Models and logics for process algebra
van der Zwaag, M.B.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Orthogonal Bisimulation Equivalence

With Jan Bergstra and Alban Ponse

We propose a refinement of branching bisimulation equivalence that we call orthogonal bisimulation equivalence. Typically, internal activity (the performance of $\tau$-steps) may be compressed, but not completely discarded. Hence, a process with $\tau$-steps cannot be equivalent to one without $\tau$-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.

1. Introduction

In concurrency theory, Milner's observation equivalence as discussed in the setting of CCS (Calculus of Communicating Systems [68], cf. [70, 71]) is a standard example of a branching time behavioral equivalence that deals with abstraction. Here 'branching time' refers to the fact that the branching structure of processes is taken into account, and 'abstraction' refers to a mechanism to hide actions that are assumed not to be observable or interesting for some other reason. In the process algebraic approaches based on ACP (Algebra of Communicating Processes [15], overviewed in [11, 35]), observation equivalence is named $\tau$-bisimulation equivalence [16], and abstraction boils down to renaming actions into the silent step (or action) $\tau$, the occurrences of which then may be eliminated according to certain axioms. Abstraction is a prominent feature in process algebra, serving both verification styles and expressive power.

A popular and relatively new semantics that deals with abstraction, proposed by van Glabbeek and Weijland in [43], is branching bisimulation equivalence (see also [44]). Branching bisimulation equivalence is a refinement of semantics such as observation equivalence, delay bisimulation equivalence [69] and $\eta$-bisimulation equivalence [8], and can be considered an improvement of these
because it fully respects the branching structure of processes. In the words of [44]: “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, \( a \cdot \tau \cdot x = a \cdot x \) in a setting with action prefixing \( a \cdot \cdot \cdot \), such as CCS [68]) is claimed to be at the very heart of abstraction (see [44]). This axiom expresses that the observational contents of the silent step \( \tau \) 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 [41, 44]; we return to this point in Section 11).

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

- Internal activity, that is, the performance of \( \tau \)-steps, may be compressed, but not completely discarded.
- Operators that act on the local structure of a process, such as the \textit{priority operator}, are compatible with this semantics and do not require any special treatment of \( \tau \).

Our bisimulation equivalence is called “orthogonal” because it respects the dichotomy between \textit{concrete processes} [10, 42], that is, processes in which no internal actions occur, and those that contain \( \tau \)-steps: a process without \( \tau \)-steps cannot be equivalent to one with \( \tau \)-steps. As a consequence, orthogonal bisimilarity is less abstract than the equivalences discussed above. Below we elaborate on the two characteristics mentioned.

Let \textit{compression} stand for the reduction of finitary internal activity (characterized by \( \tau \)-steps) to a single \( \tau \)-step. Compression is valid in orthogonal bisimilarity, and after compression, the presence of a \( \tau \)-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 \( \tau \tau \tau = \tau \tau \) is sound). Typically, in orthogonal bisimilarity one may abstract from the \textit{structure} of finitary internal activity, but not from its \textit{presence}. This is a major difference
with branching bisimulation equivalence and the coarser (larger, more identifying) semantics mentioned above.

The priority operator $\theta$ was introduced in [5]. 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 $\theta$ and abstraction are difficult to combine, and a modular approach was advocated for using both in a process algebra verification: first eliminate all occurrences of the priority operator, and then apply abstraction 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,\(^1\) is an immediate consequence of the axiom $\forall \tau = x$. The main cause for this problem is that on the term level $\tau$ can hide alternatives, so that $\forall \tau y$ can be different from $\forall y$ 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 $\forall (a \parallel b\tau c)$, where $a \parallel b\tau c$ represents $a$ in parallel with $b$ followed by $\tau$ followed by $c$, defines a behavior in which the action $a$ may be executed before $c$, a situation that cannot occur in $\forall (a \parallel bc)$. This shows that without special measures, the priority operator is not compatible with the axiom $\forall \tau = x$. However, orthogonal bisimilarity is a congruence for the priority operator (even in the case that $\tau$ has a priority).

We now consider the case of divergence, that is, the occurrence of an infinite $\tau$-path. In branching bisimulation equivalence, a $\tau$-loop may be discarded in case there is an alternative available, which can be explained as a feature: often $\tau$-loops result from the abstraction of the occurrence and recovery of an undesirable event, for example the corruption and retransmission of a data-package in a communication protocol. Discarding such a loop corresponds with the assumption that it will not be chosen infinitely often (and, following the example, with the assumption that the occurrence and recovery of an undesirable event may 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 $\tau$-loops can always be discarded, this is not the case in orthogonal bisimilarity. According to the first characteristic above, a $\tau$-loop may be discarded only if one of its exits starts with an initial $\tau$-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 [44]).

\(^1\)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.
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 $ACP_{\tau}^\text{orth}$ 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 [80] in orthogonal bisimulation equivalence. The paper ends with some remarks and conclusions in Section 11.

Note. In earlier work [83, 84], orthogonal bisimilarity was defined using a constant $\iota$ instead of $\tau$. We now consider this use of the symbol $\iota$ 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 ($\tau$) 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 $s \xrightarrow{a} \text{ if } s$ has an outgoing $a$-transition.

A transition system with termination is a transition system together with a predicate $\checkmark$ on its states; a state $s$ with $\checkmark s$ 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 $\checkmark$ to denote the single termination state.

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

We are now ready to define orthogonal bisimulation equivalence of states.
Definition 2.1. Consider a transition system \((S, L_T, T)\) with termination. A binary relation \(R\) on \(S\) is an orthogonal bisimulation, if it is symmetric, and whenever \(s \mathrel{R} r\), then

(i) if \(s \mathrel{\sqrt{s}}\), then \(r \mathrel{\sqrt{r}}\);

(ii) if \(s \xrightarrow{a} s'\) for some \(s'\) and \(a \neq \tau\), then \(r \xrightarrow{a} r'\) for some \(r'\) with \(s' \mathrel{R} r'\); and

(iii) 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 \(n \geq 0\) such that \(s' \mathrel{R} r_n\) and \(s \mathrel{R} r_i\) for all \(i < n\).

States \(s\) and \(r\) are orthogonally bisimilar, notation \(s \mathrel{\leftrightarrow_0} r\), if they are related by some orthogonal bisimulation.

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

\[
\begin{array}{c}
\xrightarrow{a} \quad \xrightarrow{\tau} \quad \xrightarrow{a} \quad \xrightarrow{\tau} \\
S_0 \quad S_1 \quad S_2
\end{array}
\]

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 \(\tau\).

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 \(\tau\)-path of length \(n\), then this path is matched by a \(\tau\)-path in the other state that consists of \(n\) consecutive \(\tau\)-paths, where each of its intermediate states can be related to an appropriate state in the original path:

Lemma 2.1. If \(R\) is an orthogonal bisimulation with \(s \mathrel{R} r\), and there is a path \(s_0 \ldots s_n\) in \(\tau\text{-paths}(s)\), for some \(n \geq 0\), then there is, for every \(i \leq n\), an \(m_i \geq 0\), such that \(r\) has a \(\tau\)-path with \(r_0^0 = r\) and \(m_n = 0\) and

(i) for all \(i < n\), \(r_i^0 \ldots r_i^{m_i} \in \tau\text{-paths}(r_i^0)\) and \(r_i^{m_i} = r_i^{0}\),

(ii) for all \(i \leq n\), if \(j < m_i\) or \(j = 0\), then \(r_i^j \mathrel{R} s_i\).

Proof. Straightforward by induction on \(n\).

Theorem 2.2. 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 bimulations $R'$ and $R''$, we show that the symmetric relation

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

is an orthogonal bisimulation, and thereby that $s_1 \triangleq_0 s_3$. Take any pair $(s, r)$ from $R$. By definition of $R$, there is a state $t$ such that either $s R' t$ and $t R'' r$, or $r R' t$ and $t R'' s$. 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 $r \xrightarrow{\tau} r'$. Since $s R' t$, the state $t$ matches the $\tau$-step to $s'$ in zero or more transitions: for some $n \geq 0$, there is a sequence $t_0 \ldots t_n$ in $\tau\text{-paths}(t)$ such that $s R' t_i$ for all $i < n$ and $s' R' t_n$. The proof is finished using Lemma 2.1.

**Strong Bisimulation.** We compare orthogonal bisimulation equivalence with strong bisimulation equivalence [74] 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 $s R r$, then

(i) if $\sqrt{s}$, then $\sqrt{r}$;

(ii) if $s \xrightarrow{a} s'$ for some $a$ and $s'$, then $r \xrightarrow{a} r'$ for some $r'$ with $s' R r'$.

States $s$ and $r$ are strongly bisimilar, notation $s \trianglelefteq r$, 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 $\tau$-transition is inert, if its source and target are orthogonally bisimilar. A state is compact, if it has no inert outgoing $\tau$-transitions, and all its successors are compact.

**Lemma 2.3.** If $s$ and $r$ are compact, then $s \triangleq_0 r$ implies $s \trianglelefteq r$.

Proof. We show that the relation

$$R = \{(s, t) \mid s \triangleq_0 r \text{ and } s, r \text{ compact}\}$$

is a strong bisimulation. Clearly, it is symmetric. Take states $s$ and $r$ with $s R r$. By definition of $R$ there exists an orthogonal bisimulation $R'$ that relates $s$ and $r$. We distinguish the following cases. First, if $\sqrt{s}$ then it must be that $\sqrt{r}$, 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 $\tau$-step to $s'$, then we know that there is a path $r_0 \ldots r_n$ in $\tau\text{-paths}(r)$ such that $s R' r_i$ for
2. Definition of the Equivalence

\( i < n \) and \( s'R'r_n \). It suffices to show that it must be that \( n = 1 \). If \( n = 0 \), then \( s' \Leftrightarrow_o r_0 \). Since \( s \Leftrightarrow_o r_0 \) 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. \( \square \)

Branching Bisimulation. We now turn to branching bisimilarity [44]. This equivalence is the finest (smallest, least identifying) of the process equivalences described in [39]. Orthogonal bisimilarity is finer than branching bisimilarity, and hence finer than the equivalences in [39].

