Preferential choice and coordination conditions

Citation for published version (APA):

Document status and date:
Published: 01/01/2005

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 02. Oct. 2020
Abstract. We present a process algebra with conditional expressions of which the conditions concern the enabledness of actions in the context in which a process is placed. With those conditions, it becomes easy to model preferential choices. A preferential choice of a process is a choice whereby certain alternatives are excluded if at least one of the other alternatives is permitted by the context in which the process is placed. Preferential choices are usually modelled rather indirectly using a priority mechanism.

Keywords: coordination conditions, preferential choice, splitting bisimulation, process algebra, Boolean algebras.


1 Introduction

In [11], we started an investigation into the potentialities of conditional expressions in the setting of the algebraic theory about processes known as ACP [8, 7]. The primary intention of the investigation is to find basic ways to increase expressiveness. The main extensions of ACP introduced in [11] are ACP<sub>c</sub> (ACP with conditional expressions), ACP<sub>cs</sub> (ACP<sub>c</sub> with signals) and ACP<sub>cr</sub> (ACP<sub>c</sub> with retrospective conditions).

In the current paper, we proceed with the investigation started in [11]. We present ACP<sub>cc</sub>, a variant of ACP<sub>c</sub> in which the conditions concern the enabledness of actions in the context in which a process is placed. Such conditions are called coordination conditions in this paper because they are primarily intended for coordination of processes that proceed in parallel. Like in ACP<sub>c</sub>, the conditions in ACP<sub>cc</sub> are taken from a Boolean algebra. This looks to be important in the case of ACP<sub>cc</sub> where the context in which a process is placed determines whether a condition holds.

The merit of ACP<sub>cc</sub> is primarily the following. If a process has different alternatives to proceed, some may be preferred in the sense that the others should be
excluded if at least one of the preferred alternatives is permitted by the context in which the process is placed. Such preferential choices are usually modelled rather indirectly using a priority mechanism, see e.g. [17], but can easily be modelled in ACP$^\cc$. See [4] for an extension of ACP with a priority mechanism. Unease about the solution to model preferential choices with a priority mechanism forms part of our motivation to develop ACP$^\cc$.

We present the main models of ACP$^\cc$ as well. They are based on labelled transition systems of which the labels consist of a condition and an action, called conditional transition systems, and a variant of bisimilarity in which a transition of one of the related transition systems may be simulated by a set of transitions of the other transition system, called splitting bisimilarity. The presented models cover finitely branching processes as well as infinitely branching processes.

We also extend ACP$^\cc$ with a preferential choice operator to show that it is easy to give defining equations for a preferential choice operator in the presence of conditional expressions of which the conditions are coordination conditions. This extension of ACP$^\cc$ can be viewed as an application of ACP$^\cc$ that remains entirely within the domain of process algebra.

The preferential choice operator appears to generalize the priority choice operator added to CCS in [16]. The preferential choice operator is defined for all possible operands, whereas syntactic restrictions are imposed on the operands of the priority choice operator from [16]. Because of those restrictions, the priority choice operator violates the basics of an algebraic approach. Therefore, we consider the preferential choice operator a notable improvement on the priority choice operator.

The structure of this paper is as follows. First of all, we introduce PA$^\delta_5$, an important subtheory of ACP$^\cc$ that does not support communication (Section 2). After that, we introduce conditional transition systems and splitting bisimilarity of conditional transition systems (Section 3) and the full splitting bisimulation models of PA$^\delta_5$, the main models of PA$^\delta_5$ (Section 4). Following this, we have a closer look at splitting bisimilarity based on structural operational semantics (Section 5). Next, we extend PA$^\delta_5$ to ACP$^\cc$ (Section 6) and adapt the full splitting bisimulation models of PA$^\delta_5$ to full splitting bisimulation models of ACP$^\cc$ (Section 7). Then, we extend ACP$^\cc$ with guarded recursion (Section 8). Thereupon, we extend ACP$^\cc$ with the preferential choice operator (Section 9) and give an example of its use (Section 10). Finally, we make some remarks about related work and mention some options for future work (Section 11).

Some familiarity with Boolean algebras is desirable. The definitions of all notions concerning Boolean algebras that are used in this paper can, for example, be found in [22].

2 PA$^\delta_5$ with Coordination Conditions

PA$\delta_5$ is a subtheory of ACP that does not support communication (see e.g. [7]). In this section, we present an extension of PA$\delta_5$ with encapsulation, pre-abstraction and guarded commands, called PA$^\alpha_5$. Encapsulation was originally incorporated
in ACP to encapsulate actions of a process from communication with actions
from the outside (see [8]). Pre-abstraction was added to ACP in [5] as a limited
kind of abstraction: the actions from which is abstracted are identified, but
they remain observable as internal actions. Guarded commands are conditional
expressions of the form \( \zeta :\to p \), where \( \zeta \) and \( p \) are expressions representing a
condition and a process, respectively. Guarded commands were added to ACP
for the first time in [6].

In \( \text{PA}_\delta^{cc} \), just as in \( \text{PA}_\delta \) or ACP extended with pre-abstraction, it is assumed
that a fixed but arbitrary finite set of actions \( A \), with \( \delta \not\in A \) and \( t \in A \), has been
given. Action \( t \) is the internal action that replaces all occurrences of actions
from which is abstracted by means of pre-abstraction. Henceforth, we write \( A_\delta \)
for \( A \cup \{ \delta \} \).

If it is permitted by the context in which a process is placed to perform
an action \( a \), we say that \( a \) is enabled in that context. In \( \text{PA}_\delta^{cc} \), we consider
conditions concerning the enabledness of actions. More precisely, conditions are
taken from the quotient algebra of the free Boolean algebra over \( \{ \mathcal{E}_a \mid a \in A \} \) by
the equivalence induced by the equation \( \mathcal{E}_t = \top \). Henceforth, we write \( \mathcal{C} \) for this
algebra and also for the domain of this algebra. The intuition is that \( \mathcal{E}_a \) holds if
action \( a \) is enabled in the context present. Because it is internal, the action \( t \) is
enabled in any context. The elements of \( \mathcal{C} \) are called coordination conditions.

The algebraic theory \( \text{PA}_\delta^{cc} \) has two sorts:

\[ \begin{align*}
&\text{– the sort } P \text{ of processes; } \\
&\text{– the sort } C \text{ of conditions.}
\end{align*} \]

The algebraic theory \( \text{PA}_\delta^{cc} \) has the following constants and operators to build
terms of sort \( P \):

\[ \begin{align*}
&\text{– the deadlock constant } \delta : P; \\
&\text{– for each } a \in A, \text{ the action constant } a : P; \\
&\text{– the binary alternative composition operator } + : P \times P \to P; \\
&\text{– the binary sequential composition operator } \cdot : P \times P \to P; \\
&\text{– the binary guarded command operator } : : C \times P \to P; \\
&\text{– the binary parallel composition operator } \| : P \times P \to P; \\
&\text{– the binary left merge operator } \lfloor \lfloor : P \times P \to P; \\
&\text{– for each } H \subseteq A \setminus \{ t \}, \text{ the unary encapsulation operator } \partial_H : P \to P; \\
&\text{– for each } I \subseteq A, \text{ the unary pre-abstraction operator } t_I : P \to P.
\end{align*} \]

The algebraic theory \( \text{PA}_\delta^{cc} \) has the following constants and operators to build
terms of sort \( C \):

\[ \begin{align*}
&\text{– the bottom constant } \bot : C; \\
&\text{– the top constant } \top : C; \\
&\text{– for each } a \in A, \text{ the enabledness constant } \mathcal{E}_a : C; \\
&\text{– the unary complement operator } - : C \to C;
\end{align*} \]

\[ ^1 \text{This limited kind of abstraction was not given a name in [5]. The name pre-}
\text{abstraction originates from [1].} \]
– the binary \textit{join} operator $\sqcup \colon C \times C \to C$;
– the binary \textit{meet} operator $\sqcap \colon C \times C \to C$;
– for each $H \subseteq A \setminus \{t\}$, the unary \textit{encapsulation} operator $\partial_H \colon C \to C$;
– for each $I \subseteq A$, the unary \textit{pre-abstraction} operator $t_I \colon C \to C$.

We use infix notation for the binary operators. The following precedence conventions are used to reduce the need for parentheses. The operators to build terms of sort $C$ bind stronger than the operators to build terms of sort $P$. The operator $\cdot$ binds stronger than all other binary operators to build terms of sort $P$ and the operator $+$ binds weaker than all other binary operators to build terms of sort $P$.

Let $p$ and $q$ be closed terms of sort $P$, $\zeta$ and $\xi$ be closed terms of sort $C$, $a \in A$, $H \subseteq A \setminus \{t\}$, and $I \subseteq A$. Intuitively, the constants and operators to build terms of sort $P$ can be explained as follows:

– $\delta$ can neither perform an action nor terminate successfully;
– $a$ first performs action $a$ unconditionally and then terminates successfully;
– $p + q$ behaves either as $p$ or as $q$, but not both;
– $p \cdot q$ first behaves as $p$, but when $p$ terminates successfully it continues by behaving as $q$;
– $\zeta \rightarrow p$ behaves as $p$ under condition $\zeta$;
– $p \parallel q$ behaves as the process that proceeds with $p$ and $q$ in parallel;
– $p \mid q$ behaves the same as $p \parallel q$, except that it starts with performing an action of $p$;
– $\partial_H(p)$ behaves the same as $p$, except that actions from $H$ are blocked.
– $t_I(p)$ behaves the same as $p$, except that it performs the internal action $t$ whenever $p$ would perform an action in $I$.

Intuitively, the constants and operators to build terms of sort $C$ can be explained as follows:

– $\mathcal{E}_a$ is a condition that holds if action $a$ is enabled in the context present;
– $\bot$ is a condition that never holds;
– $\top$ is a condition that always holds;
– $\neg \zeta$ is the opposite of $\zeta$;
– $\zeta \sqcup \xi$ is either $\zeta$ or $\xi$;
– $\zeta \sqcap \xi$ is both $\zeta$ and $\xi$;
– $\partial_H(\zeta)$ is $\zeta$ with all enabledness conditions $\mathcal{E}_a$ with $a \in H$ replaced by $\bot$;
– $t_I(\zeta)$ is $\zeta$ with all enabledness conditions $\mathcal{E}_a$ with $a \in I$ replaced by $\top$.

Some earlier extensions of ACP include conditional expressions of the form $p \triangleleft \zeta \triangleright q$; see e.g. [2]. This notation with triangles originates from [19]. We treat conditional expressions of the form $p \triangleleft \zeta \triangleright q$, where $p$ and $q$ are terms of sort $P$ and $\zeta$ is a term of sort $C$, as abbreviations. That is, we write $p \triangleleft \zeta \triangleright q$ for $\zeta \rightarrow p + \neg \zeta \rightarrow q$.

We will use the sum notation $\sum_{i \in I} p_i$, where $I = \{i_1, \ldots, i_n\}$ and $p_{i_1}, \ldots, p_{i_n}$ are terms of sort $P$, for $p_{i_1} + \ldots + p_{i_n}$. The convention is that $\sum_{i \in \emptyset} p_i$ stands for $\delta$ if $I = \emptyset$. 

