Splitting bisimulations and retrospective conditions

Bergstra, J.A.; Middelburg, K.

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 author’s 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

Citation for published version (APA):

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.

Take down policy
If you believe that this document breaches copyright please contact us (openaccess@tue.nl) providing details. We will immediately remove access to the work pending the investigation of your claim.

Download date: 25. Jan. 2019
Splitting Bisimulations and Retrospective Conditions

J.A. Bergstra\(^1\),\(^2\) and C.A. Middelburg\(^3\)

\(^1\) Programming Research Group, University of Amsterdam, P.O. Box 41882, 1009 DB Amsterdam, the Netherlands
\texttt{janb@science.uva.nl}

\(^2\) Department of Philosophy, Utrecht University, P.O. Box 80126, 3508 TC Utrecht, the Netherlands
\texttt{janb@phil.uu.nl}

\(^3\) Computing Science Department, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands
\texttt{keesm@win.tue.nl}

Abstract. We investigate conditions in the setting of the algebraic theory about processes known as ACP. We present ACP\(^c\), an extension of ACP with guarded commands, and its main models, called its full splitting bisimulation models. The conditions used in the guarded commands are taken from a Boolean algebra. We add two operators for condition evaluation to ACP\(^c\); and study their connection with alternative mechanisms found in other extensions of ACP with guarded commands, to wit state operators and signal emission. On purpose to incorporate the past in conditions, we add a retrospection operator on conditions to ACP\(^c\).

Keywords: conditional transition systems, splitting bisimulation, guarded commands, retrospective conditions, condition evaluation, state operators, signal emission, process algebra, Boolean algebras.


1 Introduction

Many theories about processes include conditional expressions of some form. Several extensions of the algebraic theory about processes known as ACP \cite{8,7} include conditional expressions of the form \(\zeta \rightarrow p \text{ or } p \triangleleft \zeta \triangleright q\); see e.g. \cite{6,2,13,3}. What is considered to be conditions and how they are evaluated differs from one extension to another. The set of conditions is usually one of the following:

- a two-valued set, usually called \(\mathbb{B}\);
- the set of all propositions with a given set of propositional variables and with finite conjunctions and disjunctions;
- the domain of a free Boolean algebra over a given set of generators.

The last alternative generalized both other alternatives. In this paper, we will focus our attention on the last alternative and implicate the other alternatives where appropriate for explanation or motivation.
We introduce ACP$^c$, an extension of ACP with conditional expressions of the form $\zeta \rightarrow p$ in which the set of conditions is the domain of a free Boolean algebra over a given set of generators. We present the main models of ACP$^c$, which 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.

Existing mechanisms that allow for a kind of condition evaluation in conditional expressions include the state operators as introduced in [2] and signal emission as introduced in [3]. However, those mechanisms were not devised for that purpose. We extend ACP$^c$ with operators devised for condition evaluation, with state operators, and with signal emission; and show how those extensions are related. The set of signals is the same as the set of conditions. For the main models of ACP$^{cs}$, the extension of ACP$^c$ with signal emission, generalizations of conditional transition systems and splitting bisimilarity are introduced.

Two kinds of operators are devised for condition evaluation, one for the case where condition evaluation is not dependent on process behaviour and the other for the case where condition evaluation is dependent on process behaviour. We show how a theory about the set of atomic conditions can be used for condition evaluation with an operator of the former kind, that the operators of the former kind are superseded by the operators of the latter kind and that those operators are in their turn superseded by the state operators. We also show that the signal emission operator corresponds to a local form of condition evaluation: unlike the forms of condition evaluation covered by the operators mentioned above, condition evaluation by means of the signal emission operator does not persist over performing an action.

We also extend ACP$^c$ with a retrospection operator on conditions, which allows for looking back on conditions under which preceding actions have been performed. For the main models of ACP$^{cr}$, the extension of ACP$^c$ with the retrospection operator, an adaptation of splitting bisimilarity is introduced. We extend ACP$^{cr}$ with the above-mentioned operators devised for condition evaluation as well.

The work presented in this paper, can easily be adapted to other process algebras based on (strong) bisimulation models, such as the strong bisimulation version of CCS [19]. Adaptation to CSP [14], which is not based on bisimulation models, will be more difficult and in part perhaps even impossible.