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

(i) if \( \sqrt{s} \), then there is an \( r' \) with \( r \Rightarrow r' \) and \( \sqrt{r'} \);

(ii) 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 \( s \Leftrightarrow_b r \), if they are related by some branching bisimulation.

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

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 \( \tau \) 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, 6, and 12).

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

(i) if \( s \xrightarrow{a} s' \) for some \( s' \), then \( r \xrightarrow{a} r' \) for some \( r' \) with \( s'Rr' \);

(ii) 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 \( s \Leftrightarrow_{ro} r \) (\( s \Leftrightarrow_{rb} r \)), if there is an orthogonal (branching) bisimulation that is rooted between \( s \) and \( r \).

Using Theorem 2.2 it is straightforward to verify that rooted orthogonal bisimilarity is an equivalence relation.

Proposition 2.4. \( \Leftrightarrow \subseteq \Leftrightarrow_o \subseteq \Leftrightarrow_b \) and \( \Leftrightarrow \subseteq \Leftrightarrow_{ro} \subseteq \Leftrightarrow_{rb} \), 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.

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

The following lemma is an easy corollary of Lemma 2.3:

**Lemma 2.5.** If all successors of $s$, $r$ are compact, then $s \equiv_{\tau_0} r$ implies $s \equiv r$.

**Divergence.** A state $s$ has $\tau$-divergence if there is an infinite $\tau$-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 $\tau$-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 $\tau$-divergence and $s_1$ has not.

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

However, infinite $\tau$-traces do not always collapse under (rooted) orthogonal bisimilarity, an example being

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

where $s_0 \not\equiv_0 s_1$ and $s_0 \not\equiv_0 s_2$. This implies that $\tau$-divergence is a context-dependent phenomenon, and that from a semantic point of view, orthogonal bisimilarity is not optimal. For this reason we define a non-collapsing version for which $\tau$-divergence is an invariant: an orthogonal bisimulation $R$ is **divergence sensitive**, if whenever $s R r$ and $s$ has $\tau$-divergence, then $r$ has $\tau$-divergence. States $s$ and $r$ are divergence sensitive orthogonally bisimilar, notation $s \equiv_{dso} r$, if they are related by a divergence sensitive orthogonal bisimulation. States $s$ and $r$ are **rooted** divergence sensitive orthogonally bisimilar, notation $s \equiv_{rdso} r$, 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. The primitives of the logic may be
considered rather standard: transition labels act as existential modal operators, and it has negation, conjunction, and an until operator. Furthermore, there is a termination predicate and a $\tau$-enabledness predicate.

Given a fixed set $L$ of labels not containing $\tau$, we define the set $\mathcal{L}$ of formulas $\phi$ by the following grammar:

$$\phi ::= \top | \tau | a\phi | \neg\phi | \phi \land \phi | \phi \lor \phi,$$

where $a$ ranges over $L$. We abbreviate the formula $\tau \land \neg\tau$ as $\bot$. Furthermore, we write $T$ for $\neg\bot$, $a$ for $a\top$, and $F\phi$ for $\top \lor \phi$.

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

- $s \models \top$, if $s \models \tau$,
- $s \models a\phi$, if $s \xrightarrow{a} s'$ and $s' \models \phi$ for some $s'$,
- $s \models \neg\phi$, if not $s \models \phi$,
- $s \models \phi \land \psi$, if $s \models \phi$ and $s \models \psi$, and
- $s \models \phi \lor \psi$, if, for some $n \geq 0$, there is a $s_0 \ldots s_n \in \tau$-paths$(s)$ such that $s_i \models \phi$ for all $i < n$ and $s_n \models \psi$.

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

Consider for example the transition system in Figure 1. Every state in this picture satisfies the formula (F$b$) $\lor a$. Also, observe that states $s_0$ and $s_1$ can reach the same states by $\tau$-steps, namely $s_2$ and $s_3$. But while $s_1$ satisfies ($\neg b$) $\lor a$, 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 $\lor$ only in its restricted form as the future operator $F$.

**Theorem 3.1.** Consider a transition system over $L_T$ with termination. For all states $s$ and $r$, $s \equiv_\tau r$ implies $s \sim r$.

**Proof.** By induction on the structure of formulas (using Lemma 2.1). 

In the other direction, the characterization is less general: we have to restrict to transition systems that are finitely branching and $\tau$-path-image-finite. A transition system is finitely branching in label $a$, if all states have finitely many
a-successors. A transition system is \(\tau\)-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 3.2.** If \(R\) is an orthogonal bisimulation with \(sRr\) and \(s \xrightarrow{\tau} s'\), then there is a path \(r_0 \ldots r_n \in \tau\text{-paths}(r)\) with \(n \geq 0\) such that \(sRr_i\) for all \(i < n\) and \(s'Rr_n\), and \(r_i \neq r_j\) for all distinct \(i, j \leq n\).

**Theorem 3.3.** Consider a transition system over \(L_\tau\) with termination that is \(\tau\)-path-image-finite and finitely branching in every label. For all states \(s\) and \(r\), \(s \sim r\) implies \(s \equiv_0 r\).

**Proof.** We show that \(\sim\) is an orthogonal bisimulation. Take any \(s, r\) with \(s \sim r\). We find directly that \(\sqrt{s}\) if and only if \(\sqrt{r}\). 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 \(s \models \neg a\), also \(r \models \neg a\). So, using that \(r\) is finitely branching in \(a\), for some \(n \geq 0\), \(r\) has \(a\)-successors \(r_0, \ldots, r_n\). We have to show that, for some \(i \leq n\), \(s' \sim r_i\). Suppose that, for all \(i \leq n\), \(s' \not\sim r_i\). Then there is, for every \(i \leq n\), a formula \(\phi_i\), such that \(s' \models \phi_i\) and \(r_i \not\models \phi_i\). Let \(\phi = a(\phi_0 \land \cdots \land \phi_n)\). 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 \(s' \sim r\) 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 \(\tau\)-step to \(s'\) appropriately. Suppose, to the contrary, that it cannot (\(\dagger\)), that is, that there is no \(r_0 \ldots r_n \in \tau\text{-paths}(r)\) with \(n \geq 0\) and \(s \sim r_i\), for all \(i < n\), and \(s' \sim r_n\) and, for all \(i, j \leq n\), if \(i \neq j\) then \(r_i \neq r_j\). This last condition is justified by Lemma 3.2.

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

\[
 n \geq 0, \forall i \leq n(s \sim r_i), \text{ and } \forall i, j \leq n(r_i \neq r_j \lor i = j).
\]

The set \(C\) is finite because \(r\) is \(\tau\)-path-image finite. It is nonempty because \(r \in C\). By assumption (\(\dagger\)), we see that, for all \(r_0 \ldots r' \in C\), there is no \(r''\) such that \(r_0 \ldots r' \xrightarrow{\tau} r''\) and \(s' \sim r''\).

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

\[
 C' = \{ r_0 \ldots r'' \mid r_0 \ldots r' \in C, r_0 \ldots r' \xrightarrow{\tau} r'', r'' \not\sim s \}
\]

The set \(C'\) is finite because \(C\) is finite and the transition system is finitely branching in \(\tau\).

Let \(\chi\) be a formula such that \(s' \models \chi\) and \(s \not\models \chi\). Such a formula \(\chi\) exists, because \(s \not\sim s'\). It is straightforward to check that \(C'\) must be nonempty, since if it were empty then \(r \not\models F\chi\), whereas \(s \models F\chi\).
So write $C' = \{\rho_0, \ldots, \rho_k\}$ for some $k \geq 0$. For all $\rho_i = r \ldots r_i \in C'$ we have that $r_i \not\sim s$ and $r_i \not\sim s'$, and hence that there are formulas $\phi_i, \psi_i$ such that $s \models \phi_i, s' \models \psi_i, r_i \not\models \phi_i$ and $r_i \not\models \psi_i$. Let $\phi = \phi_0 \land \cdots \land \phi_k$ and $\psi = \psi_0 \land \cdots \land \psi_k$. Then $s \models \phi, s' \models \psi$ and for all $i \leq k, r_i \not\models \phi$ and $r_i \not\models \psi$.

We see directly that $s \models \phi U (\psi \land \chi)$. We show that $r \not\models \phi U (\psi \land \chi)$, which contradicts the assumption that $s \sim r$. Suppose that $r \models \phi U (\psi \land \chi)$, that is, that there is a $\tau$-path $r_0 \ldots r_n$ with $r = r_0$ and $n \geq 0$, such that $r_n \models \psi \land \chi$ and $r_i \models \phi$ for all $i < n$ ($\dagger$). 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 ($\dagger$).
- $r_n \not\sim s$, 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. \(\square\)

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 $\tau$-divergence. The proofs for the corresponding counterparts of Theorem 3.1 and Theorem 3.3 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($A, \gamma$) [15] 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 $\gamma$ on $A$. The function $\gamma$ defines synchronous communication between actions. We write $a, b$ for arbitrary actions.