4
are actually axiom schemas: \( a \) and the additional axioms given in Table 2. Several axioms given in Table 2

\[
\phi \sqcup \bot = \phi \\
\phi \sqcup \neg \phi = T \\
\phi \sqcup \psi \equiv \psi \sqcup \phi \\
\phi \sqcup (\psi \sqcap \chi) = (\phi \sqcup \psi) \sqcap (\phi \sqcup \chi)
\]

Table 1. Axioms of Boolean algebras

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Symbolic Form</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA1</td>
<td>( \top \rightarrow x = x )</td>
<td>GC1</td>
</tr>
<tr>
<td>BA2</td>
<td>( \bot \rightarrow x = \delta )</td>
<td>GC2</td>
</tr>
<tr>
<td>BA3</td>
<td>( \phi \rightarrow \delta = \delta )</td>
<td>GC3</td>
</tr>
<tr>
<td>BA4</td>
<td>( \phi \rightarrow (x + y) = (x + \phi) \rightarrow y )</td>
<td>GC4</td>
</tr>
<tr>
<td>BA5</td>
<td>( \phi \rightarrow x \cdot y = (\phi \rightarrow x) \cdot y )</td>
<td>GC5</td>
</tr>
<tr>
<td>BA6</td>
<td>( \phi \rightarrow (\psi \rightarrow x) = (\phi \sqcap \psi) \rightarrow x )</td>
<td>GC6</td>
</tr>
<tr>
<td>BA7</td>
<td>( \phi \rightarrow (\psi \rightarrow x) = (\psi \rightarrow x) = (\phi \rightarrow x) \rightarrow \psi \rightarrow x )</td>
<td>GC7</td>
</tr>
<tr>
<td>BA8</td>
<td>( \phi \rightarrow (\psi \rightarrow x) = (\psi \rightarrow x) = (\phi \rightarrow x) \rightarrow \psi \rightarrow x )</td>
<td>GC8</td>
</tr>
</tbody>
</table>

Table 2. Axioms of \( \text{PA}_5^\complement \)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Symbolic Form</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>( a \sqcup x = a \cdot x )</td>
<td>ED1</td>
</tr>
<tr>
<td>M2</td>
<td>( a \cdot y = a \cdot (x \sqcup y) )</td>
<td>ED2</td>
</tr>
<tr>
<td>M3</td>
<td>( a \cdot y = a \cdot (x \sqcup y) )</td>
<td>ED3</td>
</tr>
<tr>
<td>M4</td>
<td>( a \cdot y = a \cdot (x \sqcup y) )</td>
<td>ED4</td>
</tr>
<tr>
<td>M5</td>
<td>( a \cdot y = a \cdot (x \sqcup y) )</td>
<td>ED5</td>
</tr>
<tr>
<td>M6</td>
<td>( a \cdot y = a \cdot (x \sqcup y) )</td>
<td>ED6</td>
</tr>
<tr>
<td>M7</td>
<td>( a \cdot y = a \cdot (x \sqcup y) )</td>
<td>ED7</td>
</tr>
</tbody>
</table>

The axioms of \( \text{PA}_5^\complement \) are the axioms of Boolean Algebras (BA) given in Table 1 and the additional axioms given in Table 2. Several axioms given in Table 2 are actually axiom schemas: \( a \) and \( b \) stand for arbitrary constants of sort \( \text{P} \).
\( (i.e. \ a, b \in A_3) \), \( H \) stands for an arbitrary subset of \( A \setminus \{t\} \), and \( I \) stands for an arbitrary subset of \( A \).

The axioms of BA given in Table 1 have been taken from [18]. Several alternatives for this axiomatization can be found in the literature. If we use basic laws of BA other than axioms BA1–BA8 in a step of a derivation, we will refer to them as applications of BA and not give their derivation from axioms BA1–BA8.

The terms of sort \( C \) are interpreted in \( C \) as usual.

The axioms of \( \text{PA}^\delta_\text{cc} \) include the axioms of \( \text{PA}_\delta \), the usual axioms for encapsulation and the usual axioms for pre-abstraction (see e.g. [1]). Axioms GC1–GC7 have been taken from [2], but with the axiom \( x \cdot z < \phi > y \cdot z = (x < \phi > y) \cdot z \) (CO5) replaced by \( \phi : \rightarrow x \cdot y = (\phi : \rightarrow x) \cdot y \) (GC5).

An interesting subtheory of \( \text{PA}^\delta_\text{cc} \) is \( \text{BPA}^\delta_\text{cc} \). This subtheory is obtained by removing the parallel composition operator, the left merge operator, the encapsulation operators and the pre-abstraction operators from the signature of \( \text{PA}^\delta_\text{cc} \) and removing all axioms in which these operators occur from the axioms of \( \text{PA}^\delta_\text{cc} \) – in other words, the axioms of \( \text{BPA}^\delta_\text{cc} \) are BA1–BA8, A1–A7, GC1–GC7, ET.

To prove a statement for all closed terms of sort \( P \) from the language of \( \text{BPA}^\delta_\text{cc} \), it is sufficient to prove it for all basic terms. The set \( B \) of basic terms is inductively defined by the following rules:

- \( \delta \in B \);
- if \( \zeta \) is a closed term of sort \( C \) and \( a \in A \), then \( \zeta : \rightarrow a \in B \);
- if \( \zeta \) is a closed term of sort \( C \), \( a \in A \) and \( p \in B \), then \( \zeta : \rightarrow a \cdot p \in B \);
- if \( p, q \in B \), then \( p + q \in B \).

The basic terms are exactly the closed terms of sort \( P \) from the language of \( \text{BPA}^\delta_\text{cc} \) of the form

\[
\sum_{i<n} \zeta_i : \rightarrow a_i \cdot p_i + \sum_{i<m} \xi_i : \rightarrow b_i ,
\]

where \( a_0, \ldots, a_{n-1}, b_0, \ldots, b_{m-1} \in A \), \( \zeta_0, \ldots, \zeta_{n-1}, \xi_0, \ldots, \xi_{m-1} \) are closed term of sort \( C \) and \( p_0, \ldots, p_{n-1} \) are basics terms \((n, m \geq 0)\). We can prove that all closed terms of sort \( P \) from the language of \( \text{BPA}^\delta_\text{cc} \) are derivably equal to a basic term.

**Lemma 2.1 (Elimination for \( \text{BPA}^\delta_\text{cc} \)).** For all closed terms \( p \) of sort \( P \) from the language of \( \text{BPA}^\delta_\text{cc} \), there exists a basic term \( q \in B \) such that \( \text{BPA}^\delta_\text{cc} \vdash p = q \).

**Proof.** The term rewriting system consisting of axioms A4–A7 and GC1–GC7 oriented from left to right is strongly normalizing. This can be proved by using the method of lexicographical path ordering of Kamin and Lévy (see e.g. [20]), making the signature one-sorted, taking the ordering \( : \rightarrow > \cdot > + \), \( : \rightarrow > \delta, \sqcup \), and giving the lexicographical status for the first argument to \( \cdot \) and the lexicographical status for the second argument to \( : \rightarrow \). Moreover, it is easy to see that each normal form is a basic term.

We can also prove that all closed terms of sort \( P \) from the language of \( \text{PA}^\delta_\text{cc} \) are derivably equal to a basic term.
Lemma 2.2 (Elimination for PA\textsuperscript{cc}δ). For all closed terms \( p \) of sort \( \mathcal{P} \) from the language of PA\textsuperscript{cc}δ, there exists a basic term \( q \in \mathcal{B} \) such that PA\textsuperscript{cc}δ \( \vdash p = q \).

Proof. The term rewriting system consisting of all axioms of PA\textsuperscript{cc}δ, except BA1–BA8 and A1–A3, oriented from left to right is strongly normalizing. This can be proved by using the method of lexicographical path ordering of Kamin and Lévy, making the signature one-sorted, ranking the operators \( \parallel \) and \( \| \) as in [9], taking the ordering \( \ldots > \|_3 > \|_3 > \|_2 > \|_2 > \triangleright > \cdot > +, \triangleright > \delta, \sqcap, \partial_H > \triangleright, \perp, -, \sqcup \) for all \( H \subseteq \mathcal{A} \setminus \{ t \} \), \( t_I > \triangleright, t, \top, -, \sqcup \) for all \( I \subseteq \mathcal{A} \), and giving the lexicographical status for the first argument to \( \cdot \) and the lexicographical status for the second argument to \( \triangleright \). Moreover, it is easy to see that each normal from is a basic term. \( \square \)

Moreover, we can prove that PA\textsuperscript{cc}δ is a conservative extension of BPA\textsuperscript{cc}δ.

Lemma 2.3 (Conservative extension). If \( p \) and \( q \) are closed terms of sort \( \mathcal{P} \) from the language of BPA\textsuperscript{cc}δ, then BPA\textsuperscript{cc}δ \( \vdash p = q \) iff PA\textsuperscript{cc}δ \( \vdash p = q \).

Proof. The implication from left to right follows immediately from the fact that the axioms of BPA\textsuperscript{cc}δ are included in the axioms of PA\textsuperscript{cc}δ. The implication from right to left is proved as follows. Let \( p \) and \( q \) be closed terms of sort \( \mathcal{P} \) from the language of BPA\textsuperscript{cc}δ such that PA\textsuperscript{cc}δ \( \vdash p = q \). Because it is left-linear and non-overlapping (see e.g. [20]), the term rewriting system used in the proof of Lemma 2.2 is confluent modulo axioms A1–A3. Consequently, the reductions of \( p \) and \( q \) by means of this term rewriting system yields the same normal form modulo axioms A1–A3. Moreover, the reductions of \( p \) and \( q \) only use axioms A4–A7 and GC1–GC7, oriented from left to right, because the additional operators of PA\textsuperscript{cc}δ do not occur in \( p \) and \( q \), and no rewrite rule introduces occurrences of those operators that were not already there in its left-hand side. Hence, the reduction of \( p \) into its normal form followed by the reverse of the reduction of \( q \) into its normal form is a proof of \( p = q \) in BPA\textsuperscript{cc}δ. \( \square \)

The preceding three lemmas will be useful in the completeness proof of PA\textsuperscript{cc}δ for the full splitting bisimulation models of PA\textsuperscript{cc}δ that will be introduced in Section 4.

We proceed to the presentation of the structural operational semantics of PA\textsuperscript{cc}δ. The following relations on closed terms of sort \( \mathcal{P} \) from the language of PA\textsuperscript{cc}δ are used:

- for each \( \ell \in (\mathcal{C} \setminus \{ \perp \}) \times \mathcal{A} \), a binary relation \( \ell \rightarrow \); 
- for each \( \ell \in (\mathcal{C} \setminus \{ \perp \}) \times \mathcal{A} \), a unary relation \( \ell \rightarrow \sqrt{\cdot} \).

We write \( p \xrightarrow{[\alpha]a} q \) instead of \((p, q) \in \frac{(\alpha, a)}{\rightarrow} \) and \( p \xrightarrow{[\alpha]a} \sqrt{\cdot} \) instead of \( p \in \frac{(\alpha, a)}{\rightarrow} \sqrt{\cdot} \).

The relations \( \ell \rightarrow \sqrt{\cdot} \) and \( \ell \rightarrow \) can be explained as follows:

- \( p \xrightarrow{[\alpha]a} \sqrt{\cdot} \): \( p \) is capable of performing action \( a \) under condition \( \alpha \) and then terminating successfully;
- \( p \xrightarrow{[\alpha]a} q \): \( p \) is capable of performing action \( a \) under condition \( \alpha \) and then proceeding as \( q \).
Section 5. We will return to this structural operational semantics in Section 4, we will make use of conditional transition systems and splitting bisimilarity of conditional transition systems. In Section 4, we will make use of conditional transition systems and splitting bisimilarity of conditional transition systems. In Section 5, we will show that the structural operational semantics of PA is described by the transition rules given in Table 3. We will return to this structural operational semantics in Section 5.

### Table 3. Transition rules for PA

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a \xrightarrow{I} a ) ( \checkmark )</td>
<td>( \parallel x \downarrow a \parallel )</td>
</tr>
<tr>
<td>( x \parallel y \downarrow a \parallel ) ( \checkmark )</td>
<td>( x \parallel y \downarrow a \parallel x ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \downarrow a \parallel x ) ( \checkmark )</td>
<td>( x \downarrow a \parallel x ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \downarrow a \parallel x ) ( \checkmark )</td>
<td>( x \downarrow a \parallel x ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \downarrow a \parallel x ) ( \checkmark )</td>
<td>( x \downarrow a \parallel x ) ( \checkmark )</td>
</tr>
<tr>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
<td>( x \parallel y \downarrow a \parallel y ) ( \checkmark )</td>
</tr>
</tbody>
</table>

The structural operational semantics of PA is described by the transition rules given in Table 3. We will return to this structural operational semantics in Section 5.

### 3 Transition Systems and Splitting Bisimilarity for PA

In this section, we introduce conditional transition systems and splitting bisimilarity of conditional transition systems. In Section 4, we will make use of conditional transition systems and splitting bisimilarity of conditional transition systems to construct models of PA. In Section 5, we will show that the structural operational semantics presented in Section 2 induces a conditional transition system for each closed term of sort P from the language of PA.

Conditional transition systems are labelled transition systems of which the labels consist of a condition different from ⊥ and an action. Labels of this kind are sometimes called guarded actions. Henceforth, we write \( C^= \) for \( C \setminus \{ \bot \} \).

A conditional transition system \( T \) consists of the following:

- a set \( S \) of states;
– a set $\ell \rightarrow \subseteq S \times S$, for each $\ell \in C \times A$;
– a set $\ell \rightarrow \square \subseteq S$, for each $\ell \in C \times A$;
– an initial state $s^0 \in S$.

If $(s, s') \in \ell \rightarrow$ for some $\ell \in C \times A$, then we say that there is a transition from $s$ to $s'$. We usually write $s \xrightarrow{\alpha,a} s'$ instead of $(s, s') \in (\ell \rightarrow a)$, and $s \xrightarrow{\alpha,a} \square$ instead of $s \in (\ell \rightarrow \square)$. Furthermore, we write $\rightarrow$ for the family of sets $\{\ell \rightarrow a\}_{\ell \in C \times A}$ and $\rightarrow \square$ for the family of sets $\{\ell \rightarrow \square\}_{\ell \in C \times A}$.