The structure of this paper is as follows. First of all, we introduce BPA$^c$, an important subtheory of ACP$^c$ that does not support parallelism and 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 BPA$^c$, the main models of BPA$^c$ (Section 4). Following this, we have a closer look at splitting bisimilarity based on structural operational semantics (Section 5). Next, we extend BPA$^c$ to ACP$^c$ (Section 6) and expand the full splitting bisimulation models of BPA$^c$ to full splitting...
bisimulation models of ACP^c (Section 7). Then, we extend ACP^c with guarded recursion (Section 8). Thereupon, we extend ACP^c with condition evaluation operators (Section 9), with state operators (Section 10) and with a signal emission operator (Section 11); and analyse how those operators are related. We also adapt the full splitting bisimulation models of ACP^c to the full signal-observing splitting bisimulation models of ACP^{cs}, the extension of ACP^c with signal emission (Section 12). After that, we extend BPA^δ with a retrospection operator (Section 13) and adapt the full splitting bisimulation models of BPA^δ to the full retrospective splitting bisimulation models of BPA^{crδ}, the extension of BPA^δ with retrospection (Section 14). Next, we extend BPA^{crδ} to ACP^{cr} (Section 15) and expand the full retrospective splitting bisimulation models of BPA^{crδ} to full retrospective splitting bisimulation models of ACP^{cr} (Section 16). Thereupon, we extend ACP^{cr} with condition evaluation operators as well (Section 17). We also outline an interesting application of ACP^{cr} (Section 18). Finally, we make some concluding remarks (Section 19).

2 BPA with Conditions

BPA^δ is a subtheory of ACP that does not support parallelism and communication (see e.g. [7]). In this section, we present an extension of BPA^δ with guarded commands, i.e. conditional expressions of the form \( \zeta : \rightarrow p \). The extension is called BPA^c^δ. In the extension, just as in BPA^δ, it is assumed that a fixed but arbitrary finite set of actions A, with \( \delta \notin A \), has been given. Moreover it is assumed that a fixed but arbitrary set of atomic conditions C at has been given.

Let \( \kappa \) be an infinite cardinal. Then \( C_\kappa \) is the free \( \kappa \)-complete Boolean algebra over \( C_{\text{at}} \). As usual, we identify Boolean algebras with their domain. Thus, we also write \( C_\kappa \) for the domain of \( C_\kappa \). It is well known that, if \( \kappa \) is regular, \( C_\kappa \) is isomorphic to the Boolean algebra of equivalence classes with respect to logical equivalence of the set of all propositions with elements of \( C_{\text{at}} \) as propositional variables and with conjunctions and disjunctions of less than \( \kappa \) propositions (see e.g. [21]). In BPA^c^δ, conditions are taken from \( C_{\kappa_0} \). Moreover, if \( C_{\text{at}} \) is a finite set, then \( C_\kappa = C_{\kappa_0} \) for all cardinals \( \kappa > \kappa_0 \). We are also interested in \( C_\kappa \) for cardinals \( \kappa > \kappa_0 \) because it permits us to consider infinitely branching processes in the case where \( C_{\text{at}} \) is an infinite set. Henceforth, we write \( \mathcal{C} \) for \( C_{\kappa_0} \).

The algebraic theory BPA^c^δ has two sorts:

- the sort \( \mathcal{P} \) of processes;
- the sort \( \mathcal{C} \) of (finite) conditions.

The algebraic theory BPA^c^δ has the following constants and operators to build terms of sort \( \mathcal{C} \):

- the bottom constant \( \bot : \mathcal{C} \);
- the top constant \( \top : \mathcal{C} \);

---

1 For a definition of free \( \kappa \)-complete Boolean algebras, see e.g. [21].
2 For a definition of regular cardinals, see e.g. [22, 16]. They include \( \aleph_0, \aleph_1, \aleph_2, \ldots \).
for each \( \eta \in C \), the atomic condition constant \( \eta : C \);
- the unary complement operator \(- : C \rightarrow C\);
- the binary join operator \( \sqcup : C \times C \rightarrow C\);
- the binary meet operator \( \sqcap : C \times C \rightarrow C\).

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

- the deadlock constant \( \delta : P \);
- for each \( a \in A \), the action constant \( a : P \);
- the binary alternative composition operator \(+ : P \times P \rightarrow P\);
- the binary sequential composition operator \( \cdot : P \times P \rightarrow P\);
- the binary guarded command operator \( : \rightarrow : C \times P \rightarrow P\).