The signature has a constant $d \not\in A$ (deadlock). Furthermore, the signature has binary operators $+$ (alternative composition), $\cdot$ (sequential composition), $|$ (parallel composition, merge), $\parallel$ (left merge), and $|$ (communication merge). It has a unary renaming operator $\partial_H$ (encapsulation) for every set $H \subseteq A$. We write $A_\delta$ 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($A, \gamma$) are BPA($A$) (Basic Process Algebra), which consists of the axioms A1–A5, and has sequential and alternative composition as operators, and BPA$_d(A)$, the extension of BPA($A$) with the deadlock process, axiomatized by the axioms A6 and A7. If $E$ is any of these axiom systems, then we write $\text{CT}(E)$ for its set of closed terms.
Orthogonal Bisimulation Equivalence

**Table 1.** The axioms of ACP(A, γ); a, b ∈ Aδ and H ⊆ A.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(A1)</td>
<td>x + y = y + x</td>
</tr>
<tr>
<td>(A2)</td>
<td>x + (y + z) = (x + y) + z</td>
</tr>
<tr>
<td>(A3)</td>
<td>x + x = x</td>
</tr>
<tr>
<td>(A4)</td>
<td>(x + y)z = xz + yz</td>
</tr>
<tr>
<td>(A5)</td>
<td>(xy)z = x(yz)</td>
</tr>
<tr>
<td>(A6)</td>
<td>x + δ = x</td>
</tr>
<tr>
<td>(A7)</td>
<td>δx = δ</td>
</tr>
<tr>
<td>(CM1)</td>
<td>x</td>
</tr>
<tr>
<td>(CM2)</td>
<td>a</td>
</tr>
<tr>
<td>(CM3)</td>
<td>ax</td>
</tr>
<tr>
<td>(CM4)</td>
<td>(x + y)</td>
</tr>
<tr>
<td>(CM5)</td>
<td>ax</td>
</tr>
<tr>
<td>(CM6)</td>
<td>a</td>
</tr>
<tr>
<td>(CM7)</td>
<td>ax</td>
</tr>
<tr>
<td>(CM8)</td>
<td>(x + y)</td>
</tr>
<tr>
<td>(CM9)</td>
<td>x</td>
</tr>
<tr>
<td>(CF1)</td>
<td>a</td>
</tr>
<tr>
<td>(CF2)</td>
<td>a</td>
</tr>
<tr>
<td>(D1)</td>
<td>∂H(a) = a</td>
</tr>
<tr>
<td>(D2)</td>
<td>∂H(a) = δ</td>
</tr>
<tr>
<td>(D3)</td>
<td>∂H(x + y) = ∂H(x) + ∂H(y)</td>
</tr>
<tr>
<td>(D4)</td>
<td>∂H(xy) = ∂H(x)∂H(y)</td>
</tr>
</tbody>
</table>

**Table 2.** Compression axioms; a ∈ Aδx and I ⊆ A.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(O1)</td>
<td>xτττ = xτ</td>
</tr>
<tr>
<td>(O2)</td>
<td>xτ(y + z) = x(y + z)</td>
</tr>
<tr>
<td>(O3)</td>
<td>x(τ(y + z) + z) = x(y + z)</td>
</tr>
<tr>
<td>(TI1)</td>
<td>τI(a) = a</td>
</tr>
<tr>
<td>(TI2)</td>
<td>τI(a) = τ</td>
</tr>
<tr>
<td>(TI3)</td>
<td>τI(x + y) = τI(x) + τI(y)</td>
</tr>
<tr>
<td>(TI4)</td>
<td>τI(xy) = τI(x)τI(y)</td>
</tr>
</tbody>
</table>

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

(CT(E) ∪ {⊥}, A, T),

where ⊥ 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 ⊥ has no outgoing transitions; hence, the transition system is pure (has pure
4. Process Algebra

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, [35].

We proceed now to extend these axiom systems with the constant $\tau$ for the silent step and with axioms characterizing orthogonal bisimulation equivalence. The signature for the axiom system $ACP_{\tau}^{\text{orth}}(A, \gamma)$ is obtained by extending the signature of $ACP(A, \gamma)$ with the fresh constant $\tau$ and with a unary renaming operator $\tau_I$ 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 $\tau$. In the operational semantics, we take $A_\tau$ as the set of transition labels; the silent action is simply executed like the other actions.
Orthogonal Bisimulation Equivalence

**Table 4.** Branching bisimulation axioms.

<table>
<thead>
<tr>
<th>(B1)</th>
<th>$x \tau = x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(B2)</td>
<td>$x(\tau(y + z) + z) = x(y + z)$</td>
</tr>
</tbody>
</table>

The subsystems $\text{BPA}_\tau^\text{orth}(A)$ and $\text{BPA}_\delta^\text{orth}(A)$ are the extensions of $\text{BPA}(A)$ and $\text{BPA}_\delta(A)$ with $\tau$ 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.

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

**Proof.** See the appendix (Section 12).

We end this section with two separate remarks.

First, observe that a closed $\text{BPA}_\tau^\text{orth}(A)$ term $t$ that is built from $\tau$'s only, that is, $a = \tau$ for all subterms $a \in A_\tau$ of $t$, is derivably equal to exactly one of $\tau$, $\tau \tau$ and $\tau + \tau$. This proposition can be proved straightforwardly by induction on the structure of terms; for example, we derive using axioms A3 and O3:

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

Second, rooted branching bisimilarity is axiomatized by the 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

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

### 5. Completeness

We prove completeness of the axiom system $\text{BPA}_\delta^\text{orth}(A)$, that is, we prove that any two rooted orthogonally bisimilar closed terms are derivably equal. The proof is based on Lemma 2.5 and the completeness of $\text{BPA}_\delta(A)$ 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 $\text{BPA}_\tau^\text{orth}(A)$ (without deadlock) can be proved similarly; this proof is omitted. We state that $\text{BPA}_\delta^\text{orth}(A)$ is a conservative extension of $\text{BPA}_\tau^\text{orth}(A)$.

The completeness of $\text{ACP}_\tau^\text{orth}(A, \gamma)$ follows as an easy corollary from the completeness of $\text{BPA}_\delta^\text{orth}(A)$, since the operations for parallelism can easily be eliminated from terms: every closed $\text{ACP}_\tau^\text{orth}(A, \gamma)$ term is derivably equal to a closed $\text{BPA}_\delta^\text{orth}(A)$ term. This elimination result is standard for $\text{ACP}$, and carries over to its orthogonal variant directly, as the special status of $\tau$ as an...
action does not interfere with the elimination. We state that $\text{ACP}_{\tau}^{\text{orth}}(A, \gamma)$ is a conservative extension of $\text{BPA}_{\delta \tau}^{\text{orth}}(A)$.

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

**Definition 5.1.** Let $A$ be the set of action symbols. Then:

(i) The elements of $A_{\delta \tau}$ are basic terms.
(ii) If $a \in A_{\tau}$, and $t$ is a basic term, then $a \cdot t$ is a basic term.
(iii) 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 5.1.** Every closed $\text{BPA}_{\delta \tau}^{\text{orth}}(A)$ term is derivably equal to a basic term.

*Proof.* Standard and thus omitted. \(\square\)

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

*Proof.* Using induction on $|I|$ and axioms O1 and O2. \(\square\)

**Lemma 5.3.** If $t = \sum_{i \in I} \tau_i t_i + t'$ for some nonempty finite index set $I$, with $t_i$ compact and $t \not\equiv_0 t_i$ 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 $t_i \not\equiv_0 t_j$. Since $t_i, t_j$ are compact, we have by Lemma 2.3 that $t_i \not\equiv t_j$. By the completeness of $\text{BPA}_{\delta}$ with respect to strong bisimilarity we get $t_i = t_j$. The required identity follows by axiom A3. \(\square\)

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

*Proof.* Take any closed term $t$. By Lemma 5.1 we may assume that $t$ is a basic term. We apply induction on the structure of $t$. If $t \equiv \delta$, then it has no successors. If $t \in A_{\tau}$, then its only successor is $\sqrt{\cdot}$, 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 $\sum_{i \in I} \tau u_i + uc$, where

$$uc = \sum_{j \in J} a_j u_j + \sum_{k \in K} a_k,$$
Orthogonal Bisimulation Equivalence

$u \leftrightarrow_0 u_i$ for all $i$ in $I$; $a_j \in A_\tau$ and $u \not\leftrightarrow_0 u_j$ 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 5.3, we find $u = \tau u_i + u^c$.

We know that $u_i$ is compact and that $u \leftrightarrow_0 u_i$. From these two facts, it is straightforward to verify that $u_i$ must be a summation consisting of the following summands.

(i) 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 $K$.

(ii) For every $j$ in $J$, one or more summands $a_j u_j'$ with $u_j \not\leftrightarrow_0 u_j'$. By Lemma 2.3 and the completeness of $\text{BPA}_5$, 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 $J$.

(iii) For every $l$ in some finite index set $L$, a summand $\tau u_l$, with $u_l \not\leftrightarrow_0 u_l$. We assume that $L$ is nonempty; if it is not, then infer from $u \not\leftrightarrow_0 u_i$ and the fact that $u$ has a $\tau$-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 $\tau u_l$.

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

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

**Theorem 5.5.** The system $\text{BPA}_5^{\text{orth}}(A)$ 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 5.4 we may assume that all successors of these terms are compact. By Lemma 2.5 we have that they are strongly bisimilar. Derivability follows from the fact that $\text{BPA}_5(A)$ is complete with respect to strong bisimilarity.

**Corollary 5.6.** The system $\text{ACP}_5^{\text{orth}}(A, \gamma)$ is complete with respect to rooted orthogonal bisimilarity.