The relations $\rightarrow \square$ and $\rightarrow$ can be explained as follows:

– $s \xrightarrow{\alpha,a} \square$: in state $s$, it is possible to perform action $a$ under condition $\alpha$, and by doing so to terminate successfully;
– $s \xrightarrow{\alpha,a} s'$: in state $s$, it is possible to perform action $a$ under condition $\alpha$, and by doing so to make a transition to state $s'$.

A conditional transition system may have states that are not reachable from its initial state by a number of transitions. Connected conditional transition systems are transition systems without unreachable states.

Let $T = (S, \rightarrow, \rightarrow \square, s^0)$ be a conditional transition system. Then the reachability relation of $T$ is the smallest relation $\subseteq S \times S$ such that:

– $s \rightarrow s$;
– if $s \xrightarrow{\ell} s'$ and $s' \rightarrow s''$, then $s \rightarrow s''$.

We write $RS(T)$ for $\{s \in S \mid s^0 \rightarrow s\}$. $T$ is called a connected conditional transition system if $S = RS(T)$. Henceforth, we will only consider connected conditional transition systems. This often calls for extraction of the connected part of a conditional transition system that is composed of connected conditional transition systems.

Let $T = (S, \rightarrow, \rightarrow \square, s^0)$ be a conditional transition system that is not necessarily connected. Then the connected part of $T$, written $\Gamma(T)$, is defined as follows:

$$\Gamma(T) = (S', \rightarrow', \rightarrow' \square, s^0) ,$$

where

$$S' = RS(T) ,$$

and for every $\ell \in C \times A$:

$$\ell \rightarrow' = \ell \cap (S' \times S') ,$$

$$\ell \rightarrow' \square = \ell \cap S' .$$

It is assumed that for each infinite cardinal $\kappa$ a fixed but arbitrary set $S_\kappa$ with the following properties has been given:

– the cardinality of $S_\kappa$ is greater than or equal to $\kappa$;
if \( S_1, S_2 \subseteq S_\kappa \), then \( S_1 \uplus S_2 \subseteq S_\kappa \) and \( S_1 \times S_2 \subseteq S_\kappa \).\(^2\)

Let \( \kappa \) be an infinite cardinal. Then \( \text{CTS}_\kappa \) is the set of all connected conditional transition systems \( T = (S, \rightarrow, \rightarrow_\sqrt{1}, s^0) \) such that \( S \subseteq S_\kappa \) and the branching degree of \( T \) is less than \( \kappa \), i.e. for all \( s \in S \), the cardinality of the set \( \{ (\ell, s') \in (C^- \times A) \times S \mid (s, s') \in \mathcal{L} \} \cup \{ \ell \in C^- \times A \mid s \in \mathcal{L}, \sqrt{1} \} \) is less than \( \kappa \).\(^3\)

The condition \( S \subseteq S_\kappa \) guarantees that \( \text{CTS}_\kappa \) is indeed a set.

A conditional transition system is said to be \textit{finitely branching} if its branching degree is less than \( \aleph_0 \). Otherwise, it is said to be \textit{infinitely branching}.

Conditional transition system that differ only with respect to the identity of the states are isomorphic.

Let \( T_1 = (S_1, \rightarrow_1, \rightarrow_\sqrt{1}, s^0_1) \) and \( T_2 = (S_2, \rightarrow_2, \rightarrow_\sqrt{2}, s^0_2) \) be conditional transition systems. Then \( T_1 \) and \( T_2 \) are \textit{isomorphic}, written \( T_1 \cong T_2 \), if there exists a bijective function \( b : S_1 \rightarrow S_2 \) such that:

- \( b(s^0_1) = s^0_2 \);
- \( s_1 \xrightarrow{\ell} s'_1 \) iff \( b(s_1) \xrightarrow{\ell} b(s'_1) \);
- \( s \xrightarrow{\ell, \sqrt{1}} s' \) iff \( b(s) \xrightarrow{\ell} \sqrt{2} \).

Henceforth, we will always consider two conditional transition systems essentially the same if they are isomorphic.

**Remark 3.1.** The set \( \text{CTS}_\kappa \) is independent of \( S_\kappa \). By that we mean the following. Let \( \text{CTS}_\kappa \) and \( \text{CTS}'_\kappa \) result from different choices for \( S_\kappa \). Then there exists a bijection \( b : \text{CTS}_\kappa \rightarrow \text{CTS}'_\kappa \) such that for all \( T \in \text{CTS}_\kappa \), \( T \cong b(T) \).

Bisimilarity has to be adapted to the setting with guarded actions. In the definition given below, we use a well-known notion from the field of Boolean algebras: the partial order relation \( \subseteq \) on \( C \) defined by

\[ \alpha \subseteq \beta \quad \text{iff} \quad \alpha \cup \beta = \beta . \]

Moreover, we use the notation \( \bigsqcup A \), where \( A = \{ \alpha_1, \ldots, \alpha_n \} \subseteq C \), for \( \alpha_1 \cup \ldots \cup \alpha_n \).

Let \( T_1 = (S_1, \rightarrow_1, \rightarrow_\sqrt{1}, s^0_1) \in \text{CTS}_\kappa \) and \( T_2 = (S_2, \rightarrow_2, \rightarrow_\sqrt{2}, s^0_2) \in \text{CTS}_\kappa \) (for an infinite cardinal \( \kappa \)). Then a \textit{splitting bisimulation} \( B \) between \( T_1 \) and \( T_2 \) is a binary relation \( B \subseteq S_1 \times S_2 \) such that \( B(s_1, s_2) \) and for all \( s_1, s_2 \) such that \( B(s_1, s_2) \):

- if \( s_1 \xrightarrow{[\alpha]} s'_1 \), then there is a set \( CS'_2 \subseteq C^- \times S_2 \) such that \( \alpha \subseteq \bigsqcup \text{dom}(CS'_2) \) and for all \( (\alpha', s'_2) \in CS'_2, s_2 \xrightarrow{[\alpha']} s' \) and \( B(s'_1, s'_2) \);

\(^2\) We write \( A \uplus B \) for the disjoint union of sets \( A \) and \( B \), i.e. \( A \uplus B = (A \times \{ 0 \}) \cup (B \times \{ 1 \}) \). We write \( \mu_1 \) and \( \mu_2 \) for the associated injections \( \mu_1 : A \rightarrow A \uplus B \) and \( \mu_2 : B \rightarrow A \uplus B \), defined by \( \mu_1(a) = (a, 0) \) and \( \mu_2(b) = (b, 1) \).

\(^3\) In [11], the definition of \( \text{CTS}_\kappa \) is given for an arbitrary set of \textit{atomic conditions}. In the case where the set \( \{ E_a \mid a \in A \} \) is taken as the set of atomic conditions, that definition and the definition given here are essentially the same.
– if $s_2 \xrightarrow{[\alpha]} s'_2$, then there is a set $CS'_1 \subseteq C^- \times S_1$ such that $\alpha \sqsubseteq \bigcup \text{dom}(CS'_1)$ and for all $(\alpha', s'_1) \in CS'_1$, $s_1 \xrightarrow{[\alpha']} s'_1$ and $B(s'_1, s'_2)$;
– if $s_1 \xrightarrow{[\alpha]} \sqrt{1}$, then there is a set $C' \subseteq C^-$ such that $\alpha \sqsubseteq \bigcup C'$ and for all $\alpha' \in C'$, $s_2 \xrightarrow{[\alpha']} \sqrt{2}$;
– if $s_2 \xrightarrow{[\alpha]} \sqrt{2}$, then there is a set $C' \subseteq C^-$ such that $\alpha \sqsubseteq \bigcup C'$ and for all $\alpha' \in C'$, $s_1 \xrightarrow{[\alpha']} \sqrt{1}$.

Two conditional transition systems $T_1, T_2 \in \text{CTS}_\kappa$ are splitting bisimilar, written $T_1 \equiv T_2$, if there exists a splitting bisimulation $B$ between $T_1$ and $T_2$. Let $B$ be a splitting bisimulation between $T_1$ and $T_2$. Then we say that $B$ is a splitting bisimulation witnessing $T_1 \equiv T_2$.

The name splitting bisimulation is used because a transition of one of the related transition systems may be simulated by a set of transitions of the other transition system. Splitting bisimulation should not be confused with split bisimulation [15].

It is easy to see that $\equiv$ is an equivalence on $\text{CTS}_\kappa$. Let $T \in \text{CTS}_\kappa$. Then we write $[T]_\equiv$ for $\{T' \in \text{CTS}_\kappa \mid T \equiv T'\}$, i.e. the $\equiv$-equivalence class of $T$. We write $\text{CTS}_\kappa/\equiv$ for the set of equivalence classes $\{[T]_\equiv \mid T \in \text{CTS}_\kappa\}$.

In Section 4, we will use $\text{CTS}_\kappa/\equiv$ as the domain of a structure that is a model of $\text{PA}^\text{ce}_\delta$. As the domain of a structure, $\text{CTS}_\kappa/\equiv$ must be a set. That is the case because $\text{CTS}_\kappa$ is a set. The latter is guaranteed by considering only conditional transition systems of which the set of states is a subset of $S_\kappa$.

Remark 3.2. The question arises whether $S_\kappa$ is large enough if its cardinality is greater than or equal to $\kappa$. This question can be answered in the affirmative. Let $T = (S, \to, \to\sqrt{\cdot}, s^0)$ be a connected conditional transition system of which the branching degree is less than $\kappa$. Then there exists a connected conditional transition system $T' = (S', \to', \to\sqrt{\cdot}', s^{0'})$ of which the branching degree is less than $\kappa$ such that $T \equiv T'$ and the cardinality of $S'$ is less than $\kappa$.

It is easy to see that, if we would consider conditional transition systems with unreachable states as well, each conditional transition system would be splitting bisimilar to its connected part. This justifies the choice to consider only connected conditional transition systems. It is easy to see that isomorphic conditional transition systems are splitting bisimilar. This justifies the choice to consider conditional transition systems essentially the same if they are isomorphic.

4 Full Splitting Bisimulation Models of $\text{PA}^\text{ce}_\delta$

In this section, we introduce the full splitting bisimulation models of $\text{PA}^\text{ce}_\delta$. They are models of which the domain consists of equivalence classes of conditional transition systems modulo splitting bisimilarity. The qualification “full” originates from [10]. It expresses that there exist other splitting bisimulation models,
but each of them is isomorphically embedded in a full splitting bisimulation model.

The models of PA\textsubscript{cc} are structures that consist of the following:

- a non-empty set \( D \), called the domain of the model;
- for each constant of PA\textsubscript{cc}, an element of \( D \);
- for each \( n \)-ary operator of PA\textsubscript{cc}, an \( n \)-ary operation on \( D \).

In the full splitting bisimulation models of PA\textsubscript{cc} that are introduced in this section, the domain is \( \text{CTS}_{\kappa}/\llnot \) for some infinite cardinal \( \kappa \). We obtain the models concerned by associating certain elements of \( \text{CTS}_{\kappa}/\llnot \) with the constants of PA\textsubscript{cc} and certain operations on \( \text{CTS}_{\kappa}/\llnot \) with the operators of PA\textsubscript{cc}. We begin by associating elements of \( \text{CTS}_{\kappa} \) and operations on \( \text{CTS}_{\kappa} \) with the constants and operators. The result of this is subsequently lifted to \( \text{CTS}_{\kappa}/\llnot \).

It is assumed that for each infinite cardinal \( \kappa \) a fixed but arbitrary function \( \text{ch}_{\kappa} : (\mathcal{P}(S_{\kappa}) \setminus \emptyset) \to S_{\kappa} \) such that for all \( S \in \mathcal{P}(S_{\kappa}) \setminus \emptyset \), \( \text{ch}_{\kappa}(S) \in S \) has been given.

We will use the abbreviation \( s a\rightarrow s' \) for \( s a\rightarrow s' \land (s a\rightarrow\sqrt{ } \land s' = s'') \).

We associate with each constant \( c \) of PA\textsubscript{cc} an element \( \hat{c} \) of \( \text{CTS}_{\kappa} \) and with each operator \( f \) of PA\textsubscript{cc} an operation \( \hat{f} \) on \( \text{CTS}_{\kappa} \) as follows.

- \( \hat{\delta} = (\{s^0\}, \emptyset, \emptyset, s^0) \),
  where \( s^0 = \text{ch}_{\kappa}(S_{\kappa}) \).

- \( \hat{a} = (\{s^0\}, \emptyset, \rightarrow, s^0) \),
  where \( s^0 = \text{ch}_{\kappa}(S_{\kappa}) \),
  \[ (\top, a) \rightarrow \sqrt{} = \{s^0\} , \]
  and for every \((\alpha, a') \in (C^{-} \times A) \setminus \{(\top, a)\}:
  \[ (\alpha, a') \rightarrow \sqrt{} = \emptyset \].

