslant 0$, there are u_i such that $u_0 \dots u_n$ in τ-$paths(u)$, $t'R_1u_n$, and tR_1u_i for $i < n$. So $u \xrightarrow{\tau}$ and hence $y \xrightarrow{\tau}$. Since, for $i < n$, tR_1u_i, we find that there is no $a > \tau$ with $u_i \xrightarrow{a}$. Hence $\theta(u_0)\dots\theta(u_n)$ in τ-$paths(y)$. We have that $xR\theta(u_i)$ and $x'R\theta(u_n)$ by definition of R'.

A.2.4. 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 $\diamond \in \{\cdot, \|, \lfloor\!\lfloor, |\}$ 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 follows that $u_1 \xrightarrow{\tau} \surd$ and hence $u_1 \cdot u_2 \xrightarrow{\tau} u_2$.

(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 $t_1'R_1u_1'$. We see that $u_1 \cdot u_2 \xrightarrow{\tau} u_1' \cdot u_2$ and $xRu_1' \cdot u_2$.

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 follows that $u_1 \xrightarrow{\tau} \surd$ and hence $u_1 \| u_2 \xrightarrow{\tau} u_2$.

(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 $t_1'R_1u_1'$. We see that $u_1 \| u_2 \xrightarrow{\tau} u_1' \| u_2$ and $xRu_1' \| u_2$.

(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 $t_1 \lfloor\!\lfloor t_2 \xrightarrow{\tau} x$. 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 follows that $u_1 \xrightarrow{\tau} \surd$ and hence $u_1 \lfloor\!\lfloor u_2 \xrightarrow{\tau} u_2$.

(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 $t_1'R_1u_1'$. We see that $u_1 \lfloor\!\lfloor u_2 \xrightarrow{\tau} u_1' \| u_2$ and $xRu_1' \| u_2$.

Silent steps of $u_1 \lfloor\!\lfloor u_2$ 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 follows that $u_1 \xrightarrow{\tau} \surd$ and hence $\partial_H(u_1) \xrightarrow{\tau} \surd$.

(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 $t_1'R_1u_1'$. We see that $\partial_H(u_1) \xrightarrow{\tau} \partial_H(u_1')$ and $xR\partial_H(u_1')$.

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, we have that $u_1 \xrightarrow{\tau} \surd$ and hence $\tau_I(u_1) \xrightarrow{\tau} \surd$.

(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 $xR\tau_I(u_1')$.
(3) $t_1 \xrightarrow{a} \surd$ and $a \in I$ and $x = \surd$. From $t_1 R_1 u_1$ it follows that $u_1 \xrightarrow{a} \surd$ and hence $\tau_I(u_1) \xrightarrow{a} \surd$.
(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 $xR\tau_I(u_1')$.

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 $t_1 R_1 u_1$, we find 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 $t_1 R_1 u_1$, there is no $a > \tau$ with $u_1 \xrightarrow{a}$. Since R_1 is rooted between t_1 and u_1, it must be 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 $xR\theta(u_1')$.

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

A.2.5. Alternative composition

We prove that $\leftrightarrows_{\mathrm{ro}}$ is a congruence with respect to 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 $i \in \{1, 2\}$. It follows from $t_i R_i u_i$ that $u_i \xrightarrow{a} u$ for some u with $tR_i u$. Then also $u_1 + u_2 \xrightarrow{a} u$ and tRu.
- 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, we have $u_i \xrightarrow{\tau} u$ for some u with $tR_i u$. Then also $u_1 + u_2 \xrightarrow{\tau} u$ and tRu.

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

References

[1] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, Syntax and defining equations for an interrupt mechanism in process algebra, Fund. Inform. IX (1986) 127–168.
[2] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, On the consistency of Koomen's fair abstraction rule, Theoret. Comput. Sci. 51 (1/2) (1987) 129–176.
[3] J.C.M. Baeten, R.J. van Glabbeek, Another look at abstraction in process algebra (extended abstract), in: Th. Ottmann (Ed.), Proc. 14th Internat. Colloq. on Automata, Languages and Programming (ICALP'87), Lecture Notes in Computer Science, Vol. 267, Springer, Berlin, 1987, pp. 84–94.