6. Priorities

We extend the axiom system $\text{ACP}_5^{\text{orth}}(A, \gamma)$ with the priority operator $\theta$, introduced in $\text{ACP}$ in [5]. Parameter of this operator is a partial ordering $\leq$ on the set $A_\tau$ of actions (we write $a < b$ or $b > a$ if $a \leq b$ and $a \neq b$). 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 $\theta(b + c) = b + c$. 


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 $<$. A process $x < y$ behaves 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:

$$\theta(ax + by) = \theta(ax) < by + \theta(by) < ax = (a < b) \cdot \theta(x) + (b < a) \cdot \theta(y) = a \cdot \theta(x) + \delta = a \cdot \theta(x).$$

For another example, let the priority ordering by 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 [39]. Observe that none of these equivalences identifies $\theta(t) = a(\tau b + c)$ and $\theta(u) = ab$. We conclude that, in the setting with $\tau$, the priority operator is not a congruence for the abstract process equivalences in [39]. 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 $\tau$ 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 12 it is proved that rooted orthogonal bisimilarity is a congruence with respect to the priority operator in the setting with $\tau$. We state without proof that the priority axioms are sound and that $A P C_{\tau\theta}^{orth}(A, \gamma)$ is a conservative extension of $A P C_{\tau}^{orth}(A, \gamma)$. 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 in [29], where the unless operator is equipped with a “look-ahead” facility for $\tau$-steps. Both these approaches are not fully general, in the sense...
that they do not admit that \( \tau \) (freely) enters the priority ordering. Although it may in some cases be questionable whether \( \tau \) 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 \( \tau \). Then we have that \( \tau_I \) and \( \theta \) 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 \( z = aaaz \) (and many more). Recently, a different approach to the specification of such behaviors attracted attention, namely the use of recursion operators [12, 19]. 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 [36] it is shown that \( BPA^*(A) \) axiomatizes bisimilarity over that signature. The system \( ACP^*(A, \gamma) \) is defined by adding the axioms BKS1–BKS4. In the setting with \( \tau \) and the binary Kleene star, the system \( BPA_{\tau}^{*\text{orth}} \) is defined by extending \( BPA^*(A) \) with the axioms O1–O3 (see Table 2) and the axioms O4 and O5

<table>
<thead>
<tr>
<th>TABLE 6. Priority axioms; ( a, b \in A_{\delta\tau} ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P1) ( a \triangleleft b = a ) if ( a \neq b )</td>
</tr>
<tr>
<td>(P2) ( a \triangleleft b = \delta ) if ( a &lt; b )</td>
</tr>
<tr>
<td>(P3) ( x \triangleleft yz = x \triangleleft y )</td>
</tr>
<tr>
<td>(P4) ( x \triangleleft (y + z) = (x \triangleleft y) \triangleleft z )</td>
</tr>
<tr>
<td>(P5) ( xy \triangleleft z = (x \triangleleft z)y )</td>
</tr>
<tr>
<td>(P6) ( (x + y) \triangleleft z = x \triangleleft z + y \triangleleft z )</td>
</tr>
<tr>
<td>(TH1) ( \theta(a) = a )</td>
</tr>
<tr>
<td>(TH2) ( \theta(xy) = \theta(x)\theta(y) )</td>
</tr>
<tr>
<td>(TH3) ( \theta(x + y) = \theta(x) &lt; y + \theta(y) &lt; x )</td>
</tr>
</tbody>
</table>
7. Recursion Operators and Fairness

### Table 7. The binary Kleene star axioms.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(BKS1)</td>
<td>( x^*y = x(x^*y) + y )</td>
</tr>
<tr>
<td>(BKS2)</td>
<td>( x^*(yz) = (x^*y)z )</td>
</tr>
<tr>
<td>(BKS3)</td>
<td>( (x + y)^<em>z = x^</em>(y((x + y)^*z) + z) )</td>
</tr>
<tr>
<td>(BKS4)</td>
<td>( \omega_H(x^<em>y) = \omega_H(x)^</em>\omega_H(y) )</td>
</tr>
<tr>
<td>(BKS5)</td>
<td>( \tau_I(x^<em>y) = \tau_I(x)^</em>\tau_I(y) )</td>
</tr>
<tr>
<td>(O4)</td>
<td>( x((\tau \tau)^*y) = x(\tau^*y) ) if ( \tau y = \tau \tau y )</td>
</tr>
<tr>
<td>(O5)</td>
<td>( x((\tau + \tau \tau)^*y) = x((\tau \tau)^*y) )</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Transition</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \rightarrow a )</td>
<td>( x \rightarrow x' )</td>
</tr>
<tr>
<td>( x^*y \rightarrow x^*y )</td>
<td>( x^*y \rightarrow (x^*y) )</td>
</tr>
<tr>
<td>( y \rightarrow a )</td>
<td>( y \rightarrow y' )</td>
</tr>
<tr>
<td>( x^*y \rightarrow x^*y )</td>
<td>( x^*y \rightarrow (x^*y) )</td>
</tr>
<tr>
<td>( x \rightarrow a )</td>
<td>( x \rightarrow x' )</td>
</tr>
<tr>
<td>( x^*y \rightarrow (x^*y)(x^*y) )</td>
<td>( x^*y \rightarrow x'((x^*y)(x^*y)) )</td>
</tr>
</tbody>
</table>

given in Table 7 below. Note that these last two axioms are easily proved valid in orthogonal bisimulation equivalence. Finally, the system \( \text{ACP}_{\text{orth}}^*(A, \gamma) \) is defined by adding all axioms from Table 7 to \( \text{ACP}_{\text{orth}}^*(A, \gamma) \).

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 state processes are definable. This restriction can be relaxed by adding the push-down operator (\( $ \), see [19]), 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 non-regular processes can be defined. A typical example is the term \( R \) given by

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

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 Figure 2. Observe that a register holding value $n$ is modelled by the process

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

In ACP* registers can synchronize with terms representing register machine programs. As an example, let

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

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

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

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

$$\partial_H ((\text{pred}_1' \cdot \text{succ}_2')^* \text{zero}_1' \cdot x \parallel R_1(n) \parallel R_2(m)), $$

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 $\delta$, there are no finite equational axiomatizations of the binary Kleene star operator. Therefore we provide the following adaptation of RSP, the Recursive Specification Principle:

$$(\text{RSP}^*) \quad \text{If } x = yx + z \text{ and } \partial_A (y) = \delta, \text{ then } x = y^* z.$$ 

Here the second condition acts as a guardedness restriction: it excludes terms with an initial $\tau$ 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 [19, 22], but we shall not use it.
Fairness. Due to the character and common use of \( \tau \), one may want to abstract from infinite sequences or loops consisting only of \( \tau \)-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

\[
(FIR^b) \quad \tau(\tau^*x) = \tau x,
\]

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

<table>
<thead>
<tr>
<th>Table 9. Fairness axioms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(OFIR1) ( x(\tau^*(y + \tau z)) = x(y + \tau z) )</td>
</tr>
<tr>
<td>(OFIR2) ( x(\tau^*(y + \tau)) = x(y + \tau) )</td>
</tr>
</tbody>
</table>

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 \( \sqrt{\ } \) not having outgoing transitions, and with at least one other state (different from \( \sqrt{\ } \), see Section 4). Expressing a pure transition system \( T \) up to some behavioral equivalence \( \sim \) in axiom system \( E \) comes down to showing that for each state \( s \) in \( T \) different from \( \sqrt{\ } \) there is a term \( t \) over \( E \) satisfying \( s \sim t \).

In [6], Baeten, Bergstra and Klop proved the following basic expressiveness result: each recursive pure transition system (or 'process graph') can be expressed up to rooted \( \tau \)-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 \( \tau \)'s in the following way: given a
transition system $T = (S, L, T)$, its sequential $\tau$-saturation $T_\tau$ is defined by $(S_\tau, L_\tau, T_\tau)$ where

- $S_\tau = \{s, s_\tau \mid s \in S\}$ (and $s \in S$ implies $s_\tau \not\in S$),
- $L_\tau = L \cup \{\tau\}$,
- $T_\tau = \{s \xrightarrow{a} t_\tau, t_\tau \xrightarrow{\tau} t \mid s \xrightarrow{a} t \in T\}$.

We view binary Kleene star and push-down as a modern alternative to the so-called finite guarded recursive specifications as used in the expressiveness result in [6]. First, we prove in detail that we can express the sequential $\tau$-saturation of any finite pure transition system with labels in $L \subseteq A$ up to rooted divergence sensitive orthogonal bisimulation equivalence in $\text{ACP}^\text{orth*}(A, \gamma)$, 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 $\text{ACP}^\text{orth*}(A, \gamma)$ up to rooted orthogonal bisimulation equivalence, for a suitable, finite set $A$ of actions.

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

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

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

with $\alpha_{j,k}$ and $\beta_j$ finite sums of actions or $\delta$ 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 $X_j \xrightarrow{b} \sqrt\cdot$ there is a summand $b$ in $\beta_j$, and conversely, each summand of $\alpha_{j,k}$ and $\beta_j$ is associated with a transition. If there are no transitions with source $X_j$ and target $X_k$ ($\sqrt\cdot$), then $\alpha_{j,k}$ ($\beta_j$, respectively) equals $\delta$. As a consequence, $T_\tau$ can be characterized by

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

We define process terms that mimic the transitions of $T_\tau$. 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 \( l \)th process to start, and
- \( r_l : (l > 0) \) read instruction to start the \( l \)th process.