- Let \( T_i = (S_i, \rightarrow, i, \rightarrow, s_i^0) \in \text{CTS}_{\kappa} \) for \( i = 1, 2 \). Then
  \[ T_1 \uplus T_2 = \Gamma(S, \rightarrow, i, \rightarrow, s_i^0) \],
  where \( s_i^0 = \text{ch}_{\kappa}(S_{\kappa} \setminus (S_1 \uplus S_2)) \),
  \( S = \{s_1^0\} \cup (S_1 \uplus S_2) \),
  and for every \((\alpha, a) \in C^{-} \times A:
  \[ (\alpha, a) \rightarrow \sqrt{} = \{(s^0, \mu_1(s)) | s^0 \uplus \mu_1(s) \}\]
  \[ \cup \{(s^0, \mu_2(s)) | s^0 \uplus \mu_2(s) \}\]
  \[ \cup \{(\mu_1(s), \mu_1(s')) | s \uplus \mu_1(s') \}\]
  \[ \cup \{(\mu_2(s), \mu_2(s')) | s \uplus \mu_2(s') \} \],

12
\[
\frac{(a,a)}{(a,a)} \overline{\sqrt{}} = \{ s^0_i \mid s^0_1 \overset{[a]a}{\rightarrow} \sqrt{1} \}
\]
\[
\cup \{ s^0_i \mid s^0_2 \overset{[a]a}{\rightarrow} \sqrt{2} \}
\]
\[
\cup \{ \mu_1(s) \mid s \overset{[a]a}{\rightarrow} \sqrt{1} \}
\]
\[
\cup \{ \mu_2(s) \mid s \overset{[a]a}{\rightarrow} \sqrt{2} \}
\]

Let \( T_i = (S_i, \rightarrow_i, \rightarrow_i, s^0_i) \in \text{CTS}_\kappa \) for \( i = 1, 2 \). Then
\[
T_1 \upharpoonright T_2 = \Gamma(S, \rightarrow, \rightarrow, s^0_1),
\]
where
\[
S = S_1 \cup S_2,
\]
and for every \((a, a) \in C^- \times A:\)
\[
\frac{(a,a)}{(a,a)} = \{(\mu_1(s), \mu_2(s')) \mid s \overset{[a]a}{\rightarrow} s' \}
\]
\[
\cup \{ (\mu_1(s), \mu_2(s')) \mid s \overset{[a]a}{\rightarrow} \sqrt{1} \}
\]
\[
\cup \{ (\mu_2(s), \mu_2(s')) \mid s \overset{[a]a}{\rightarrow} \sqrt{2} \},
\]
\[
\frac{(a,a)}{(a,a)} \sqrt{\alpha} = \{ \mu_2(s') \mid s \overset{[a]a}{\rightarrow} \sqrt{2} \}
\]

Let \( T = (S, \rightarrow, \rightarrow, s^0) \in \text{CTS}_\kappa \). Then
\[
\alpha \vdash T = \Gamma(S, \rightarrow', \rightarrow', s^0'),
\]
where for every \((a', a) \in C^- \times A:\)
\[
\frac{(a',a)}{(a',a)} = \{(s^0, s') \mid \exists \beta \bullet s^0 \overset{[\beta]a}{\rightarrow} s' \land \alpha' = \alpha \land \beta\}
\]
\[
\cup \{ (s, s') \mid s \overset{[a']a}{\rightarrow} s' \land s \neq s^0 \}
\]
\[
\frac{(a',a)}{(a',a)} \sqrt{\alpha'} = \{ s^0 \mid \exists \beta \bullet s^0 \overset{[\beta]a}{\rightarrow} \sqrt{\alpha'} \land s = s^0 \}
\]
\[
\cup \{ s \mid s \overset{[a']a}{\rightarrow} \sqrt{\alpha'} \land s \neq s^0 \},
\]

Let \( T_i = (S_i, \rightarrow_i, \rightarrow_i, s^0_i) \in \text{CTS}_\kappa \) for \( i = 1, 2 \). Then
\[
T_1 \parallel T_2 = (S, \rightarrow, \rightarrow, s^0),
\]
where
\[
s^0_i = (s^0_1, s^0_2),
\]
\[
s^0 = \text{ch}_\kappa(S \setminus (S_1 \cup S_2)),
\]
\[
S = ((S_1 \cup \{ s^0 \}) \times (S_2 \cup \{ s^0 \})) \setminus \{ (s^0, s^0) \},
\]
and for every \((a, a) \in C^- \times A:\)
\[
\frac{(a,a)}{(a,a)} = \{ ((s_1, s_2), (s'_1, s'_2)) \mid (s'_1, s_2) \in S \land s_1 \overset{[a]a}{\rightarrow} s'_1 \land s^0 \}
\]
\[
\cup \{ ((s_1, s_2), (s_1, s'_2)) \mid (s_1, s'_2) \in S \land s_2 \overset{[a]a}{\rightarrow} s'_2 \land s^0 \},
\]
\[
\frac{(a,a)}{(a,a)} \sqrt{\alpha} = \{ (s_1, s^0) \mid s_1 \overset{[a]a}{\rightarrow} \sqrt{1} \}
\]
\[
\cup \{ (s^0, s_2) \mid s_2 \overset{[a]a}{\rightarrow} \sqrt{2} \}.
\]
Let $T_i = (S_i, \rightarrow_i, \rightarrow_i \sqrt{i}, s_i^0) \in \text{CTS}_\kappa$ for $i = 1, 2$. Suppose that $T_1 \parallel T_2 = (S, \rightarrow, \rightarrow \sqrt{2}, s^0)$ where $S = ((S_1 \cup \{s^\prime\}) \times (S_2 \cup \{s^\prime\})) \setminus \{(s^\prime, s^\prime)\}$ and $s^\prime = \text{ch}_\kappa(S_\kappa \setminus (S_1 \cup S_2))$. Then

\[ T_1 \parallel T_2 = \Gamma(S', \rightarrow', \rightarrow \sqrt{2}, s^{0'}) \]

where

$s^{0'} = \text{ch}_\kappa(S_\kappa \setminus S)$,

$S' = \{s^{0'}\} \cup S$,

and for every $(\alpha, a) \in C^- \times A$:

\[ \overrightarrow{(a,a)} = \{(s', s_1^0) \mid s_1^0 \rightarrow_{a_1} s' \} \cup \{(a,a)\} \]

– Let $T = (S, \rightarrow, \rightarrow \sqrt{2}, s^0) \in \text{CTS}_\kappa$. Then

\[ \partial_H(T) = \Gamma(S, \rightarrow', \rightarrow' \sqrt{2}, s^0) \]

where for every $(\alpha, a) \in C^- \times (A \setminus H)$:

\[ \overrightarrow{(a,a)} = \{(s, s') \mid \exists \alpha' \cdot s \stackrel{[\alpha']}{\rightarrow} s' \land \alpha = \partial_H(\alpha')\} \]

\[ \overrightarrow{(a,a)} = \{s \mid \exists \alpha' \cdot s \stackrel{[\alpha']}{\rightarrow} \land \alpha = \partial_H(\alpha')\} \]

and for every $(\alpha, a) \in C^- \times H$:

\[ \overrightarrow{(a,a)} = \emptyset \]

\[ \overrightarrow{(a,a)} = \emptyset \]

– Let $T = (S, \rightarrow, \rightarrow \sqrt{2}, s^0) \in \text{CTS}_\kappa$. Then

\[ \widehat{t_1}(T) = \Gamma(S, \rightarrow', \rightarrow' \sqrt{2}, s^0) \]

where for every $(\alpha, a) \in C^- \times ((A \setminus I) \setminus \{t\})$:

\[ \overrightarrow{(a,a)} = \{(s, s') \mid \exists \alpha' \cdot s \stackrel{[\alpha']}{\rightarrow} s' \land \alpha = t_1(\alpha')\} \]

\[ \overrightarrow{(a,a)} = \{s \mid \exists \alpha' \cdot s \stackrel{[\alpha']}{\rightarrow} \land \alpha = t_1(\alpha')\} \]

and for every $\alpha \in C^-$:

\[ \overrightarrow{(a,t)} = \{(s, s') \mid \exists \alpha' \cdot a \cdot s \stackrel{[\alpha']}{\rightarrow} s' \land a \in I \cup \{t\} \land \alpha = t_1(\alpha')\} \]

\[ \overrightarrow{(a,t)} = \{s \mid \exists \alpha' \cdot a \cdot s \stackrel{[\alpha']}{\rightarrow} \land a \in I \cup \{t\} \land \alpha = t_1(\alpha')\} \]

and for every $(\alpha, a) \in C^- \times (I \setminus \{t\})$:

\[ \overrightarrow{(a,a)} = \emptyset \]

\[ \overrightarrow{(a,a)} = \emptyset \]
In the definition of alternative composition on $\mathbb{CTS}_\kappa$, the connected part of a conditional transition system is extracted because the initial states of the conditional transition systems $T_1$ and $T_2$ may be unreachable from the new initial state. The new initial state is introduced because, in $T_1$ and/or $T_2$, there may exist a transition back to the initial state. In the definition of sequential composition on $\mathbb{CTS}_\kappa$, the connected part of a conditional transition system is extracted because the initial state of the conditional transition system $T_2$ may be unreachable from the initial state of the conditional transition system $T_1$ – due to absence of termination in $T_1$.

Remark 4.1. The elements of $\mathbb{CTS}_\kappa$ and the operations on $\mathbb{CTS}_\kappa$ defined above are independent of $\text{ch}_\kappa$. Different choices for $\text{ch}_\kappa$ lead for each constant of $\text{PA}_\kappa$ to isomorphic elements of $\mathbb{CTS}_\kappa$ and lead for each operator of $\text{PA}_\kappa$ to operations on $\mathbb{CTS}_\kappa$ with isomorphic results.

We can easily show that splitting bisimilarity is a congruence with respect to alternative composition, sequential composition, guarded command, parallel composition, left merge, encapsulation and pre-abstraction.

Lemma 4.1 (Congruence). Let $\kappa$ be an infinite cardinal. Then for all $T_1, T_2, T'_1, T'_2 \in \mathbb{CTS}_\kappa$ and $\alpha \in C$, $T_1 \equiv T'_1$ and $T_2 \equiv T'_2$ imply $T_1 \parallel T_2 \equiv T'_1 \parallel T'_2$, $T_1 \overset{\hat{\cdot}}{\overset{\hat{\cdot}}{\hat{\cdot}}}(T_2 \equiv T'_1 \overset{\hat{\cdot}}{\overset{\hat{\cdot}}{\hat{\cdot}}}(T'_2), \alpha \equiv \hat{\alpha} \equiv \hat{\alpha} \equiv \hat{\alpha} \equiv \hat{\alpha} \equiv \hat{\alpha} \equiv \hat{\alpha}$, $T_1 \parallel T_2 \equiv T'_1 \parallel T'_2$, $T_1 \parallel T_2 \equiv T'_1 \parallel T'_2$, $\partial_H(\hat{T}_1) \equiv \partial_H(\hat{T}'_1)$ and $\hat{\partial}_H(\hat{T}_1) \equiv \hat{\partial}_H(\hat{T}'_1)$.

Proof. Let $T_i = (S_i, \rightarrow_i, -\vee_i, s_i^0)$ and $T'_i = (S'_i, \rightarrow'_i, -\vee'_i, s'_i^0)$ for $i = 1, 2$. Let $R_1$ and $R_2$ be splitting bisimulations witnessing $T_1 \equiv T'_1$ and $T_2 \equiv T'_2$, respectively. Then we construct relations $R_\parallel, R_\parallel$, $R_\parallel, R_\parallel, R_\parallel, R_\parallel, R_\parallel, R_\parallel$, as follows:

- $R_\parallel = \{(s^0, s'^0)\} \cup \{(s^0, s'^0)\} \cup \{(s^0, s'^0)\} \cap (S \times S')$, where $S$ and $S'$ are the sets of states of $T_1 \parallel T_2$ and $T'_1 \parallel T'_2$, respectively, and $s^0$ and $s'^0$ are the initial states of $T_1 \parallel T_2$ and $T'_1 \parallel T'_2$, respectively;
- $R_\parallel = \{(s^0, s'^0)\} \cup \{(s^0, s'^0)\} \cap (S \times S')$, where $S$ and $S'$ are the sets of states of $T_2 \parallel T_2$, respectively, and $s^0$ and $s'^0$ are the initial states of $T_2 \parallel T_2$, respectively;
- $R_\parallel = \{(s^0, s'^0)\} \cap (S \times S')$, where $S$ and $S'$ are the sets of states of $T_1 \parallel T_2$, respectively, and $s^0$ and $s'^0$ are the initial states of $T_1 \parallel T_2$, respectively;
- $R_\parallel = \{(s^0, s'^0)\} \cup \{(s^0, s'^0)\} \cap (S \times S')$, where $S$ and $S'$ are the sets of states of $T_1 \parallel T_2$, respectively, and $s^0$ and $s'^0$ are the initial states of $T_1 \parallel T_2$, respectively;
Here, we write $\mu_i(R_i)$ for $\{(\mu_i(s), \mu_i(s')) \mid R_i(s, s')\}$, where $\mu_i$ is used to denote both the injection of $S_1$ into $S_1 \uplus S_2$ and the injection of $S'_1$ into $S'_1 \uplus S'_2$. Given the definitions of alternative composition, sequential composition, guarded command, parallel composition, left merge, encapsulation and pre-abstraction, it is easy to see that $R_\uplus, R_\cap, R_\parallel, R_\hat{\partial}_H$ and $R_{\hat{t}_i}$ are splitting bisimulations witnessing $T_1 \uplus T_2 \equiv T_1' \uplus T_2', T_1 \cap T_2 \equiv T_1' \cap T_2', \alpha \vDash T_1 \equiv \alpha \vDash T_1'$, $T_1 \parallel T_2 \equiv T_1'' \parallel T_2''$, $\tilde{t}_i(T_1) \equiv \tilde{t}_i(T_1'')$, respectively.