[4] J.C.M. Baeten, C. Verhoef, Concrete process algebra, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Handbook of Logic in Computer Science, Vol. 4, Oxford University Press, Oxford, 1995, pp. 149–268.
[5] J.C.M. Baeten, W.P. Weijland, Process Algebra, Cambridge Tracts in Theoretical Computer Science, Vol. 18, Cambridge University Press, Cambridge, 1990.
[6] J.A. Bergstra, I. Bethke, A. Ponse, Process algebra with iteration and nesting, Comput. J. 37 (4) (1994) 243–258.
[7] J.A. Bergstra, W.J. Fokkink, A. Ponse, Process algebra with recursive operations, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, Amsterdam, 2001, pp. 333–389.
[8] J.A. Bergstra, J.W. Klop, Process algebra for synchronous communication, Inform. and Control 60 (1–3) (1984) 109–137.
[9] J.A. Bergstra, J.W. Klop, Algebra of communicating processes with abstraction, Theoret. Comput. Sci. 37 (1) (1985) 77–121.
[10] J.A. Bergstra, A. Ponse, Non-regular iterators in process algebra, Theoret. Comput. Sci. 269 (1–2) (2001) 203–229.
[11] J.A. Bergstra, A. Ponse, Register-machine based processes, J. ACM 48 (6) (2001) 1207–1241.
[12] R.N. Bol, J.F. Groote, The meaning of negative premises in transition system specifications, J. ACM 43 (5) (1996) 863–914.
[13] J.C. Bradfield, C. Sterling, Modal logics and mu-calculi: an introduction, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, Amsterdam, 2001, pp. 293–330.
[14] R. Cleaveland, G. Lüttgen, V. Natarjan, Priority in process algebra, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, Amsterdam, 2001, pp. 711–765.
[15] W.J. Fokkink, Introduction to Process Algebra, Texts in Theoretical Computer Science, Springer, Berlin, 2000.
[16] W.J. Fokkink, H. Zantema, Basic process algebra with iteration: completeness of its equational axioms, Comput. J. 37 (4) (1994) 259–267.
[17] 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. Extended abstract in: E. Best (Ed.), CONCUR'93, 4th Internat. Conf. on Concurrency Theory, Lecture Notes in Computer Science, Vol. 715, Springer, Berlin, 1993.
[18] R.J. van Glabbeek, A complete axiomatization for branching bisimulation congruence of finite-state behaviours, in: A.M. Borzyszkowski, S. Sokolowski (Eds.), Proc. Mathematical Foundations of Computer Science 1993 (MFCS), Lecture Notes in Computer Science, Vol. 711, Springer, Berlin, 1993, pp. 473–484.
[19] R.J. van Glabbeek, What is branching time semantics and why to use it? Bull. EATCS 53 (1994), 190–198; M. Nielsen (Ed.), The Concurrency Column.
[20] R.J. van Glabbeek, The linear time–branching time spectrum I; the semantics of concrete, sequential processes, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, Amsterdam, 2001, pp. 3–99.
[21] R.J. van Glabbeek, W.P. Weijland, Branching time and abstraction in bisimulation semantics (extended abstract), in: G.X. Ritter (Ed.), Information Processing 89, Proc. IFIP 11th World Computer Congress, North-Holland, Amsterdam, 1989, pp. 613–618, Full version in [22].
[22] R.J. van Glabbeek, W.P. Weijland, Branching time and abstraction in bisimulation semantics, J. ACM 43 (3) (1996) 555–600.
[23] M. Hennessy, R. Milner, Algebraic laws for nondeterminism and concurrency, J. ACM 32 (1985) 137–161.
[24] R. Milner, A Calculus of Communicating Systems, in: Lecture Notes in Computer Science, Vol. 92, Springer, Berlin, 1980.
[25] R. Milner, Modal characterisation of observable machine behaviour, in: G. Astesiano, C. Bohm (Eds.), Proc. CAAP 81, Lecture Notes in Computer Science, Vol. 112, Springer, Berlin, 1981, pp. 25–34.
[26] R. Milner, Calculi for synchrony and asynchrony, Theoret. Comput. Sci. 28 (3) (1983) 267–310.
[27] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, 1989.

[28] M.L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, Englewood Cliffs N.J., 1967.
[29] R. De Nicola, F.W. Vaandrager, Three logics for branching bisimulation, J. ACM 42 (2) (1995) 458–487.
[30] D.M.R. Park, Concurrency and automata on infinite sequences, in: P. Deussen (Ed.), 5th GI Conf, Lecture Notes in Computer Science, Vol. 104, Springer, Berlin, 1981, pp. 167–183.
[31] C. Sterling, Modal and temporal logics for processes, Logics for Concurrency, Lecture Notes in Computer Science, Vol. 1043, Springer, Berlin, 1996, pp. 149–237.
[32] A.S. Tanenbaum, Computer Networks, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1989.
[33] F.W. Vaandrager, Two simple protocols, in: J.C.M. Baeten (Ed.), Applications of Process Algebra, Cambridge Tracts in Theoretical Computer Science, Vol. 17, Cambridge University Press, Cambridge, 1990, pp. 23–45.
[34] M.B. van der Zwaag, Some verifications in process algebra with iota, Report P9806, Programming Research Group, University of Amsterdam, 1998.
[35] M.B. van der Zwaag, A verification in process algebra with iota, in: J.F. Groote, S.P. Luttik, J.J. van Wamel (Eds.), Proc. Third Internat. Workshop on Formal Methods for Industrial Critical Systems, CWI, Amsterdam, The Netherlands, 1998, pp. 347–368.