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 \( P \).

The constants and operators of \( \text{BPA}_c^\delta \) to build terms of sort \( P \) are the constants and operators of \( \text{BPA}_\delta \) and additionally the guarded command operator.

Let \( p \) and \( q \) be closed terms of sort \( P \) and \( \zeta \) be a closed term of sort \( C \). Intuitively, the constants and operators to build terms of sort \( P \) can be explained as follows:

- \( \delta \) cannot perform any action;
- \( a \) 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 \( \zeta \).

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 [20]. 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 \( \zeta :\rightarrow p + \neg \zeta :\rightarrow q \).

The axioms of \( \text{BPA}_c^\delta \) are the axioms of Boolean Algebras (BA) given in Table 1 and the additional axioms given in Table 2. Axioms A1–A7 are the axioms of \( \text{BPA}_\delta \). So \( \text{BPA}_c^\delta \) imports the (equational) axioms of both BA and \( \text{BPA}_\delta \). The axioms of BA given in Table 1 have been taken from [18]. Several alternatives for this axiomatization can be found in the literature (e.g. in [21, 23]). If we use basic laws of BA other than axioms BA1–BA8, such as \( \phi \sqcap \phi = \phi \) and \( \neg (\phi \sqcap \psi) = \neg \phi \sqcup \neg \psi \), in a step of a derivation, we will refer to them as applications of BA and not give their derivation from axioms BA1–BA8. Axioms GC1–GC7 have been taken from [2], but with the axiom \( x \cdot z \triangleleft \phi \triangleright y \cdot z = (x \triangleleft \phi \triangleright y) \cdot z \) (CO5) replaced by \( \phi :\rightarrow x \cdot y = (\phi :\rightarrow x) \cdot y \) (GC5).
Table 1. Axioms of Boolean algebras

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA1</td>
<td>$\phi \sqcup \bot = \phi$</td>
<td></td>
</tr>
<tr>
<td>BA2</td>
<td>$\phi \sqcup -\phi = \top$</td>
<td></td>
</tr>
<tr>
<td>BA3</td>
<td>$\phi \sqcup \psi = \psi \sqcup \phi$</td>
<td></td>
</tr>
<tr>
<td>BA4</td>
<td>$(\phi \sqcup (\psi \sqcap \chi)) = (\phi \sqcup \psi) \sqcap (\phi \sqcup \chi)$</td>
<td></td>
</tr>
<tr>
<td>BA5</td>
<td>$\phi \sqcap \top = \phi$</td>
<td></td>
</tr>
<tr>
<td>BA6</td>
<td>$\phi \sqcap -\phi = \bot$</td>
<td></td>
</tr>
<tr>
<td>BA7</td>
<td>$\phi \sqcap \psi = \psi \sqcap \phi$</td>
<td></td>
</tr>
<tr>
<td>BA8</td>
<td>$\phi \sqcap (\psi \sqcup \chi) = (\phi \sqcap \psi) \sqcup (\phi \sqcap \chi)$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Axioms of $\text{BPA}_c$

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>$x + y = y + x$</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>$(x + y) + z = x + (y + z)$</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>$x + x = x$</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>$(x + y) \cdot z = x \cdot z + y \cdot z$</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>$(x \cdot y) \cdot z = x \cdot (y \cdot z)$</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>$x + \delta = x$</td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>$\delta \cdot x = \delta$</td>
<td></td>
</tr>
</tbody>
</table>

Example 2.1. Consider a careful pedestrian who uses a crossing with traffic lights to cross a road with busy traffic safely. When the pedestrian arrives at the crossing and the light for pedestrians is green, he or she simply crosses the street. However, when the pedestrian arrives at the crossing and the light for pedestrians is red, he or she first makes a request for green light (e.g. by pushing a button) and then crosses the street when the light has changed. This behaviour can be described in $\text{BPA}_c$ as follows:

$$\text{PED} = \text{arrive} \cdot (\text{green} :\rightarrow \text{cross} + \text{red} :\rightarrow (\text{make-req} \cdot (\text{green} :\rightarrow \text{cross}))).$$

The careful pedestrian described above does not cross the street if the light for pedestrians does not change from red to green after a request for green light. Whether the change from red to green will ever happen is not described here.