The full splitting bisimulation models $\mathfrak{P}_\kappa^\kappa$, one for each infinite cardinal $\kappa$, consist of the following:

- a set $\mathcal{P}$, called the domain of $\mathfrak{P}_\kappa^\kappa$;
- for each constant $c$ of $\text{PA}_\kappa^\kappa$, an element $\bar{c}$ of $\mathcal{P}$;
- for each $n$-ary operator $f$ of $\text{PA}_\kappa^\kappa$, an $n$-ary operation $\bar{f}$ on $\mathcal{P}$;

where those ingredients are defined as follows:

\[
\begin{align*}
\mathcal{P} &= \text{CTS}_\kappa/\equiv, \\
\bar{\delta} &= [\delta]_\equiv, \\
\bar{a} &= [a]_\equiv, \\
[T_1]_\equiv \oplus [T_2]_\equiv &= [T_1 \oplus T_2]_\equiv, \\
[T_1]_\equiv \cap [T_2]_\equiv &= [T_1 \cap T_2]_\equiv, \\
\bar{\partial}_H([T_1]_\equiv) &= [\partial_H(T_1)]_\equiv, \\
\bar{t}_i([T_1]_\equiv) &= [\tilde{t}_i(T_1)]_\equiv.
\end{align*}
\]

The operations alternative composition, sequential composition, guarded command, parallel composition, left merge, encapsulation and pre-abstraction on $\text{CTS}_\kappa/\equiv$ are well-defined because $\equiv$ is a congruence with respect to the corresponding operations on $\text{CTS}_\kappa$.

The structures $\mathfrak{P}_\kappa^\kappa$ are models of $\text{PA}_\kappa^\kappa$.

**Theorem 4.1 (Soundness).** For each infinite cardinal $\kappa$, we have $\mathfrak{P}_\kappa^\kappa \models \text{PA}_\kappa^\kappa$.

**Proof.** Because $\equiv$ is a congruence, it is sufficient to show that all axioms are sound. The soundness of all axioms follows easily from the definitions of the ingredients of $\mathfrak{P}_\kappa^\kappa$. \qed

The axioms of $\text{PA}_\kappa^\kappa$ constitute a complete axiomatization of the full splitting bisimulation models.

**Theorem 4.2 (Completeness).** Let $\kappa$ be an infinite cardinal. Then we have, for all closed terms $p$ and $q$ of sort $\mathbf{P}$ from the language of $\text{PA}_\kappa^\kappa$, $\mathfrak{P}_\kappa^\kappa \models p = q$ implies $\text{PA}_\kappa^\kappa \vdash p = q$.

**Proof.** By Lemma 2.2, for each closed term $p$ of sort $\mathbf{P}$ from the language of $\text{PA}_\kappa^\kappa$, there is a closed term $q$ of sort $\mathbf{P}$ from the language of $\text{BPA}_\kappa^\kappa$ such that $p = q$ is derivable from the axioms of $\text{PA}_\kappa^\kappa$; and by Lemma 2.3, all equations
between closed terms of sort \( P \) from the language of \( \text{BPA}^\delta_{cc} \) that can be derived from the axioms of \( \text{PA}^\delta_{cc} \) can be derived from the axioms of \( \text{BPA}^\delta_{cc} \). Therefore, it is sufficient to prove that the axioms of \( \text{BPA}^\delta_{cc} \) constitute a complete axiomatization of the restrictions of the full splitting bisimulation models of \( \text{PA}^\delta_{cc} \) to the constants and operators of \( \text{BPA}^\delta_{cc} \). In [9], a proof of completeness of the axioms of \( \text{BPA}^\tau \) for the graph models of \( \text{BPA}^\tau \) is given. That proof introduces a strongly normalising graph rewriting system with the following two properties: (i) two process graphs are bisimilar iff their normal forms are isomorphic and (ii) every rewriting step corresponds to a proof step in \( \text{BPA}^\tau \). The proof from [9] can easily be adapted to conditional transition systems, splitting bisimilarity and \( \text{BPA}^\delta_{cc} \).

The one-step reductions are in this case sharing of double states as in [9], and two types of joining of transitions: (a) replacing \( s \xrightarrow{[\alpha]} s'' \) and \( s \xrightarrow{[\beta]} s'' \) by \( s \xrightarrow{[\alpha \sqcup \beta]} s'' \) and (b) replacing \( s \xrightarrow{[\alpha]} \sqrt{\cdot} \) and \( s \xrightarrow{[\beta]} \sqrt{\cdot} \) by \( s \xrightarrow{[\alpha \sqcup \beta]} \sqrt{\cdot} \).

As to be expected, the full splitting bisimulation models are related by isomorphic embeddings.

**Theorem 4.3 (Isomorphic Embedding).** Let \( \kappa \) and \( \kappa' \) be infinite cardinals such that \( \kappa < \kappa' \). Then \( \mathbb{P}^\delta_{cc}^\kappa \) is isomorphically embedded in \( \mathbb{P}^\delta_{cc}^{\kappa'} \).

**Proof.** The proof is analogous to the proof of the corresponding property for the full splitting bisimulation models of \( \text{ACP}^c \) given in [11]. \( \Box \)

### 5 SOS-Based Splitting Bisimilarity for \( \text{PA}^\delta_{cc} \)

It is customary to associate transition systems with closed terms (of sort \( P \)) from the language of an ACP-like theory about processes by means of structural operational semantics and to identify closed terms if their associated transition systems are equivalent by a bisimilarity-based notion of equivalence.

The structural operational semantics of \( \text{PA}^\delta_{cc} \) presented in Section 2 determines a conditional transition system for each closed term of sort \( P \) from the language of \( \text{PA}^\delta_{cc} \). These transition systems are special in the sense that their states are closed terms of sort \( P \) from the language of \( \text{PA}^\delta_{cc} \).

Let \( p \) be a closed term of sort \( P \) from the language of \( \text{PA}^\delta_{cc} \). Then the transition system of \( p \) induced by the structural operational semantics of \( \text{PA}^\delta_{cc} \), written \( \text{CTS}(p) \), is the connected conditional transition system \( \Gamma(S, \rightarrow, \rightarrow\sqrt{\cdot}, s^0) \), where:

- \( S \) is the set of closed terms of sort \( P \) from the language of \( \text{PA}^\delta_{cc} \);
- \( \xrightarrow{(\alpha,a)} \subseteq S \times S \) and \( \xrightarrow{(\alpha,a)}\sqrt{\cdot} \subseteq S \) for each \( \alpha \in C \setminus \{\bot\} \) and \( a \in A \) are the smallest subsets of \( S \times S \) and \( S \), respectively, for which the transition rules from Table 3 hold;
- \( s^0 \in S \) is the closed term \( p \).

Let \( p \) and \( q \) be closed terms of sort \( P \) from the language of \( \text{PA}^\delta_{cc} \). Then we say that \( p \) and \( q \) are splitting bisimilar, written \( p \cong q \), if \( \text{CTS}(p) \cong \text{CTS}(q) \).
Clearly, the structural operational semantics does not give rise to infinitely branching conditional transition systems. For each closed term \( p \) of sort \( P \) from the language of \( \text{PA}_\delta^{cc} \), there exists a \( T \in \text{CTS}_{\aleph_0} \) such that \( \text{CTS}(p) \sim T \). In Section 4, it has been shown that it is possible to consider infinitely branching conditional transition systems as well.

6 ACP with Coordination Conditions

In order to support communication, we generalize the parallel composition operator of \( \text{PA}_\delta^{cc} \), resulting in \( \text{ACP}_\delta^{cc} \).

Just as in \( \text{PA}_\delta^{cc} \), it is assumed that a fixed but arbitrary finite set of actions \( A \), with \( \delta \notin A \) and \( t \in A \), has been given. In \( \text{ACP}_\delta^{cc} \), it is further assumed that a fixed but arbitrary commutative and associative communication function \( | : A_\delta \times A_\delta \rightarrow A_\delta \), such that \( \delta | a = \delta \) and \( t | a = \delta \) for all \( a \in A_\delta \), has been given. The function \(|\) is regarded to give the result of synchronously performing any two actions for which this is possible, and to be \( \delta \) otherwise.

The theory \( \text{ACP}_\delta^{cc} \) is an extension of \( \text{PA}_\delta^{cc} \). It has the constants and operators of \( \text{PA}_\delta^{cc} \) and in addition:

- the binary communication merge operator \(| : P \times P \rightarrow P|\);
- for each \( U \subseteq A \) with \( t \in U \), the unary enabledness update operator \( \Psi_U : C \rightarrow C \).

We use infix notation for the the communication merge operator as well.

Let \( p \) and \( q \) be closed terms of sort \( P \), \( \zeta \) be a closed term of sort \( C \), and \( U \subseteq A \) with \( t \in U \). Intuitively, the additional operators can be explained as follows:

- \( p | q \) behaves the same as \( p \parallel q \), except that it starts with performing an action of \( p \) and an action of \( q \) synchronously;
- \( \Psi_U(\zeta) \) is \( \zeta \) with all enabledness conditions \( E_a \) with \( a \notin U \) replaced by \( \bot \) and all enabledness conditions \( E_a \) with \( a \in U \) replaced by \( \top \).

The axioms of \( \text{ACP}_\delta^{cc} \) are the axioms of \( \text{PA}_\delta^{cc} \) with axioms M1–M4 replaced by axioms CM1–CM4, CME, C1–C3 and EU1–EU7 from Table 4. In axiom schema CME, the \( a_i, b_i, a'_j \) and \( b'_j \) stand for arbitrary action constants, and the \( \zeta_i, \xi_i, \zeta'_j \) and \( \xi'_j \) stand for arbitrary closed terms of sort \( C \). Moreover, for every \( U,V,W \subseteq A \) with \( t \in U,V,W \), \( \Phi_{U,V,W} \) iff for all \( u \in A \setminus \{t\} \):

\[
\begin{align*}
u \in V & \iff \exists j < n \ (\psi_W(\zeta'_j) = \top \land u | a'_j \in U) \\
& \lor \exists j < m \ (\psi_W(\zeta'_j) = \top \land u | b'_j \in U)
\end{align*}
\]

and

\[
\begin{align*}
u \in W & \iff \exists i < n \ (\psi_V(\xi_i) = \top \land u | a_i \in U) \\
& \lor \exists i < m \ (\psi_V(\xi'_i) = \top \land u | b_i \in U)
\end{align*}
\]

Axioms CM2–CM4 of \( \text{ACP}_\delta^{cc} \) are in fact the same as axioms M2–M4 of \( \text{PA}_\delta^{cc} \). The axioms of \( \text{ACP}_\delta^{cc} \) do not include axioms CM5–CM9 of \( \text{ACP} \) (see e.g. [8]), i.e.
Table 4. Axioms for $\text{ACP}^{\text{cc}}$ $(a, b, c \in A)$

<table>
<thead>
<tr>
<th>Expression</th>
<th>Axiom</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \parallel y = x \parallel y + x \parallel y$</td>
<td>CM1</td>
</tr>
<tr>
<td>$a \parallel x = a \cdot x$</td>
<td>CM2</td>
</tr>
<tr>
<td>$a \cdot x \parallel y = a \cdot (x \parallel y)$</td>
<td>CM3</td>
</tr>
<tr>
<td>$(x + y) \parallel z = x \parallel (z + y)$</td>
<td>CM4</td>
</tr>
<tr>
<td>$(\sum_{i&lt;n} \xi_i \rightarrow a_i \cdot x_i + \sum_{i&lt;m} \xi_i \rightarrow b_i) \parallel (\sum_{j&lt;i} \zeta_j \rightarrow a'_j \cdot x'<em>j + \sum</em>{j&lt;m'} \zeta'_j \rightarrow b'_j)$</td>
<td>CME</td>
</tr>
<tr>
<td>$a \mid b = b \mid a$</td>
<td>C1</td>
</tr>
<tr>
<td>$(a \mid b) \mid c = a \mid (b \mid c)$</td>
<td>C2</td>
</tr>
<tr>
<td>$\delta \mid a = \delta$</td>
<td>C3</td>
</tr>
</tbody>
</table>

The axioms of $\text{ACP}$ for communication merge, and axioms GC9–GC10 of $\text{ACP}^{\text{cc}}$ (see e.g. [11]), i.e. the additional axioms of $\text{ACP}^{\text{cc}}$ for communication merge.