Let \( H = \{ r_l, s_l | l = 0, 1, \ldots, n \} \), and, for \( j = 1, \ldots, n \),

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

In the case of \(*\), consider the following process terms:

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

We derive:

\[
\partial_H(F_j G \parallel K) = \partial_H((\sum_{k=1}^{n} \alpha_{j,k} s_k G + \beta_j G) \parallel K)
\]

\[
= \sum_{k=1}^{n} \alpha_{j,k} \cdot \partial_H(s_k G \parallel K) + \beta_j \cdot \partial_H(G \parallel K)
\]

\[
= \sum_{k=1}^{n} \alpha_{j,k} \cdot i \cdot \partial_H(G \parallel s_k K) + \beta_j \cdot i
\]

\[
= \sum_{k=1}^{n} \alpha_{j,k} \cdot i \cdot \partial_H(F_k G \parallel K) + \beta_j \cdot i.
\]

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

\[
X_j \Leftrightarrow_{\text{rdso}} \tau_{[l]} \circ \partial_H(F_j G \parallel K)
\]

in \( \text{TS}(\text{ACP}_\tau^{\text{orth*}}(A, \gamma)) \cup T_\tau \) (with single termination state \( \sqrt{.} \)).

In the case of \$\$, consider process terms

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

Then

\[
X_j \Leftrightarrow_{\text{rdso}} \tau_{[l]} \circ \partial_H(F_j M \parallel N)
\]

for each \( j = 1, \ldots, n \). This can be shown along the same lines, using a denumerable infinity of copies of the transitions of \( T_\tau \): 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, $X_j \leftrightarrow_{rdso} Y_j(l)$ for each state $X_j$ of $T_\tau$ and each value of $l$. So it suffices to show that also

$$\tau_{\{i\}} \circ \partial_H(F_j \, M \ || \ N) \leftrightarrow_{rdso} Y_j(0).$$

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

$$\partial_H(F_j \cdot M^{k+1} \ || \ N^{k+1}) = \sum_{k=1}^n \alpha_{j,k} \cdot \partial_H(s_k M^{k+1} \ || \ N^{k+1}) + \beta_j \cdot \partial_H(M^{k+1} \ || \ N^{k+1})$$

$$= \sum_{k=1}^n \alpha_{j,k} \cdot i \cdot \partial_H(M^{k+1} \ || \ s_k N^{k+2}) + \beta_j \cdot i^{k+1}$$

$$= \sum_{k=1}^n \alpha_{j,k} \cdot i^2 \cdot \partial_H(F_k \, M^{k+2} \ || \ N^{k+2}) + \beta_j \cdot i^{k+1}.$$  

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

$$\tau_{\{i\}} \circ \partial_H(F_j \cdot M^{k+1} \ || \ N^{k+1}) =$$

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

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

The above result shows that each regular process can be defined modulo sequential $\tau$-saturation and rooted divergence sensitive orthogonal bisimilar-\(y\) in $\text{ACP}^{\text{orth}}(A, \gamma)$, provided we adopt (at least) one of $\ast$ and $\$\, and $A$ is sufficiently large (but finite). For non-regular, computable processes (that is, processes that can be characterized by a recursive pure transition system) we have the following expressiveness result: the sequential $\tau$-saturation of a recursive pure transition system with (finite) label set $L \subseteq A$ and bounded fan-out can be expressed in $\text{ACP}^{\text{orths}}_\tau(A, \gamma)$ and $\text{ACP}^{\text{orths}}(A, \gamma)$ up to rooted divergence sensitive orthogonal bisimulation equivalence, provided $A$ is sufficiently large. For example, one can express the sequential $\tau$-saturation of a stack over a finite data type using the approach in [19].

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, [72]) in process algebra (a detailed explanation can be found in [22]). Recall the straightforward representation of registers presented in the previous section. Furthermore, each register machine program has a straightforward representation in $\text{BPA}^*(A)$. It easily follows that each (unary) recursive function $f$ can be 'implemented' in $\text{ACP}^\text{ks}(A, \gamma)$ in the following sense: let $P$ represent in $\text{BPA}^*(A)$ a register machine program that computes $f$ using three registers. Then there is a context $\text{Con}[\_\,\_](n)$ where $n$ refers to a register value such that

$$\text{Con}[P \, x \, \gamma](n) \leftrightarrow \begin{cases} \delta(n) \cdot \text{Con}[x](f(n)) & \text{if } f(n) \text{ is defined}, \\ \tau * \delta & \text{otherwise}. \end{cases}$$
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, \( \text{Con}[\_](n) \) can be extended to \( \text{Con}[\_](n_1, \ldots, n_k) \) for the computation of \( k > 1 \) computable functions in a sequential fashion:

\[
\text{Con}[P_1 \cdots P_k x](n, 0, \ldots, 0) \iff ig'(n) \cdot \text{Con}[x](f_1(n), \ldots, f_k(n))
\]

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

Now let \( T = (S, L, T) \) 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 \( \sqrt{\_} = 0 \)). With the above implementation scheme at hand, it is not hard to express the sequential \( \tau \)-saturation of \( T \) up to rooted divergence sensitive orthogonal bisimilarity in \( \text{ACP}_r^{\text{orths}}(A, \gamma) \). 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 \( 1 \leq k \leq m \), 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 \( 1 \leq j \leq m + 1 \) if \( s = 0 \), and otherwise

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

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

\[
\text{Con}[P x](s, 0, \ldots, 0) \overset{i}{\mapsto} \text{Con}[x](s_1, \ldots, s_{m+1}),
\]

where \( \overset{i}{\mapsto} \) is the transitive closure of \( \overset{i}{\mapsto} \).

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

\[
\text{Con}[M x](s_1, \ldots, s_m, s_{m+1}) \overset{i}{\mapsto} \begin{cases} 
\text{Con}[\delta x](s_1, \ldots, s_m, 0) & \text{if } s_{m+1} = 1, \\
\text{Con}[\sum_{j=1}^{k} a_j F_j x](s_1, \ldots, s_m, 0) & \text{if } s_{m+1} > 1.
\end{cases}
\]

\(^2\)Note: in [22] the number 0 was reserved for the deadlock state. Here the choice to let 0 stand for successful termination allows a more elegant presentation.
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

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

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 $r_{[i]}$ we can express $T_r$ up to rooted divergence sensitive orthogonal bisimilarity: for each $s \in S \subseteq S_r \setminus \{\sqrt{\} \}$ it follows that if $s$ has menu $a_1, \ldots, a_k$ then it has transitions

$$s \xrightarrow{a_j} (s_j) \xrightarrow{r} s_j$$

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

$$s \equiv_{\text{rds}} r_{[i]}(\text{Con}[\sum_{j=1}^{k} a_j F_j PQ](s_1, \ldots, s_k, \ldots, s_m, 0))$$

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

Following the proof of Theorem 8.1 above, it is straightforward how this approach should be adapted to $ACP^\text{orth}(A, \gamma)$ (thus, without $*$, cf. the related results in [19, 22]). The above can be summarized as follows:

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

We note that for each term over $ACP^\text{orth}(A, \gamma)$ or $ACP^\text{orth}(A, \gamma)$, 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 $\tau$-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 \( \sqrt{\cdot} = 0 \)). As an example we shall find a process algebraic expression for the state 9 in the sequential \( \tau \)-saturation of the transition system (1).

\[
9 \xrightarrow{a} 6 \xrightarrow{b} \sqrt{\cdot}
\]

This \( \tau \)-saturation is given below (2).

\[
9 \xrightarrow{a} 6_{\tau} \xrightarrow{\tau} 6 \xrightarrow{b} \sqrt{\tau} \xrightarrow{\tau} \sqrt{\cdot}
\]

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,
- 1 for deadlock,
- 2 for a,
- 3 for b,
- 4 for a, a,
- 5 for a, b,
- 6 for b, 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 \( j \)th 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

\[\text{Con}[Px](s, 0, 0) \mapsto \text{Con}[x](s_1, s_2, s_3).\]
10. Verification of a PAR Protocol

We define the menu interpretation process $M$ partly in a graphical manner in Figure 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 first register and empty all the others. In this case they are given by

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

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 $(MP)^*exit_2 E$. Now, state 9 in (2) is expressed by

$$\tau_{[i]}(\text{Con}[aF_i P Q](6, 0, 0))$$

(see Figure 4), that is,

$$9 \leftrightarrow \tau_{[i]}(\text{Con}[aF_i P Q](6, 0, 0)),$$

as the only difference is in the length of $\tau$-sequences.

10. Verification of a PAR Protocol

We prove the correctness of a Positive Acknowledgement and Retransmission (PAR) protocol [80]. The architecture of the protocol is depicted in Figure 5. We refer to [81] for an earlier verification in process algebra.

![Architecture of the protocol](image)

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

Table 10. Specification.

<table>
<thead>
<tr>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S = (S_0 \cdot S_1) \cdot \delta$</td>
</tr>
<tr>
<td>$S_n = \sum_{d \in D} r(d) \cdot s_0(dn) \cdot S_{dn}$</td>
</tr>
<tr>
<td>$S_{dn} = (\text{time_out} \cdot s_0(dn)) \cdot r_3$</td>
</tr>
<tr>
<td>$K = (\sum_{f \in F} r_0(f) \cdot K_f) \cdot \delta$</td>
</tr>
<tr>
<td>$K_f = i \cdot s_1(f) + i \cdot s_1(\perp) + i$</td>
</tr>
<tr>
<td>$L = (r_2 \cdot (i \cdot s_3 + i)) \cdot \delta$</td>
</tr>
<tr>
<td>$R = (R_0 \cdot R_1) \cdot \delta$</td>
</tr>
<tr>
<td>$R_n = (r_1(\perp) + \sum_{d \in D} r_1(d(1 - n)) \cdot s_2) \cdot \sum_{d \in D} r_1(dn) \cdot s(d) \cdot s_2$</td>
</tr>
</tbody>
</table>

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 [80], 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.