The terms of sort $\mathbf{C}$ are interpreted in $\mathcal{C}$ as usual.

We proceed to the presentation of the structural operational semantics of $\text{BPA}_c$. The following relations on closed terms of $\text{BPA}_c$ are used:

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

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

The relations $\ell\rightarrow\sqrt{}$ and $\ell\rightarrow$ can be explained as follows:

- $p \xrightarrow[\alpha]{a} \sqrt{}$: $p$ 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 $q$. 

5
Table 3. Transition rules for BPA\textsuperscript{c}δ

\[\begin{array}{llll}
\frac{\alpha}{[\top]a} \Rightarrow \\
\frac{x \rightarrow [\phi]a}{x + y \rightarrow [\phi]a} \\
\frac{x \rightarrow [\phi]a}{x \rightarrow [\phi]a, x'} \\
\frac{x \rightarrow [\phi]a}{x \rightarrow [\phi]a, y'} \\
\frac{x \rightarrow [\phi]a}{x \rightarrow [\phi]a, x' + y'} \\
\frac{x \rightarrow [\phi]a}{x \rightarrow [\phi]a, x' + y'} \\
\frac{\psi := x}{\phi \land \psi \neq \bot} \\
\end{array}\]

The structural operational semantics of BPA\textsuperscript{c}δ 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 BPA\textsuperscript{c}δ

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 BPA\textsuperscript{c}δ. 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 BPA\textsuperscript{c}δ.

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

Let \(\kappa\) be an infinite cardinal. Then a \(\kappa\)-conditional transition system \(T\) consists of the following:
- a set \(S\) of states;
- a set \(\ell \subseteq S \times S\), for each \(\ell \in C_{\kappa}^{-} \times A\);
- a set \(\ell \rightarrow \subseteq S\), for each \(\ell \in C_{\kappa}^{-} \times A\);
- an initial state \(s^0 \in S\).

If \((s, s') \in \ell\), then we say that there is a transition from \(s\) to \(s'\). We usually write \(s \xrightarrow{[\alpha]a} s'\) instead of \((s, s') \in \ell\) and \(s \xrightarrow{[\alpha]a} \top\) instead of \(s \in \ell\). Furthermore, we write \(\rightarrow\) for the family of sets \((\ell)_{\ell \in C_{\kappa}^{-} \times A}\) and \(\rightarrow \top\) for the family of sets \((\ell)_{\ell \in C_{\kappa}^{-} \times A}\).

The relations \(\ell \rightarrow\) and \(\ell \rightarrow \top\) can be explained as follows:
- \(s \xrightarrow{[\alpha]a} \top\); 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. Unreachable states, and the transitions between them, are not relevant to the behaviour represented by the transition system. Connected conditional transition systems are transition systems without unreachable states.

Let \( T = (S, \rightarrow, \rightarrow\sqrt{\cdot}, s^0) \) be a \( \kappa \)-conditional transition system (for an infinite cardinal \( \kappa \)). Then the reachability relation of \( T \) is the smallest relation \( \rightarrow \subseteq S \times S \) such that:

- \( s \rightarrow s \);
- if \( s \xrightarrow{\ell} s' \) and \( s' \rightarrow s'' \), then \( s \rightarrow s'' \).

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

Let \( T = (S, \rightarrow, \rightarrow\sqrt{\cdot}, s^0) \) be a \( \kappa \)-conditional transition system (for an infinite cardinal \( \kappa \)) that is not necessarily connected. Then the connected part of \( T \), written \( \Gamma(T) \), is defined as follows:

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

where

\[ S' = \text{RS}(T), \]

and for every \( \ell \in C^-_\kappa \times A \):

\[ \xrightarrow{\ell} = \xrightarrow{\ell} \cap (S' \times S'), \]

\[ \xrightarrow{\ell} \sqrt{\cdot} = \xrightarrow{\ell} \sqrt{\cdot} \cap S'. \]

It is assumed that for each infinite cardinal \( \kappa \) a fixed but arbitrary set \( S_\kappa \) of cardinality greater than or equal to \( \kappa \), and that is closed under disjoint union and cartesian product, has been given.