CME, the axiom schema that replaces CM5–CM9 and GC9–GC10 in $\text{ACP}^{\text{cc}}$, needs further explanation. Consider processes $p$, $q$ and $p \mid q$. The process $p$ and the context in which $p \mid q$ is placed form parts of the context in which $q$ is placed; and the process $q$ and the context in which $p \mid q$ is placed form parts of the context in which $p$ is placed. Any subset of $A$ that includes $t$ may be the set of all actions that are enabled in the context in which $p \mid q$ is placed. Suppose that $U \subseteq A$ with $t \in U$ is the set of all actions that are enabled in the context in which $p \mid q$ is placed. Furthermore, suppose that $V \subseteq A$ with $t \in V$ and $W \subseteq A$ with $t \in W$ are the sets of all actions that are enabled in the contexts in which $p$ and $q$, respectively, are placed. Then the following must hold for all $a \in A \setminus \{t\}$:

- $a \in V$ iff $q$, with exactly the actions in $W$ enabled, can perform an action $b$ such that $a \mid b \in U$ (because $a$ is enabled in $q$ together with the context in which $p \mid q$ is placed);
- $a \in W$ iff $p$, with exactly the actions in $V$ enabled, can perform an action $b$ such that $a \mid b \in U$ (because $a$ is enabled in $p$ together with the context in which $p \mid q$ is placed).
V and W determine whether the conditions under which p and q can perform their initial actions evaluate to \( \top \) or \( \bot \). In the case of p, for example, the basis is that \( \mathcal{E}_a \) evaluates to \( \top \) if \( a \in V \), and \( \bot \) otherwise. All this is made precise in axiom schema CME. Notice that V and W need not exist for each U: the behaviour of p and q may inhibit proper mutual enabling of actions for \( p \mid q \). In such cases, \( p \mid q \) is considered to be incapable of doing anything. In other words, if V and W do not exist for some U, \( p \mid q \) is considered to behave the same as \( \delta \) in the event of U being the set of all actions that are enabled in the context in which \( p \mid q \) is placed.

We proceed with giving a few examples. In the examples, we take A such that \( A = A \cup \{ \pi \mid a \in A \} \cup \{ t \} \) for some set A. Moreover, we take \( \vdash : A_\delta \times A_\delta \to A_\delta \) such that, for all \( a \in A \) and \( b \in A \), \( a \mid \pi = t \), \( a \mid b = \delta \) if \( b \neq \pi \), \( \pi \mid b = \delta \) if \( b \neq a \), and \( t \mid b = \delta \). We start with giving an example of processes p and q of which the behaviour inhibits proper mutual enabling of actions for \( p \mid q \), whatever actions are enabled in the context in which \( p \mid q \) is placed. Consider the processes \( p \equiv \mathcal{E}_a : \Rightarrow b \) and \( q \equiv \mathcal{E}_b : \Rightarrow \pi \). The behaviour of these processes inhibits proper mutual enabling of actions for \( p \mid q \) whatever actions are enabled in the context in which \( p \mid q \) is placed: if a is enabled by q, then b is enabled by p and consequently a is not enabled by q; and if a is not enabled by q, then b is not enabled by p and consequently a is enabled by q. Hence, we have

\[
(\mathcal{E}_a : \Rightarrow b) \mid (\neg \mathcal{E}_b : \Rightarrow \pi) = \delta .
\]

We proceed with giving an example of processes p and q of which the behaviour does not inhibit proper mutual enabling of actions for \( p \mid q \), whatever actions are enabled in the context in which \( p \mid q \) is placed. Consider the processes \( p \equiv \mathcal{E}_a : \Rightarrow b + \tau \) and \( q \equiv \pi + \mathcal{E}_b : \Rightarrow c \). There is a unique proper mutual enabling, viz. the mutual enabling in which exactly the actions a and \( \tau \) are enabled by q, and exactly the actions \( \bar{b} \) and c are enabled by p. Hence, we have

\[
(\mathcal{E}_a : \Rightarrow b + \tau) \mid (\pi + \mathcal{E}_b : \Rightarrow c) = t .
\]

Finally, we give an example of processes p and q of which the behaviour is such that there are two proper mutual enablings of actions for \( p \mid q \), whatever actions are enabled in the context in which \( p \mid q \) is placed. Consider the processes \( p \equiv \mathcal{E}_a : \Rightarrow b + \neg \mathcal{E}_a : \Rightarrow c \) and \( q \equiv \mathcal{E}_b : \Rightarrow \pi + \neg \mathcal{E}_b : \Rightarrow \tau \). There are two proper mutual enablings, viz. the mutual enabling in which only the action a is enabled by q and only the action \( \bar{b} \) is enabled by p, and the mutual enabling in which only the action c is enabled by q and only the action \( \pi \) is enabled by p. Hence, we have

\[
(\mathcal{E}_a : \Rightarrow b + \neg \mathcal{E}_a : \Rightarrow c) \mid (\mathcal{E}_b : \Rightarrow \pi + \neg \mathcal{E}_b : \Rightarrow \tau) = \delta + t = t .
\]

Axioms similar to the axioms of ACP and ACP\(^c\) for communication merge are too much to expect for ACP\(^{2c}\): the mutual enabling of actions involved in the communication merge of two processes is a matter which can only be resolved by looking at the processes as a whole.

Like in the case of PA\(^c\), we can prove that all closed terms of sort P from the language of ACP\(^{2c}\) are derivably equal to a basic term.
**Lemma 6.1 (Elimination for ACP\textsuperscript{cc}).** For all closed terms p of sort P from the language of ACP\textsuperscript{cc}, there exists a basic term q ∈ B such that ACP\textsuperscript{cc} ⊢ p = q.

**Proof.** Let CME’ be CME with the conditions \( \prod_{i \in U} E_i \cap \prod_{i \in A \setminus U} -E_i \) moved inward (applying GC4) and combined with the conditions \( \Psi_V(\xi_i) \cap \Psi_W(\xi_j'), \Psi_V(\xi_i) \cap \Psi_W(\xi_j') \) and \( \Psi_V(\xi_i) \cap \Psi_W(\xi_j') \) (applying GC6). The term rewriting system consisting of all axioms of ACP\textsuperscript{cc}, except BA1–BA8 and A1–A3 and with CME replaced by CME’, oriented from left to right is strongly normalizing. This can be proved by using the method of lexicographical path ordering of Kamin and Lévy, making the signature one-sorted, ranking the operators \( \|, \| \) as in [9], taking the ordering \( \ldots > \|_3 > \|_3, \|_3 > \|_2 > \|_2, \|_2 > \vdash > \cdot > +, \vdash > \delta, \sqcap, \partial_H > \vdash, \sqcup, \sqcup \) for all \( H \subseteq A \setminus \{t\} \), \( t_I > :\vdash, t, \top, -, \sqcup \) for all \( I \subseteq A \), \( \|_2 > a, E_a, -, \psi_U \) for all \( a \in A \) and \( U \subseteq A \), and giving the lexicographical status for the first argument to \( \cdot \) and the lexicographical status for the second argument to \( :\vdash \). Moreover, it is easy to see that each normal form is a basic term.

We can also prove that ACP\textsuperscript{cc} is a conservative extension of BPA\textsubscript{\( \delta \)}.

**Lemma 6.2 (Conservative extension).** If p and q are closed terms of sort P from the language of BPA\textsubscript{\( \delta \)}, then BPA\textsubscript{\( \delta \)} ⊢ p = q iff ACP\textsuperscript{cc} ⊢ p = q.

**Proof.** The proof is analogous to the proof of Lemma 2.3.

The preceding two lemmas will be useful in the completeness proof of ACP\textsuperscript{cc} for the full splitting bisimulation models of ACP\textsuperscript{cc} that will be introduced below.

Associating transition systems with closed terms of sort P from the language of ACP\textsuperscript{cc} in the style of structural operational semantics looks to be impossible. The problem is to define by means of transition rules, for every \( U, V, W \) of ACP\textsuperscript{cc} with the operator \( \| \), \( \| \) have to be adapted and an operation on \( \text{CTS}_\kappa / = \) associated with the operator \( \| \) has to be added.

Like before, we begin by associating operations \( \parallel^\prime \), \( \parallel^\prime \) and \( \parallel^\prime \) on \( \text{CTS}_\kappa \) with the operators \( \|, \| \) and \( \| \).

- Let \( T_i = (S_i, \rightarrow_i, -\rightarrow_i, s^0_i) \in \text{CTS}_\kappa \) for \( i = 1, 2 \). Then
  \[ T_1 \parallel^\prime T_2 = \Gamma(S, \rightarrow, -\rightarrow, s^0) \],
  where
  \[ s^0 = (s^0_1, s^0_2) \],
  \[ s^\prime = ch_\kappa(S_\kappa \setminus (S_1 \cup S_2)) \],
  \[ S = ((S_1 \cup \{s^\prime\}) \times (S_2 \cup \{s^\prime\})) \setminus \{(s^\prime, s^\prime)\} \].

7 Full Splitting Bisimulation Models of ACP\textsuperscript{cc}

In this section, we adapt the full splitting bisimulation models of PA\textsuperscript{cc} to ACP\textsuperscript{cc}. In order to cover communication, the operations on \( \text{CTS}_\kappa / = = \) associated with the operators \( \| \) and \( \| \) have to be adapted and an operation on \( \text{CTS}_\kappa / = = \) associated with the operator \( \| \) has to be added.

Like before, we begin by associating operations \( \parallel^\prime \), \( \parallel^\prime \) and \( \parallel^\prime \) on \( \text{CTS}_\kappa \) with the operators \( \|, \| \) and \( \| \).

- Let \( T_i = (S_i, \rightarrow_i, -\rightarrow_i, s^0_i) \in \text{CTS}_\kappa \) for \( i = 1, 2 \). Then
  \[ T_1 \parallel^\prime T_2 = \Gamma(S, \rightarrow, -\rightarrow, s^0) \],
  where
  \[ s^0 = (s^0_1, s^0_2) \],
  \[ s^\prime = ch_\kappa(S_\kappa \setminus (S_1 \cup S_2)) \],
  \[ S = ((S_1 \cup \{s^\prime\}) \times (S_2 \cup \{s^\prime\})) \setminus \{(s^\prime, s^\prime)\} \].