**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 $k \leq 3$, are the send and receive actions over the internal port $k$. We let $\gamma(s_k, r_k) = \gamma(r_k, s_k) = c_k$ for $k = 2, 3$, 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 $\gamma$ undefined otherwise. The action $c_2$ models the passing of an acknowledgement from $R$ to $L$. The action $c_3$ is the passing of an acknowledgement 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

$$
X = \theta(\delta_H(S \parallel K \parallel R \parallel L)).
$$
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 \(\delta\). The derivation of the linearization may be found in Table 12. The linearization is illustrated in Figure 6.

Using RSP\(^*\) we derive from the equations for \(X\) derived in Table 12:

\[
X_d^0 = ((i \cdot c_1(\bot) + i) \cdot \text{time.out} \cdot c_0(d0))^i \cdot c_1(d0) \cdot s(d) \cdot c_2 \cdot X_d^2,
\]

\[
X_d^2 = (i \cdot \text{time.out} \cdot c_0(d0) \cdot \tilde{X}_d^4)^i \cdot c_3 \cdot Y,
\]

\[
X_d^3 = [S_{d0} S_1 S || K_{d0} K || R || L]
\]

\[
X_d^4 = [S_{d0} S_1 S || K_{d0} K || R_1 R || L]
\]

\[
Y = [S_1 S || K || R_1 R || L]
\]
Orthogonal Bisimulation Equivalence

TABLE 12. Linearization.

\[ X = \sum_{d \in D} r(d) \cdot [s_0(d) \cdot S_{d0}S_1 S \ || \ K \ || \ R \ || \ L] \]
\[ = \sum_{d \in D} r(d) \cdot c_0(d) \cdot X_d^0 \]

\[ X_d^0 = i \cdot [S_{d0}S_1 S \ || \ s_1(d) \cdot K \ || \ R \ || \ L] \]
\[ + i \cdot [S_{d0}S_1 S \ || \ s_1(\perp) \cdot K \ || \ R \ || \ L] + i \cdot X_d^1 \]
\[ = i \cdot c_1(d) \cdot [S_{d0}S_1 S \ || \ K \ || \ s(d) \cdot s_2 \cdot R_1 R \ || \ L] \]
\[ + (i \cdot c_1(\perp) + i) \cdot X_d^1 \]
\[ = i \cdot c_1(d) \cdot s(d) \cdot c_2 \cdot X_d^2 + (i \cdot c_1(\perp) + i) \cdot X_d^1 \]

\[ X_d^1 = \text{time.out} \cdot [s_0(d) \cdot S_{d0}S_1 S \ || \ K \ || \ R \ || \ L] \]
\[ = \text{time.out} \cdot c_0(d) \cdot X_d^0 \]

\[ X_d^2 = i \cdot [S_{d0}S_1 S \ || \ K \ || \ R_1 R \ || \ s_3 \cdot L] + i \cdot X_d^3 \]
\[ = i \cdot c_3 \cdot Y + i \cdot X_d^3 \]

\[ X_d^3 = \text{time.out} \cdot [s_0(d) \cdot S_{d0}S_1 S \ || \ K \ || \ R_1 R \ || \ L] \]
\[ = \text{time.out} \cdot c_0(d) \cdot X_d^4 \]

\[ X_d^4 = i \cdot [S_{d0}S_1 S \ || \ s_1(d) \cdot K \ || \ R_1 R \ || \ L] \]
\[ + i \cdot [S_{d0}S_1 S \ || \ s_1(\perp) \cdot K \ || \ R_1 R \ || \ L] + i \cdot X_d^3 \]
\[ = i \cdot c_1(d) \cdot [S_{d0}S_1 S \ || \ K \ || \ s_2 \cdot R_1 R \ || \ L] + (i \cdot c_1(\perp) + i) \cdot X_d^3 \]
\[ = i \cdot c_1(d) \cdot c_2 \cdot X_d^2 + (i \cdot c_1(\perp) + i) \cdot X_d^3 \]

\[ \overline{X}_d^4 = ((i \cdot c_1(\perp) + i) \cdot \text{time.out} \cdot c_0(d0))^* i \cdot c_1(d0) \cdot c_2. \]

In a similar way we can derive

\[ Y = \sum_{d \in E} r(d) \cdot c_0(d1) \cdot Y_d^0, \]

\[ Y_d^0 = ((i \cdot c_1(\perp) + i) \cdot \text{time.out} \cdot c_0(d1))^* i \cdot c_1(d1) \cdot s(d) \cdot c_2 \cdot Y_d^2, \]

\[ Y_d^2 = (i \cdot \text{time.out} \cdot c_0(d1) \cdot \overline{Y}_d^4)^* i \cdot c_3 \cdot \overline{X}, \]

\[ \overline{Y}_d^4 = ((i \cdot c_1(\perp) + i) \cdot \text{time.out} \cdot c_0(d1))^* i \cdot c_1(d1) \cdot c_2. \]

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 $\tau_I$ inwards and compressing $\tau$'s

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

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

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

**Remark 10.1.** In [83], 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

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

### 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 [44]. 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." [44, 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\(^3\), 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 $\alpha \tau x = \alpha x$ (cf. [40, 44]).

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

\(^3\)This notion is formally defined in [41].
Orthogonal Bisimulation Equivalence

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 $\tau$-cycles usually result from the abstraction of the occurrence and 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 $\tau$-cycles happen to start with a $\tau$-step. However, this is not always the case in the daily practice of protocol specification in process algebra (cf. Remark 10.1). For this reason, the divergence sensitive variant of orthogonal bisimulation equivalence is distinguished (which simply never discards $\tau$-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 [41] 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 $\tau$’s are removed, leads to a characterization of orthogonal bisimilarity along the same lines as in [41]. 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, [11]) that the priority operator is not compatible with failure or ready semantics.4

---

4 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 $\theta(P) \not\equiv_F \theta(Q)$. 
12. Appendix: Congruence Proofs

We prove that rooted orthogonal bisimilarity is a congruence relation with respect to all operators of $ACP^{\text{orth}}_{t\theta}$. The proof is both straightforward and elaborate. Consider the transition system of closed $ACP^{\text{orth}}_{t\theta} (A, \gamma)$ terms for some arbitrary $A$ and $\gamma$. 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 $\sqrt{\cdot}$).

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

Consider (for $i = 1, 2$) process terms $t_i$ and $u_i$ such that $t_i \equiv_{ro} u_i$. We prove that $t_1 \diamond t_2 \equiv_{ro} u_1 \diamond u_2$ for all $\diamond \in \{\cdot, ||, [\cdot], [\cdot]_1\}$, and that $\nabla(t_1) \equiv_{ro} \nabla(u_1)$ for all $\nabla \in \{\partial_H, \tau_I, \theta\}$ with arbitrary $I, H \subseteq A$ and priority ordering $<$ on $A_T$. 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.

$$ R' = \left\{ (t \diamond v, u \diamond w), (\nabla(t), \nabla(u)) \mid \begin{array}{l}
(t, u) \in R_1 \setminus \{(\sqrt{\cdot}, \sqrt{\cdot})\}, \\
(v, w) \in R_2 \setminus \{(\sqrt{\cdot}, \sqrt{\cdot})\}, \\
\diamond \in \{\cdot, ||, [\cdot], [\cdot]_1\}, \\
\nabla \in \{\partial_H, \tau_I, \theta\} 
\end{array} \right\} $$

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

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 $t R_1 u$ and $v R_2 w$ by definition of $R'$. Since $R_1$ and $R_2$ are symmetric, we find that $u R_1 t$ and $w R_2 v$. Hence, by definition of $R'$, it follows that $(y, x) \in R$.
- If $x \equiv \nabla(t)$ for some $\nabla$, then use the definition of $R'$ and the symmetry of $R_1$.

Concrete Action Steps. Take any $(x, y) \in R'$ and assume that $x \xrightarrow{a} x'$ for some $a$ and $x'$ with $a \neq \tau$. We have to show that $y \xrightarrow{a} y'$ for some $y'$ with $x' R'y'$. 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 $t R_1 u$ and $v R_2 w$. We see that either $t \xrightarrow{a} \sqrt{\cdot}$ and $x' = v$, or $t \xrightarrow{a} t'$ and $x' = t' \cdot v$. In the first case also $u \xrightarrow{a} \sqrt{\cdot}$ since $t R_1 u$. Hence $y \xrightarrow{a} w$. Since $v R_2 w$, also $v R w$, qed. In the second case, we find that $u \xrightarrow{a} u'$ for some $u'$ with $t' R_1 u'$. Then $y \xrightarrow{a} u' \cdot w$. We get $t' \cdot v Ru' \cdot w$ using the definition of $R'$. 

Orthogonal Bisimulation Equivalence

- If \( x \equiv t \parallel v \), then, by definition of \( R' \), \( y \equiv u \parallel w \) such that \( t R_1 u \) and \( v R_2 w \). We distinguish 8 possibilities:
  
  (i) \( t \xrightarrow{a} \sqrt{\cdot} \) and \( x' = v \). In this case also \( u \xrightarrow{a} \sqrt{\cdot} \), and hence \( y \xrightarrow{a} w \), qed.
  
  (ii) \( t \xrightarrow{a} t' \) and \( x' = t' \parallel v \). In this case \( u \xrightarrow{a} u' \) with \( t' R_1 u' \). Then \( y \xrightarrow{a} u' \parallel w \). Using the definition of \( R' \), we get \( x' R u' \parallel w \).
  
  (iii) \( v \xrightarrow{a} \sqrt{\cdot} \) and \( x' = t \). Like case (i).
  
  (iv) \( v \xrightarrow{a} v' \) and \( x' = t \parallel v' \). Like case (ii).
  