Let \( \kappa \) be an infinite cardinal. Then \( \text{CTS}_\kappa \) is the set of all connected \( \kappa \)-conditional transition systems \( T = (S, \rightarrow, \rightarrow\sqrt{\cdot}, s^0) \) 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 \( \{(\ell, s') \in (C^-_\kappa \times A) \times S \mid (s, s') \in \xrightarrow{\ell}\} \cup \{\ell \in C^-_\kappa \times A \mid s \in \xrightarrow{\ell}\sqrt{\cdot}\} \) is less than \( \kappa \).

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

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

The identity of the states of a conditional transition system is not relevant to the behaviour represented by it. Conditional transition system that differ only with respect to the identity of the states are isomorphic.
Let $T_1 = (S_1, \rightarrow_1, \varepsilon_1, s_1^0)$ and $T_2 = (S_2, \rightarrow_2, \varepsilon_2, s_2^0)$ be $\kappa$-conditional transition systems (for an infinite cardinal $\kappa$). Then $T_1$ and $T_2$ are isomorphic, written $T_1 \cong T_2$, if there exists a bijective function $b: S_1 \rightarrow S_2$ such that:

- $b(s_1^0) = s_2^0$;
- $s_1 \xrightarrow{\ell_1} s_1' \iff b(s_1) \xrightarrow{\ell_2} b(s_1')$;
- $s \xrightarrow{\varepsilon_1} \sqrt{1} \iff b(s) \xrightarrow{\varepsilon_2} \sqrt{2}$.

If $T_1$ and $T_2$ are isomorphic, then they differ only with respect to the identity of the states. 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 two well-known notions from the field of Boolean algebras: a partial order relation $\subseteq$ on $C_\kappa$ and a unary operation $\bigcup$ on the set of all subsets of $C_\kappa$ of cardinality less than $\kappa$ (for each infinite cardinal $\kappa$). The relation $\subseteq$ and the operation $\bigcup$ are defined by

$$\alpha \subseteq \beta \iff \alpha \sqcup \beta = \beta$$

and

$$\bigcup C \text{ is the supremum of } C \text{ in } (C_\kappa, \subseteq),$$

respectively. The operation $\bigcup$ is defined for all subsets of $C_\kappa$ of cardinality less than $\kappa$ because $C_\kappa$ is $\kappa$-complete.

Let $T_1 = (S_1, \rightarrow_1, \varepsilon_1, s_1^0) \in \text{CTS}_\kappa$ and $T_2 = (S_2, \rightarrow_2, \varepsilon_2, s_2^0) \in \text{CTS}_\kappa$ (for an infinite cardinal $\kappa$). Then a splitting bisimulation $B$ between $T_1$ and $T_2$ is a binary relation $B \subseteq S_1 \times S_2$ such that $B(s_1^0, s_2^0)$ and for all $s_1, s_2$ such that $B(s_1, s_2)$:

- if $s_1 \xrightarrow{\alpha, a} s_1'$, then there is a set $CS'_1 \subseteq C^-_\kappa \times S_2$ of cardinality less than $\kappa$ such that $\alpha \subseteq \bigcup (\text{dom}(CS'_1))$ and for all $(\alpha', s'_2) \in CS'_2$, $s_2 \xrightarrow{\alpha', a, 2} s'_2$ and $B(s_1', s_2')$;
- if $s_2 \xrightarrow{\alpha_2, a} s_2'$, then there is a set $CS_1' \subseteq C^-_\kappa \times S_1$ of cardinality less than $\kappa$ such that $\alpha \subseteq \bigcup (\text{dom}(CS_1'))$ and for all $(\alpha', s'_1) \in CS_1'$, $s_1 \xrightarrow{\alpha', a, 1} s_1'$ and $B(s_1', s_2')$;
- if $s_1 \xrightarrow{\alpha, a} \sqrt{1}$, then there is a set $C' \subseteq C^-_\kappa$ of cardinality less than $\kappa$ such that $\alpha \subseteq \bigcup C'$ and for all $\alpha' \in C'$, $s_2 \xrightarrow{\alpha', a} \sqrt{2}$;
- if $s_2 \xrightarrow{\alpha, a} \sqrt{2}$, then there is a set $C' \subseteq C^-_\kappa$ of cardinality less than $\kappa$ such that $\alpha \subseteq \bigcup C'$ and for all $\alpha' \in C'$, $s_2 \xrightarrow{\alpha, a} \sqrt{1}$.