21
and for every \((\alpha, a) \in C^- \times A\):

\[
\frac{(\alpha, a)}{\Phi_{\mathcal{U}, \mathcal{V}, \mathcal{W}}(s_1, s_2)} = \{(s_1, s_2), (s_1', s_2') \mid (s_1', s_2') \in S \land s_1 \xrightarrow{[\alpha, a]}_{t_1} s_1' \land s_2' \}
\]

\[
\bigcup \{(s_1, s_2), (s_1', s_2') \mid (s_1', s_2') \in S \land s_2 \xrightarrow{[\alpha, a]}_{t_2} s_2' \land s_1' \}
\]

\[
\bigcup \{(s_1, s_2), (s_1', s_2') \mid (s_1', s_2') \in S \land \bigvee_{a' \in U \cup V \cup W} \Phi_{\mathcal{U}, \mathcal{V}, \mathcal{W}}(s_1, s_2) \land \neg \mathcal{E}_u \cap \bigcap \mathcal{E}_{u \in A \setminus U} - \mathcal{E}_u = \alpha \land \Psi_V(\alpha') \cap \Psi_W(\beta') = \top \land a' | b' = a\}
\]

and for every \(U, V, W \subseteq A\) with \(t \in U, V, W\) and for every \((s_1, s_2) \in S\), \(\Phi_{\mathcal{U}, \mathcal{V}, \mathcal{W}}(s_1, s_2)\) iff for all \(u \in A \setminus \{t\}\):

\[
u \in V \iff \exists \alpha'', a'', s'' \xrightarrow{[\alpha'', a'']}_{t_2} s'' \land \Psi_W(\alpha'') = \top \land u | a'' \in U
\]

\[
\lor \exists \alpha'', a'' \xrightarrow{[\alpha'', a'']}_{t_2} \top \land \Psi_W(\alpha''') = \top \land u | a'' \in U
\]

and

\[
u \in W \iff \exists \alpha'', a'', s'' \xrightarrow{[\alpha'', a'']}_{t_1} s'' \land \Psi_V(\alpha''') = \top \land u | a'' \in U
\]

\[
\lor \exists \alpha'', a'' \xrightarrow{[\alpha'', a'']}_{t_1} \top \land \Psi_V(\alpha''') = \top \land a'' | u \in U
\]

Let \(T_i = (S_i, t_i, \rightarrow, \sqrt{\_}, s_i^0) \in \text{CTS}_k\) for \(i = 1, 2\). Suppose that \(T_1 \parallel T_2 = (S, t_1, \rightarrow, \sqrt{\_}, s^0)\) where \(S = \left(\{s_i^1 \cup \{s'\} \times \{s_2 \cup \{s\} \} \setminus \{s', s\}\}\right)\) and \(s'' = \text{ch}(S_k \setminus (S_1 \cup S_2))\). Then

\(T_1 \parallel T_2 = \Gamma(S', t', \rightarrow, \sqrt{s}, s'')\)

where

\(s'' = \text{ch}(S_k \setminus S)\),

\(S' = \{s''\} \cup S\),

and for every \((\alpha, a) \in C^- \times A\):

\[
\frac{(\alpha, a)}{s''} = \{\{(s''', (s, s_2')) \mid s''' \xrightarrow{[\alpha, a]}_{t_1} s \land s'\} \cup \{(\alpha, a)\}
\]

22
Let \( T_i = (S_i, \rightarrow_i, \rightarrow_i, s_i^0) \) \( \in \mathbb{CTS}_\kappa \) for \( i = 1, 2 \). Suppose that \( T_1 \parallel T_2 = (S, \rightarrow, \rightarrow, s^0) \) where \( S = ((S_1 \cup \{s^\prime\}) \times (S_2 \cup \{s^\prime\})) \setminus \{(s^\prime, s^\prime)\} \) and \( s^\prime = \text{ch}_\kappa(S_\kappa \setminus (S_1 \cup S_2)) \). Then
\[
T_1 \parallel T_2 = \Gamma(S', \rightarrow', \rightarrow', s^0'),
\]
where
\[
s^0' = \text{ch}_\kappa(S_\kappa \setminus S),
\]
\[
S' = \{s^0'\} \cup S,
\]
and for every \( (\alpha, a) \in C^- \times A \):
\[
\frac{(s^0, (s_1, s_2))}{\frac{\phi'_{U,V,W}(s_1^0, s_2^0)}{\phi'_{U,V,W}(s_1^0, s_2^0)}}
\]
\[
\frac{\phi'_{U,V,W}(s_1^0, s_2^0)}{\phi'_{U,V,W}(s_1^0, s_2^0)}
\]
\[
\frac{\phi'_{U,V,W}(s_1^0, s_2^0)}{\phi'_{U,V,W}(s_1^0, s_2^0)}
\]
\[
\phi'_{U,V,W}(s_1^0, s_2^0)
\]
and \( \Phi'_{U,V,W}(s_1, s_2) \) is as in the definition of \( \parallel' \).

We can show that splitting bisimilarity is a congruence with respect to parallel composition, left merge and communication merge.

**Lemma 7.1 (Congruence).** Let \( \kappa \) be an infinite cardinal. Then for all \( T_1, T_2, T'_1, T'_2 \in \mathbb{CTS}_\kappa \), \( T_1 \parallel T'_1 \) and \( T_2 \parallel T'_2 \) imply \( T_1 \parallel T_2 \parallel T'_1 \parallel T'_2 \), \( T_1 \parallel T'_2 \parallel T'_1 \parallel T'_2 \) and \( T_1 \parallel T_2 \parallel T'_1 \parallel T'_2 \).

**Proof.** Although parallel composition and left merge as considered in the setting of \( \text{ACP}^\text{cc} \) differs from parallel composition and left merge as considered in the setting of \( \text{PA}^\text{cc} \) \( \delta \), witnessing splitting bisimulations can be constructed in the same way as in the proof of Lemma 4.1. For communication merge, the witnessing splitting bisimulation is constructed like for left merge. It is straightforward to show that the constructed relations are splitting bisimulations indeed. However, it is not so easy as in the proof of Lemma 4.1. The most important complication is that we have to verify whether the constructed relation, say \( R \), has the following property: \( R((s_1, s_2), (s'_1, s'_2)) \) implies \( \Phi'_{U,V,W}(s_1, s_2) \) iff \( \Phi'_{U,V,W}(s'_1, s'_2) \) for all \( U, V, W \subseteq A \) with \( t \in U, V, W \). \( \square \)
The full splitting bisimulation models $P^{cc}_\kappa$ of ACP$^{cc}$, one for each infinite cardinal $\kappa$, are the full splitting bisimulation models $P^{cc}_\kappa$ of PA$^{cc}_\delta$ with adapted operations $\parallel'$ and $\lceil \rceil'$ on CTS$_{\kappa/\supseteq}$ for the operators $\parallel$ and $\lceil \rceil$, and an additional operation $\lceil \rceil'$ on CTS$_{\kappa/\supseteq}$ for the operator $\mid$. Those operations are defined as follows:

\begin{align*}
[T_1]_{\supseteq} \parallel' [T_2]_{\supseteq} &= [T_1 \parallel' T_2]_{\supseteq}, \\
[T_1]_{\supseteq} \lceil [T_2]_{\supseteq} &= [T_1 \lceil T_2]_{\supseteq}, \\
[T_1]_{\supseteq} \lceil' [T_2]_{\supseteq} &= [T_1 \lceil' T_2]_{\supseteq}.
\end{align*}

Parallel composition, left merge and communication merge on CTS$_{\kappa/\supseteq}$ are well-defined because $\supseteq$ is a congruence with respect to the corresponding operations on CTS$_\kappa$.

The structures $P^{cc}_\kappa$ are models of ACP$^{cc}$.

**Theorem 7.1 (Soundness).** For each infinite cardinal $\kappa$, we have $P^{cc}_\kappa \vDash ACP^{cc}$.

**Proof.** Because $\supseteq$ is a congruence, and all changes with respect to $P^{cc}_\kappa$ concern the operations associated with the operators $\parallel$, $\lceil \rceil$ and $\mid$, it is sufficient to show that all axioms from Table 4 are sound. The soundness of all those axioms follows easily from the definitions of the ingredients of $P^{cc}_\kappa$. $\qed$

The axioms of ACP$^{cc}$ constitute a complete axiomatization of the full splitting bisimulation models.

**Theorem 7.2 (Completeness).** Let $\kappa$ be an infinite cardinal. Then we have, for all closed terms $p$ and $q$ of sort $P$ from the language of ACP$^{cc}$, $P^{cc}_\kappa \vDash p = q$ implies ACP$^{cc} \vdash p = q$.

**Proof.** By Lemma 6.1, for each closed term $p$ of sort $P$ from the language of ACP$^{cc}$, there is a closed term $q$ of sort $P$ from the language of BPA$^{cc}_\delta$ such that $p = q$ is derivable from the axioms of ACP$^{cc}$; and by Lemma 6.2, all equations between closed terms of sort $P$ from the language of BPA$^{cc}_\delta$ that can be derived from the axioms of ACP$^{cc}$ can be derived from the axioms of BPA$^{cc}_\delta$. Therefore, it is sufficient to prove that the axioms of BPA$^{cc}_\delta$ constitute a complete axiomatization of the restrictions of the full splitting bisimulation models of ACP$^{cc}$ to the constants and operators of BPA$^{cc}_\delta$. In the proof of Theorem 4.2, it is shown that the axioms of BPA$^{cc}_\delta$ constitute a complete axiomatization of the restrictions of the full splitting bisimulation models of PA$^{cc}_\delta$ to the constants and operators of BPA$^{cc}_\delta$. Because the restrictions of the full splitting bisimulation models of PA$^{cc}_\delta$ to the constants and operators of BPA$^{cc}_\delta$ coincide with the restrictions of the full splitting bisimulation models of ACP$^{cc}$ to the constants and operators of BPA$^{cc}_\delta$, the proof is completed. $\Box$

It is easy to see that Theorem 4.3 goes through for $P^{cc}_\kappa$.

In this section, the full splitting bisimulation models $P^{cc}_\kappa$ of PA$^{cc}_\delta$ have been expanded to obtain the full splitting bisimulation models $P^{cc}_\kappa$ of ACP$^{cc}$. Henceforth, we will loosely write $P^{cc}_\kappa$ for $P^{cc}_\kappa$. 24
Table 5. Axioms for recursion

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle X \mid E \rangle = \langle tX \mid E \rangle</td>
<td>if \ X = tX \in E</td>
</tr>
<tr>
<td>E \Rightarrow X = \langle X \mid E \rangle</td>
<td>if \ X \in V(E)</td>
</tr>
</tbody>
</table>

8 Guarded Recursion

In order to allow for the description of (potentially) non-terminating processes, we add guarded recursion to \( \text{ACP}^{cc} \).

A recursive specification over \( \text{ACP}^{cc} \) is a set of recursive equations \( E = \{ X = tX \mid X \in V \} \) where \( V \) is a set of variables and each \( tX \) is a term of sort \( P \) from the language of \( \text{ACP}^{cc} \) that only contains variables from \( V \). We write \( V(E) \) for the set of all variables that occur on the left-hand side of an equation in \( E \). A solution of a recursive specification \( E \) is a set of processes (in some model of \( \text{ACP}^{cc} \)) \( \{ P_X \mid X \in V(E) \} \) such that the equations of \( E \) hold if, for all \( X \in V(E) \), \( X \) stands for \( P_X \).

Let \( t \) be a term of sort \( P \) from the language of \( \text{ACP}^{cc} \) containing a variable \( X \). We call an occurrence of \( X \) in \( t \) guarded if \( t \) has a subterm of the form \( a \cdot t' \) containing this occurrence of \( X \). A recursive specification over \( \text{ACP}^{cc} \) is called a guarded recursive specification if all occurrences of variables in the right-hand sides of its equations are guarded or it can be rewritten to such a recursive specification using the axioms of \( \text{ACP}^{cc} \) and the equations of the recursive specification. We are only interested in models of \( \text{ACP}^{cc} \) in which guarded recursive specifications have unique solutions.

For each guarded recursive specification \( E \) and each variable \( X \in V(E) \), we introduce a constant of sort \( P \) standing for the unique solution of \( E \) for \( X \). This constant is denoted by \( \langle X \mid E \rangle \). We often write \( X \) for \( \langle X \mid E \rangle \) if \( E \) is clear from the context. In such cases, it should also be clear from the context that we use \( X \) as a constant.

We will also use the following notation. Let \( t \) be a term of sort \( P \) from the language of \( \text{ACP}^{cc} \) and \( E \) be a guarded recursive specification over \( \text{ACP}^{cc} \). Then we write \( \langle t \mid E \rangle \) for \( t \) with, for all \( X \in V(E) \), all occurrences of \( X \) in \( t \) replaced by \( \langle X \mid E \rangle \).

The additional axioms for recursion are given in Table 5. Both RDP and RSP are axiom schemas. Side conditions are added to restrict the variables, terms and guarded recursive specifications for which \( X \), \( tX \) and \( E \) stand. The additional axioms for recursion are known as the recursive definition principle (RDP) and the recursive specification principle (RSP). The equations \( \langle X \mid E \rangle = \langle tX \mid E \rangle \) for a fixed \( E \) express that the constants \( \langle X \mid E \rangle \) make up a solution of \( E \). The conditional equations \( E \Rightarrow X = \langle X \mid E \rangle \) express that this solution is the only one.

In the full splitting bisimulation models of \( \text{ACP}^{cc} \), guarded recursive specifications over \( \text{ACP}^{cc} \) have unique solutions.

Theorem 8.1 (Unique solutions). For each infinite cardinal \( \kappa \), guarded recursive specifications over \( \text{ACP}^{cc} \) have unique solutions in \( \text{ACP}^{cc} \).
Table 6. Axioms for preferential choice ($a \in A$)

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta \leftrightarrow x = x$</td>
<td>PC1</td>
</tr>
<tr>
<td>$a \cdot x \leftrightarrow y = a \cdot x + -E_a \rightarrow y$</td>
<td>PC2</td>
</tr>
<tr>
<td>$(x + y) \leftrightarrow z = x \leftrightarrow (y \leftrightarrow z) + y \leftrightarrow (x \leftrightarrow z)$</td>
<td>PC3</td>
</tr>
<tr>
<td>$(\phi \rightarrow x) \leftrightarrow y = \phi \rightarrow (x \leftrightarrow y) + -\phi \rightarrow y$</td>
<td>PC4</td>
</tr>
</tbody>
</table>

Proof. The proof is analogous to the proof of the corresponding property for the full splitting bisimulation models of ACP$^c$ given in [11].