  (v) \( t \xrightarrow{b} \sqrt{\cdot} \) and \( v \xrightarrow{c} \sqrt{\cdot} \) and \( \gamma(b, c) = a \) and \( x' = \sqrt{\cdot} \). In this case, we have \( u \xrightarrow{b} \sqrt{\cdot} \) and \( w \xrightarrow{c} \sqrt{\cdot} \). Hence \( y \xrightarrow{a} \sqrt{\cdot} \). It holds that \( \sqrt{R \sqrt{\cdot}} \), since \( \sqrt{R_1 \sqrt{\cdot}} \) (and \( \sqrt{R_2 \sqrt{\cdot}} \)).
  
  (vi) \( t \xrightarrow{b} t' \) and \( v \xrightarrow{c} \sqrt{\cdot} \) and \( \gamma(b, c) = a \) and \( x' = t' \). Straightforward.
  
  (vii) \( t \xrightarrow{b} \sqrt{\cdot} \) and \( v \xrightarrow{c} v' \) and \( \gamma(b, c) = a \) and \( x' = v' \). Straightforward.
  
  (viii) \( t \xrightarrow{b} t' \) and \( v \xrightarrow{c} v' \) and \( \gamma(b, c) = a \) and \( x' = t' \parallel v' \). Straightforward.
  
- If \( x \equiv t \parallel v \), then, by definition of \( R' \), \( y \equiv u \parallel w \) such that \( t R_1 u \) and \( v R_2 w \). We distinguish 2 possibilities:
  
  (i) \( t \xrightarrow{a} \sqrt{\cdot} \) and \( x' = v \). In this case also \( u \xrightarrow{a} \sqrt{\cdot} \), and hence \( y \xrightarrow{a} w \), qed.
  
  (ii) \( t \xrightarrow{a} t' \) and \( x' = t' \parallel v \). In this case \( u \xrightarrow{a} u' \) with \( t' R_1 u' \). Then \( y \xrightarrow{a} u' \parallel w \). Using the definition of \( R' \), we get \( x' R u' \parallel w \).
  
- If \( x \equiv t | v \), then, by definition of \( R' \), \( y \equiv u | w \) such that \( t R_1 u \) and \( v R_2 w \). We distinguish 4 possibilities:
  
  (i) \( t \xrightarrow{b} \sqrt{\cdot} \) and \( v \xrightarrow{c} \sqrt{\cdot} \) and \( \gamma(b, c) = a \) and \( x' = \sqrt{\cdot} \). In this case we have \( u \xrightarrow{b} \sqrt{\cdot} \) and \( w \xrightarrow{c} \sqrt{\cdot} \). Hence \( y \xrightarrow{a} \sqrt{\cdot} \). It holds that \( \sqrt{R \sqrt{\cdot}} \), since \( \sqrt{R_1 \sqrt{\cdot}} \) (and \( \sqrt{R_2 \sqrt{\cdot}} \)).
  
  (ii) \( t \xrightarrow{b} t' \) and \( v \xrightarrow{c} \sqrt{\cdot} \) and \( \gamma(b, c) = a \) and \( x' = t' \). Straightforward.
  
  (iii) \( t \xrightarrow{b} \sqrt{\cdot} \) and \( v \xrightarrow{c} v' \) and \( \gamma(b, c) = a \) and \( x' = v' \). Straightforward.
  
  (iv) \( t \xrightarrow{b} t' \) and \( v \xrightarrow{c} v' \) and \( \gamma(b, c) = a \) and \( x' = t' \parallel v' \). Straightforward.
  
- If \( x \equiv \partial_H(t) \), then, by definition of \( R' \), \( y \equiv \partial_H(u) \) such that \( t R_1 u \). If \( x' = \sqrt{\cdot} \), then \( t \xrightarrow{a} \sqrt{\cdot} \) and \( a \notin H \). Then also \( u \xrightarrow{a} \sqrt{\cdot} \), and \( y \xrightarrow{a} \sqrt{\cdot} \), qed.
  
If \( x' = \sqrt{\cdot} \), 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 \( t' R_1 u' \). 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.
Appendix: Congruence Proofs

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

  If \( x' = \sqrt{.} \), 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 \( x'R'\theta(u') \).

**Silent Steps.** Take any \((x, y) \in R' \) and assume that \( x \xrightarrow{t} x' \) for some \( x' \). We have to show that \( y \xrightarrow{t} y' \) and that, for some \( n \geq 0 \), there are \( y_k \) such that \( y_0 \cdots y_n \) in \( \tau\text{-}paths(y) \) and \( x'R_1y_n \) and \( x\xrightarrow{R}y_k \) for all \( i < n \).

We make a case distinction on the form of \( x \).

- If \( x \equiv t \cdot v \), then, by definition of \( R' \), \( y \equiv u \cdot w \) such that \( tR_1u \) and \( vR_2w \).

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

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

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

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

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

  (iii) \( v \xrightarrow{\tau} \sqrt{.} \) and \( x' = t \). Like case (i).

  (iv) \( v \xrightarrow{\tau} v' \) and \( x' = t || v' \). Like case (ii).

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

  (i) \( t \xrightarrow{\tau} \sqrt{.} \) 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
orthogonal bisimulation equivalence

\( \tau \text{-paths}(u) \) and \( u_n = \sqrt{\tau} \) and \( t R_1 u_i \) for all \( i < n \). Then also
\( (u_0 \parallel w)(u_1 \parallel w) \cdots (u_{n-1} \parallel w) w \) in \( \tau \text{-paths}(y) \), and, by definition of \( R' \), \( x R' u_i \parallel w \) for all \( 0 < i < n \).

(ii) \( t \xrightarrow{\tau} t' \) and \( x' = t' \parallel v \). In this case \( u \xrightarrow{\tau} \) and hence \( y \xrightarrow{\tau} \).

Furthermore, for some \( n \geq 0 \), there are \( u_i \) such that \( u_0 \cdots u_n \) in \( \tau \text{-paths}(u) \) and \( t' R_i u_n \) and \( t R_1 u_i \) for all \( i < n \). Then also
\( (u_0 \parallel w)(u_1 \parallel w) \cdots (u_n \parallel w) \) in \( \tau \text{-paths}(y) \), and, by definition of \( R' \), \( x R' u_i \parallel w \), for all \( 0 < i < n \), and \( x' R'u_i \parallel w \).

- If \( x = t | v \), then \( x \) cannot have an outgoing \( \tau \)-step.

- If \( x = \partial_H(t) \), then, by definition of \( R' \), \( y \equiv \partial_H(u) \) such that \( t R_1 u \). We see that \( t \xrightarrow{\tau} \) and hence \( u \xrightarrow{\tau} \) and \( y \xrightarrow{\tau} \).

If \( x' = \sqrt{\tau} \), then \( t \xrightarrow{\tau} \sqrt{\tau} \). Furthermore, for some \( n > 0 \), there are \( u_i \) such that \( u_0 \cdots u_n \) in \( \tau \text{-paths}(u) \) and \( u_n = \sqrt{\tau} \) and \( t R_1 u_i \) for all \( i < n \). Then also \( \partial_H(u_0) \cdots \partial_H(u_{n-1}) \sqrt{\tau} \) in \( \tau \text{-paths}(y) \) and \( x R' \partial_H(u_i) \) for all \( i < n \).

- If \( x' \neq \sqrt{\tau} \), then \( x' = \partial_H(t') \), for some \( t' \) with \( t \xrightarrow{\tau} t' \). Furthermore, for some \( n \geq 0 \), there are \( u_i \) such that \( u_0 \cdots u_n \) in \( \tau \text{-paths}(u) \) and \( t' R_i u_n \) and \( t R_1 u_i \) for all \( i < n \). Then also \( \partial_H(u_0) \cdots \partial_H(u_n) \sqrt{\tau} \) in \( \tau \text{-paths}(y) \) and \( x R' \partial_H(u_i) \) and \( x R' \partial_H(u_n) \) for all \( i < n \).

- If \( x = \tau_I(t) \), then, by definition of \( R' \), \( y \equiv \tau_I(u) \) such that \( t R_1 u \). We distinguish 4 possibilities:

(i) \( t \xrightarrow{\tau} \sqrt{\tau} \) and \( x' = \sqrt{\tau} \). We see that \( u \xrightarrow{\tau} \) and hence \( y \xrightarrow{\tau} \). Furthermore, for some \( n > 0 \), there are \( u_i \) such that \( u_0 \cdots u_n \) in \( \tau \text{-paths}(u) \) and \( u_n = \sqrt{\tau} \) and \( t R_1 u_i \) for all \( i < n \). Then also \( \tau_I(u_0) \cdots \tau_I(u_{n-1}) \sqrt{\tau} \) in \( \tau \text{-paths}(y) \) and \( x R' \tau_I(u_i) \) for all \( i < n \).

(ii) \( t \xrightarrow{\tau} t' \) and \( x' = \tau_I(t') \). We see that \( u \xrightarrow{\tau} \) and hence \( y \xrightarrow{\tau} \). Furthermore, for some \( n \geq 0 \), there are \( u_i \) such that \( u_0 \cdots u_n \) in \( \tau \text{-paths}(u) \) and \( t' R_i u_n \) and \( t R_1 u_i \) for all \( i < n \). Then also \( \tau_I(u_0) \cdots \tau_I(u_n) \sqrt{\tau} \) in \( \tau \text{-paths}(y) \) and \( x R' \tau_I(u_i) \) and \( x R' \tau_I(u_n) \) for all \( i < n \).

(iii) \( t \xrightarrow{a} \sqrt{\tau} \) and \( a \in I \) and \( x' = \sqrt{\tau} \). In this case, \( u \xrightarrow{a} \sqrt{\tau} \) and hence \( y \xrightarrow{\tau} \).