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 $T_1 \equiv T_2$. 

8
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 \( \text{CTS}_\kappa \) as the domain of a structure that is a model of \( \text{BPA}_\delta \). 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 \( T = (S, \rightarrow, \rightarrow^\square, s^0) \) be a connected \( \kappa \)-conditional transition system of which the branching degree is less than \( \kappa \). Then there exists a connected \( \kappa \)-conditional transition system \( T' = (S', \rightarrow', \rightarrow^\square', s^0') \) 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 \( \text{BPA}_\delta^c \)

In this section, we introduce the full splitting bisimulation models of \( \text{BPA}_\delta^c \). 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 \( \text{BPA}_\delta^c \) are structures that consist of the following:

- a non-empty set \( D \), called the domain of the model;
- for each constant of \( \text{BPA}_\delta^c \), an element of \( D \);
- for each \( n \)-ary operator of \( \text{BPA}_\delta^c \), an \( n \)-ary operation on \( D \).

In the full splitting bisimulation models of \( \text{BPA}_\delta^c \) that are introduced in this section, the domain is \( \text{CTS}_\kappa /\equiv \) for some infinite cardinal \( \kappa \). We obtain the models concerned by associating certain elements of \( \text{CTS}_\kappa /\equiv \) and certain operations on \( \text{CTS}_\kappa /\equiv \) with the constants and operators of \( \text{BPA}_\delta^c \). 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 \( \text{CTS}_\kappa /\equiv \).
It is assumed that for each infinite cardinal $\kappa$ a fixed but arbitrary function $\text{ch}_\kappa : (\mathcal{P}(\mathcal{S}_\kappa) \setminus \emptyset) \to \mathcal{S}_\kappa$ such that for all $S \in \mathcal{P}(\mathcal{S}_\kappa) \setminus \emptyset$, $\text{ch}_\kappa(S) \in S$ has been given.

We associate with each constant $c$ of $\text{BPA}_c^\kappa$ an element $\hat{c}$ of $\text{CTS}_\kappa$ and with each operator $f$ of $\text{BPA}_c^\kappa$ an operation $\hat{f}$ on $\text{CTS}_\kappa$ as follows.

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

- $\hat{a} = (\{s^0\}, \emptyset, \rightarrow \sqrt{1}, s^0)$, where $s^0 = \text{ch}_\kappa(\mathcal{S}_\kappa)$, $(\top, a) \rightarrow \sqrt{1} = \{s^0\}$, and for every $(\alpha, a') \in (\mathcal{C}_\kappa^+ \times A) \setminus \{(\top, a)\}$:
  $(\alpha, a') \rightarrow \sqrt{1} = \emptyset$.

- Let $T_i = (S, \rightarrow_i, \rightarrow_i, s^0) \in \text{CTS}_\kappa$ for $i = 1, 2$. Then
  $$T_1 \hat{+} T_2 = \Gamma(S, \rightarrow, \rightarrow, s^0)$$
  where $s^0 = \text{ch}_\kappa(\mathcal{S}_\kappa \setminus (S_1 \uplus S_2))$, $S = \{s^0\} \cup (S_1 \uplus S_2)$, \(^3\) and for every $(\alpha, a) \in \mathcal{C}_\kappa^- \times A$:
  \[
  (\alpha, a) \rightarrow = \{(s^0, \mu_1(s)) \mid s^0 \rightarrow_1 s_1^{\alpha_1} \} \\
  \cup \{(s^0, \mu_2(s)) \mid s^0 \rightarrow_2 s_2^{\alpha_2} \} \\
  \cup \{(\mu_1(s), \mu_1(s')) \mid s \rightarrow_1 s_1^{\alpha_1} s' \} \\
  \cup \{(\mu_2(s), \mu_2(s')) \mid s \rightarrow_2 s_2^{\alpha_2} s' \}, \]

  $(\alpha, a) \rightarrow \sqrt{1} = \{s^0 \mid s^0 \rightarrow_1^{\alpha_1} \}$
  $\cup \{s^0 \mid s^0 \rightarrow_2^{\alpha_2} \}$
  $\cup \{\mu_1(s) \mid s \rightarrow_1^{\alpha_1} \}$
  $\cup \{\mu_2(s) \mid s \rightarrow_2^{\alpha_2} \}$.