Thus, the full splitting bisimulation models $\mathcal{P}_c''$ of ACP$^{cc}$ with guarded recursion are simply the expansions of the full splitting bisimulation models $\mathcal{P}_c$ of ACP$^{cc}$ obtained by associating with each constant $\langle X|E \rangle$ the unique solution of $E$ for $X$ in the full splitting bisimulation model concerned.

9 Preferential Choice

In the presence of conditional expressions of which the conditions concern the enabledness of actions in the context in which a process is placed, it is easy to give defining equations for a preferential choice operator. In this section, we extend ACP$^{cc}$ with the binary preferential choice operator $+: P \times P \rightarrow P$.

Let $p$ and $q$ be closed terms of sort $P$. Intuitively, the preferential choice operator can be explained as follows:

- $p + \rightarrow q$ behaves as $p$ if the context in which it is placed permits it to behave as $p$, and as $q$ otherwise.

The additional axioms for preferential choice are the axioms given in Table 6. From the axioms of ACP$^{cc}$ and axioms PC1–PC4, we can easily derive the equation $(\phi \rightarrow a \cdot x + \psi \rightarrow b \cdot y) \rightarrow z = \phi \rightarrow a \cdot x + \psi \rightarrow b \cdot y + -((\phi \cap E_a) \cup (\psi \cap E_b)) \rightarrow z$.

The following generalization of this result gives a full picture of the preferential choice operator.

Proposition 9.1 (Characterization). From the axioms of ACP$^{cc}$ and axioms PC1–PC4, the following is derivable for all $n, m \geq 0$, for all $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1} \in A$:

\[
\left( \sum_{i<n} \phi_i \rightarrow a_i \cdot x_i + \sum_{i<m} \psi_i \rightarrow b_i \right) \rightarrow y = \\
\sum_{i<n} \phi_i \rightarrow a_i \cdot x_i + \sum_{i<m} \psi_i \rightarrow b_i \\
+ - \left( \bigcup_{i<n} (\phi_i \cap E_{a_i}) \cup \bigcup_{i<m} (\psi_i \cap E_{b_i}) \right) \rightarrow y .
\]

Proof. Easy, by induction on $n + m$. 

A corollary of Proposition 9.1 is that the preferential choice operator is associative.
Corollary 9.1  (Associativity). For all closed terms $p$, $q$, $r$ of sort $\mathbf{P}$ from the language of $\text{ACP}^{cc}$ extended with preferential choice, $p \vdash (q \vdash r) = (p \vdash q) \vdash r$ is derivable from the axioms of $\text{ACP}^{cc}$ and axioms $\text{PC1–PC4}$.

Another corollary of Proposition 9.1 is that all occurrences of the preferential choice operator can be eliminated from closed terms.

Corollary 9.2  (Elimination). For all closed terms $p$ of sort $\mathbf{P}$ from the language of $\text{ACP}^{cc}$ extended with preferential choice, there exists a closed term $q$ of sort $\mathbf{P}$ from the language of $\text{ACP}^{cc}$ such that $p = q$ is derivable from the axioms of $\text{ACP}^{cc}$ and axioms $\text{PC1–PC4}$.

The full bisimulation models of $\text{ACP}^{cc}$ with preferential choice are the expansions of the full bisimulation models $\text{P}^{cc}$ of $\text{ACP}^{cc}$ obtained by first associating with the operator $\rightarrow$ a corresponding operation on $\text{CTS}_\kappa$ and then lifting the result of this to $\text{CTS}_\kappa/\equiv$. This calls for extraction of the initial guarded actions of a conditional transition system.

Let $T = (S, -\rightarrow, -\rightarrow\sqrt{\cdot}, s^0) \in \text{CTS}_\kappa$. Then the initial guarded actions of $T$, written $I(T)$, is the set $\{(\alpha, a) \in C^- \times A \mid \exists s \in S \cdot s^0 \overset{[\alpha]a}{\rightarrow} s \lor s^0 \overset{[\alpha]a}{\rightarrow} \sqrt{\cdot}\}$.

We proceed with associating with the operator $\rightarrow$ an operation $\hat{\rightarrow}$ on $\text{CTS}_\kappa$ as follows.

- Let $T_i = (S_i, -\rightarrow, -\rightarrow\sqrt{\cdot}, s_i^0) \in \text{CTS}_\kappa$ for $i = 1, 2$. Then

\[ T_1 \hat{\rightarrow} T_2 = \Gamma(S, -\rightarrow, -\rightarrow\sqrt{\cdot}, s^0), \]

where

\[ s^0 = ch_\kappa(S_\kappa \setminus (S_1 \uplus S_2)), \]

\[ S = \{s^0\} \cup (S_1 \uplus S_2), \]

and for every $(\alpha, a) \in C^- \times A$:

\[ \frac{\alpha = \bigcup (\alpha', a') \in I(T_1) (\alpha' \cap E_{a'})) \cap \beta}{(\alpha, a)} \]

\[ = \{s^0, \mu_1(s)\} \mid s^0 \overset{[\alpha]a}{\rightarrow} s \}
\]

\[ \cup \{s^0, \mu_2(s)\} \mid \exists \beta \cdot s^0 \overset{[\beta]a}{\rightarrow} s \land \alpha = -\bigcup (\alpha', a') \in I(T_1) (\alpha' \cap E_{a'})) \cap \beta\}
\]

\[ \cup \{\mu_1(s), \mu_1(s')\} \mid s \overset{[\alpha]a}{\rightarrow} s'\}
\]

\[ \cup \{\mu_2(s), \mu_2(s')\} \mid s \overset{[\alpha]a}{\rightarrow} s'\}, \]

\[ \frac{\alpha = \bigcup (\alpha', a') \in I(T_1) (\alpha' \cap E_{a'})) \cap \beta}{(\alpha, a)\sqrt{\cdot}} \]

\[ = \{s^0 \mid s_1^0 \overset{[\alpha]a}{\rightarrow} \sqrt{1}\}
\]

\[ \cup \{s^0 \mid \exists \beta \cdot s_2^0 \overset{[\alpha]a}{\rightarrow} \sqrt{2} \land \alpha = -\bigcup (\alpha', a') \in I(T_1) (\alpha' \cap E_{a'})) \cap \beta\}
\]

\[ \cup \{\mu_1(s) \mid s \overset{[\alpha]a}{\rightarrow} \sqrt{1}\}
\]

\[ \cup \{\mu_2(s) \mid s \overset{[\alpha]a}{\rightarrow} \sqrt{2}\}. \]

27
We can show that splitting bisimilarity is a congruence with respect to preferential choice.

**Lemma 9.1 (Congruence).** Let \( \kappa \) be an infinite cardinal. Then for all \( T_1, T_2, T_1', T_2' \in \text{CTS}_\kappa \), \( T_1 \simeq T_1' \) and \( T_2 \simeq T_2' \) imply \( T_1 \lhd T_2 \simeq T_1' \lhd T_2' \).

**Proof.** Although preferential choice differs from alternative composition (being a non-preferential choice), a witnessing splitting bisimulation can be constructed in the same way as in the proof of Lemma 4.1. It is straightforward to show that the constructed relation is a splitting bisimulation indeed. As compared with alternative composition, all we have to do more is to show that \( T_1 \simeq T_1' \) implies \( \bigcup_{(\alpha', \alpha') \in \mathcal{I}(T_1)} (\alpha' \cap \mathcal{E}_d) = \bigcup_{(\alpha', \alpha') \in \mathcal{I}(T_1')} (\alpha' \cap \mathcal{E}_d) \).

The operation \( \lhd \) on \( \text{CTS}_\kappa \) is defined as follows:

\[
[T_1]_{\simeq} \lhd [T_2]_{\simeq} = [T_1 \lhd T_2]_{\simeq}.
\]

## 10 Example

In this section, we give an example of the use of the preferential choice operator. The example is adapted from [4]. The example concerns a printer with a control panel. The printer will print an infinite sequence \( \langle c_0, c_1, c_2, \ldots \rangle \) of characters from a finite set of characters \( C \), but can be interrupted by means of the control panel. The control panel has two buttons, a start button to indicate that the printing must be started and a stop button to indicate that the printing must be stopped. Whenever a button is pushed, a corresponding message is sent to the printer. The printer prints characters from the infinite sequence, but can also receive messages from the control panel. The recursive specification of the control panel is as follows:

\[
C = b(\text{start}) \cdot s(\text{start}) \cdot C + b(\text{stop}) \cdot s(\text{stop}) \cdot C.
\]

and the recursive specification of the printer is as follows (\( i \geq 0 \)):

\[
P = W_0
\]

\[
W_i = r(\text{start}) \cdot P_i + r(\text{stop}) \cdot W_i
\]

\[
P_i = (r(\text{start}) \cdot P_i + r(\text{stop}) \cdot W_i) \vdash p(c_i) \cdot P_{i+1}.
\]

In this example, we take \( | : A_\delta \times A_\delta \rightarrow A_\delta \) such that \( s(\text{start}) \parallel r(\text{start}) = c(\text{start}) \) and \( s(\text{stop}) \parallel r(\text{stop}) = c(\text{stop}) \), and in all other cases it yields \( \delta \). The whole system is described by the \( \partial_H(C \parallel P) \), where \( H = \{ s(\text{start}), r(\text{start}), s(\text{stop}), r(\text{stop}) \} \).

In the recursive specification of the printer, the preferential choice operator is used in the equation for \( P_i \) to describe that receiving a message from the control panel must be given preference to printing a character. It follows from the axioms for preferential choice that the preferential choice operator can be eliminated from this equation. This yields the following equation:

\[
P_i = r(\text{start}) \cdot P_i + r(\text{stop}) \cdot W_i + - (\mathcal{E}_r(\text{start}) \parallel \mathcal{E}_r(\text{stop})) \vdash p(c_i) \cdot P_{i+1}.
\]

28
From the axioms of ACP\textsuperscript{cc}, the additional axioms for preferential choice and RSP, we can derive that \( \partial_H(C \parallel P) \) is the solution of the following recursive specification \((i \geq 0)\):

\[
S = S^w_0 \\
S^w_i = b(start) \cdot c(start) \cdot S^p_i + b(stop) \cdot c(stop) \cdot S^w_i \\
S^p_i = b(start) \cdot c(start) \cdot S^p_i + b(stop) \cdot c(stop) \cdot S^w_i + p(c_i) \cdot S^p_{i+1}.
\]

This shows that the reaction to button pushing is immediate: printing of characters never takes place between button pushing and the reaction to button pushing.

### 11 Concluding Remarks

We have presented ACP\textsuperscript{cc}, an extension of ACP with conditional expressions of which the conditions concern the enabledness of actions in the context in which a process is placed, as well as the main models of ACP\textsuperscript{cc}. To the best of our knowledge, there is no other work in the field of process algebra studying conditions of this kind. However, there are several extensions of ACP that include conditional expressions of some kind. As a matter of fact, ACP\textsuperscript{cc} is a variant of a recent extension of ACP with conditional expressions called ACP\textsuperscript{c} [11, 12]. Earlier extensions of ACP that include conditional expressions can be found in [6, 2, 14, 3].

A striking point of ACP\textsuperscript{cc} is that its axioms do not include axioms similar to the axioms of ACP for communication merge (axioms CM5–CM9, see e.g. [7]). Such axioms are too much to expect: the mutual enabling of actions involved in the communication merge of two processes is a matter which can only be resolved by looking at the processes as a whole. The same need of a global approach makes structural operational semantics unsuited for associating transition systems with closed terms.

We do not have a clear notion of the applications of ACP\textsuperscript{cc}. We have treated an application of ACP\textsuperscript{cc} that remains within the domain of process algebra: the extension of ACP\textsuperscript{cc} with a preferential choice operator. This operator appears to generalize the priority choice operator added to CCS in [16].

ACP\textsuperscript{cc} includes pre-abstraction, but not abstraction. Abstraction is usually based on observation equivalence [21] or branching bisimulation equivalence [23], which both abstract from both the structure of finitary internal activity and its presence. Abstraction from its presence is not considered to be well-suited to the setting of ACP\textsuperscript{cc}. Orthogonal bisimulation equivalence [13], an equivalence introduced recently, abstracts from the structure of finitary internal activity, but not from its presence. Therefore, this equivalence looks to be better suited to the setting of ACP\textsuperscript{cc}.

One option for future work is development of an extension of ACP\textsuperscript{cc} with abstraction based on orthogonal bisimulation equivalence. Another option for
future work is investigation into ways to combine the kind of conditions considered in $\text{ACP}^{cc}$ with other kinds of conditions, in particular with the retrospective conditions of $\text{ACP}^{ct}$ [11].

References