(iv) \( t \xrightarrow{a} t' \) and \( a \in I \) and \( x' = \tau_I(t') \). In this case, \( u \xrightarrow{a} u' \) for some \( u' \) with \( t' R_1 u' \). So \( y \xrightarrow{\tau} \tau_I(u') \). By definition of \( R' \), we have \( x R' \tau_I(u') \).

- If \( x = \theta(t) \), then, by definition of \( R' \), \( y \equiv \theta(u) \) such that \( t R_1 u \).

If \( x' = \sqrt{\tau} \), then \( t \xrightarrow{\tau} \sqrt{\tau} \), and there is no \( a > \tau \) with \( t \xrightarrow{a} \). Since \( t R_1 u \), we have that for some \( n > 0 \), there are \( u_i \) such that \( u_0 \cdots u_n \) in \( \tau \text{-paths}(u) \), \( u_n = \sqrt{\tau} \), and \( t R_1 u_i \) for \( i < n \). Since, for \( i < n \), \( t R_1 u_i \), it follows that there is no \( a > \tau \) with \( u_i \xrightarrow{a} \). Hence \( \theta(u_0) \cdots \theta(u_{n-1}) u_n \) in \( \tau \text{-paths}(y) \). We have that \( x R \theta(u_i) \) by definition of \( R' \).
If \( x' \neq \sqrt{.} \), then \( t \xrightarrow{.} t' \) for some \( t' \) with \( x' \equiv \theta(t') \), and there is no \( a > \tau \) with \( t \xrightarrow{a} \). Since \( tR_1u \), we have that for some \( n \geq 0 \), there are \( u_i \) such that \( u_0 \ldots u_n \) in \( \tau\text{-paths}(u) \), \( t'R_1u_n \), and \( tR_1u_i \) for \( i < n \). So \( u \xrightarrow{.} \) and hence \( y \xrightarrow{.} \). Since, for \( i < n \), \( tR_1u_i \), we find that there is no \( a > \tau \) with \( u_i \xrightarrow{a} \). Hence \( \theta(u_0) \ldots \theta(u_n) \) in \( \tau\text{-paths}(y) \). We have that \( xR\theta(u_i) \) and \( x'R\theta(u_n) \) by definition of \( R' \).

**Rootedness.** We have to show that \( R \) is rooted between \( t_1 \circ t_2 \) and \( u_1 \circ u_2 \), and between \( \uparrow(t_1) \) and \( \uparrow(u_1) \), for all \( \circ \in \{ \cdot, ||, \, \}_1 \) and \( \uparrow \in \{ \partial_H, \tau_I, \theta \} \).

- **Sequential composition.** Assume that \( t_1 \cdot t_2 \xrightarrow{.} x \). We distinguish two possibilities:
  (i) \( t_1 \xrightarrow{.} \sqrt{.} \) and \( x = t_2 \). Since \( R_1 \) is rooted between \( t_1 \) and \( u_1 \), it follows that \( u_1 \xrightarrow{.} \sqrt{.} \) and hence \( u_1 \cdot u_2 \xrightarrow{.} u_2 \), qed.
  (ii) \( t_1 \xrightarrow{.} t_1' \) 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 \( u_1 \xrightarrow{.} u_1' \) and \( t_1'R_1u_1' \). We see that \( u_1 \cdot u_2 \xrightarrow{.} u_1' \cdot u_2 \) and \( xR\theta(u_1') \cdot u_2 \), qed.

Silent steps of \( u_1 \cdot u_2 \) are treated symmetrically.

- **Merge.** Assume that \( t_1 \sqcup t_2 \xrightarrow{.} \). We distinguish 4 possibilities:
  (i) \( t_1 \xrightarrow{.} \sqrt{.} \) and \( x = t_2 \). Since \( R_1 \) is rooted between \( t_1 \) and \( u_1 \), it follows that \( u_1 \xrightarrow{.} \sqrt{.} \) and hence \( u_1 \sqcup u_2 \xrightarrow{.} u_2 \), qed.
  (ii) \( t_1 \xrightarrow{.} t_1' \) and \( x = t_1' \sqcup t_2 \). Since \( R_1 \) is rooted between \( t_1 \) and \( u_1 \), there must be some \( u_1' \) with \( u_1 \xrightarrow{.} u_1' \) and \( t_1'R_1u_1' \). We see that \( u_1 \sqcup u_2 \xrightarrow{.} u_1' \sqcup u_2 \) and \( xR\theta(u_1') \sqcup u_2 \), qed.
  (iii) \( t_2 \xrightarrow{a} \sqrt{.} \) and \( x = t_1 \). Like case (i).
  (iv) \( t_2 \xrightarrow{a} t_2' \) and \( x = t_1 \sqcup t_2' \). Like case (ii).

Silent steps of \( u_1 \sqcup u_2 \) are treated symmetrically.

- **Left merge.** Assume that \( t_1 \sqcup t_2 \xrightarrow{.} \). We distinguish 2 possibilities:
  (i) \( t_1 \xrightarrow{.} \sqrt{.} \) and \( x = t_2 \). Since \( R_1 \) is rooted between \( t_1 \) and \( u_1 \), it follows that \( u_1 \xrightarrow{.} \sqrt{.} \) and hence \( u_1 \sqcup u_2 \xrightarrow{.} u_2 \), qed.
  (ii) \( t_1 \xrightarrow{.} t_1' \) and \( x = t_1' \sqcup t_2 \). Since \( R_1 \) is rooted between \( t_1 \) and \( u_1 \), there must be some \( u_1' \) with \( u_1 \xrightarrow{.} u_1' \) and \( t_1'R_1u_1' \). We see that \( u_1 \sqcup u_2 \xrightarrow{.} u_1' \sqcup u_2 \) and \( xR\theta(u_1') \sqcup u_2 \), qed.

Silent steps of \( u_1 \sqcup u_2 \) are treated symmetrically.

- **Communication merge.** No \( \tau \)-steps.

- **Encapsulation.** Assume that \( \partial_H(t_1) \xrightarrow{.} x \). We distinguish 2 possibilities:
  (i) \( t_1 \xrightarrow{.} \sqrt{.} \) and \( x = \sqrt{.} \). Since \( R_1 \) is rooted between \( t_1 \) and \( u_1 \), it follows that \( u_1 \xrightarrow{.} \sqrt{.} \) and hence \( \partial_H(u_1) \xrightarrow{.} \sqrt{.} \), qed.
(ii) $t_1 \xrightarrow{r} t_1'$ and $x = \partial_H(t_1')$. Since $R_1$ is rooted between $t_1$ and $u_1$, there must be some $u_1'$ with $u_1 \xrightarrow{r} u_1'$ and $t_1'R_1u_1'$. We see that $\partial_H(u_1) \xrightarrow{r} \partial_H(u_1')$ and $x R \partial_H(u_1')$, qed.

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

- **Hiding.** Assume that $\tau_I(t_1) \xrightarrow{r} x$. We distinguish 4 possibilities:
  
  (i) $t_1 \xrightarrow{r} \sqrt{\cdot}$ and $x = \sqrt{\cdot}$. Since $R_1$ is rooted between $t_1$ and $u_1$, we have that $u_1 \xrightarrow{r} \sqrt{\cdot}$ and hence $\tau_I(u_1) \xrightarrow{r} \sqrt{\cdot}$, qed.
  
  (ii) $t_1 \xrightarrow{r} t_1'$ and $x = \tau_I(t_1')$. Since $R_1$ is rooted between $t_1$ and $u_1$, there must be some $u_1'$ with $u_1 \xrightarrow{r} u_1'$ and $t_1'R_1u_1'$. We see that $\tau_I(u_1) \xrightarrow{r} \tau_I(u_1')$ and $x R \tau_I(u_1')$, qed.
  
  (iii) $t_1 \xrightarrow{a} \sqrt{\cdot}$ and $a \in I$ and $x = \sqrt{\cdot}$. From $t_1 R_1 u_1$ it follows that $u_1 \xrightarrow{a} \sqrt{\cdot}$ and hence $\tau_I(u_1) \xrightarrow{a} \sqrt{\cdot}$, qed.
  
  (iv) $t_1 \xrightarrow{a} t_1'$ 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 $t_1'R_1u_1'$. We see that $\tau_I(u_1) \xrightarrow{r} \tau_I(u_1')$ and $x R \tau_I(u_1')$, qed.

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

- **Priority.** Assume that $\theta(t_1) \xrightarrow{r} x$. If $x = \sqrt{\cdot}$, then it must be that $t_1 \xrightarrow{r} \sqrt{\cdot}$ and there is no $a > \tau$ with $t_1 \xrightarrow{a} \tau$. Since $t_1 R_1 u_1$, we find that there is no $a > \tau$ with $u_1 \xrightarrow{a} \tau$.

If $x \neq \sqrt{\cdot}$, then it must be that $t_1 \xrightarrow{r} t_1'$ for some $t_1'$ with $x = \theta(t_1')$, and there is no $a > \tau$ with $t_1 \xrightarrow{a} \tau$. Since $t_1 R_1 u_1$, there is no $a > \tau$ with $u_1 \xrightarrow{a} \tau$. Since $R_1$ is rooted between $t_1$ and $u_1$, it must be that $u_1 \xrightarrow{r} u_1'$ for some $u_1'$ with $t_1'R_1u_1'$. By definition of $R'$, we find that $x R \theta(u_1')$.

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

**Alternative Composition.** We prove that $\leftrightarrow_{r_0}$ 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 $t R_i u$. Then also $u_1 + u_2 \xrightarrow{a} u$ and $t R u$. 


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

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