\(^3\) We use $\uplus$ to denote disjoint union of sets. The associated injections are denoted by $\mu_1$ and $\mu_2$. 

10
Remark 4.1. The elements of CTS, due to absence of termination in T, in the definition of alternative composition on CTS, Proposition 4.1 (Congruence). Let κ be an infinite cardinal. Then for all T_1, T_2, T'_1, T'_2 ∈ CTS_κ and α ∈ C_κ, T_1 ⊆ T'_1 and T_2 ⊆ T'_2 imply T_1 ∨ T_2 ⊆ T'_1 ∨ T'_2, T_1 ∨ T_2 ⊆ T'_1 ∨ T'_2 and α := T_1 ⊆ α := T'_1.
Proof. Let \( T_i = (S_i, \rightarrow_i, \cdot_i, \rightarrow^\vee_i, s^0_i) \) and \( T'_i = (S'_i, \rightarrow'_i, \cdot_i', \rightarrow^\vee_i', s'^0_i) \) for \( i = 1, 2 \). Let \( R_1 \) and \( R_2 \) be splitting bisimulations witnessing \( T_1 \cong T_1' \) and \( T_2 \cong T_2' \), respectively. Then we construct relations \( R_+ \), \( R_- \) and \( R_\cong \) as follows:

- \( R_+ = \{(s^0, s'^0)\} \cup \mu_1(R_1) \cup \mu_2(R_2) \cap (S \times S') \), where \( S \) and \( S' \) are the sets of states of \( T_1 \) and \( T_1' \) respectively, and \( s^0 \) and \( s'^0 \) are the initial states of \( T_1 \) and \( T_1' \) respectively;
- \( R_- = \{(\mu_1(R_1) \cup \mu_2(R_2)) \cap (S \times S') \} \), where \( S \) and \( S' \) are the sets of states of \( T_1 \) and \( T_1' \) respectively;
- \( R_\cong = R_1 \cap (S \times S') \), where \( S \) and \( S' \) are the sets of states of \( \alpha \models T_1 \) and \( \alpha \models T_1' \) 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 \( S_i \) into \( S_1 \uplus S_2 \) and the injection of \( S'_i \) into \( S'_1 \uplus S'_2 \). Given the definitions of alternative composition, sequential composition and guarded command, it is easy to see that \( R_+ \), \( R_- \) and \( R_\cong \) are splitting bisimulations witnessing \( T_1 \hat{\rightarrow} T_2 \cong T_1' \hat{\rightarrow} T_2' \), \( T_1 \hat{\rightarrow} T_2 \cong T_1' \hat{\rightarrow} T_2' \) and \( \alpha \models T_1 \cong \alpha \models T_1' \), respectively.

The full splitting bisimulation models \( \Psi^\kappa \), one for each infinite cardinal \( \kappa \), consist of the following:

- a set \( \mathcal{P} \), called the domain of \( \Psi^\kappa \);
- for each constant \( c \) of \( \text{BPA}^\kappa \), an element \( \bar{c} \) of \( \mathcal{P} \);
- for each \( n \)-ary operator \( f \) of \( \text{BPA}^\kappa \), an \( n \)-ary operation \( \bar{f} \) on \( \mathcal{P} \);

where those ingredients are defined as follows:

\[
\mathcal{P} = \text{CTS}_\kappa / \equiv, \quad [T_1]_\equiv \hat{\rightarrow} [T_2]_\equiv = [T_1 \hat{\rightarrow} T_2]_\equiv, \\
\tilde{\delta} = [\tilde{\delta}]_\equiv, \quad [T_1]_\equiv \hat{\rightarrow} [T_2]_\equiv = [T_1 \hat{\rightarrow} T_2]_\equiv, \\
\tilde{\alpha} = [\tilde{\alpha}]_\equiv, \quad \alpha \models [T_1]_\equiv = [\alpha \models T_1]_\equiv.
\]

The operations alternative composition, sequential composition and guarded command 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 \( \Psi^\kappa \) are models of \( \text{BPA}^\kappa \).

**Theorem 4.1 (Soundness of \( \text{BPA}^\kappa \)).** For each infinite cardinal \( \kappa \), we have \( \Psi^\kappa \models \text{BPA}^\kappa \).

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 \( \Psi^\kappa \). □

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

\[\Phi \] is the Gothic capital P.
Theorem 4.2 (Isomorphic Embedding). Let $\kappa$ and $\kappa'$ be infinite cardinals such that $\kappa < \kappa'$. Then $\mathcal{P}_\kappa$ is isomorphically embedded in $\mathcal{P}_{\kappa'}$.

Proof. It follows immediately from the definitions of $\text{CTS}_\kappa$, $\text{CTS}_{\kappa'}$ and $\equiv$ that for each $P \in \text{CTS}_\kappa/\equiv$, there exists a unique $P' \in \text{CTS}_{\kappa'}/\equiv$ such that $P \subseteq P'$. Now consider the function $h: \text{CTS}_\kappa/\equiv \rightarrow \text{CTS}_{\kappa'}/\equiv$ where for each $P \in \text{CTS}_\kappa/\equiv$, $h(P)$ is the unique $P' \in \text{CTS}_{\kappa'}/\equiv$ such that $P \subseteq P'$. It follows immediately from the definition of $h$ that $h$ is injective. Moreover, it follows easily from the definitions of the operations on $\text{CTS}_\kappa/\equiv$ and $\text{CTS}_{\kappa'}/\equiv$ that $h$ is a homomorphism from $\mathcal{P}_\kappa$ to $\mathcal{P}_{\kappa'}$. \square

5 SOS-Based Splitting Bisimilarity for BPA$_c^\delta$

It is customary to associate transition systems with closed terms of 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 splitting bisimilar.

The structural operational semantics of BPA$_c^\delta$ presented in Section 2 determines a conditional transition system for each process that can be denoted by a closed term of BPA$_c^\delta$. These transition systems are special in the sense that their states are closed terms of BPA$_c^\delta$.

Let $p$ be a closed term of BPA$_c^\delta$. Then the transition system of $p$ induced by the structural operational semantics of BPA$_c^\delta$, written $\text{CTS}(p)$, is the connected conditional transition system $\Gamma(S, \rightarrow, \rightarrow\sqrt{}, s^0)$, where:

- $S$ is the set of closed terms of BPA$_c^\delta$;
- the sets $\{(\alpha,a)\} \subseteq S \times S$ and $\{\alpha\} \sqrt{} \subseteq S$ 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 BPA$_c^\delta$. Then we say that $p$ and $q$ are splitting bisimilar, written $p \equiv q$, if $\text{CTS}(p) \equiv \text{CTS}(q)$.

Clearly, the structural operational semantics does not give rise to infinitely branching conditional transition systems. For each closed term $p$ of BPA$_c^\delta$, there exists a $T \in \text{CTS}_{\aleph_0}$ such that $\text{CTS}(p) \cong T$. In Section 4, it has been shown that it is possible to consider infinitely branching conditional transition systems as well.

6 ACP with Conditions

In order to support parallelism and communication, we add parallel composition and encapsulation operators to BPA$_c^\delta$, resulting in ACP$^c$.

Like in BPA$_c^\delta$, it is assumed that a fixed but arbitrary finite set of actions $A$, with $\delta \not\in A$, and a fixed but arbitrary set of atomic conditions $C_\Delta$ has been
We use infix notation for the additional binary operators as well.

\[ a \cdot x \parallel y = a \cdot (x \parallel y) \]

\[ a \cdot x \parallel y = a \cdot (x \parallel y) \]

\[ (x + y) \parallel z = x \parallel (z + y) \parallel z \]

\[ a \cdot x \mid b = (a \mid b) \cdot x \]

\[ a \mid b \cdot x = (a \mid b) \cdot x \]

\[ a \cdot x \mid b \cdot y = (a \mid b) \cdot (x \parallel y) \]

\[ (x + y) \mid z = x \mid z + y \mid z \]

\[ x \mid (y + z) = x \mid y + x \mid z \]

\[ a \mid b = b \mid a \]

\[ a \mid b \mid c = a \mid (b \mid c) \]

\[ \delta \mid a = \delta \]

We use infix notation for the additional binary operators as well.

The constants and operators of ACP are the constants and operators of BPA and additionally the guarded command operator. Let \( p \) and \( q \) be closed terms of ACP. Intuitively, the additional operators can be explained as follows:

- \( p \parallel q \) behaves as the process that proceeds with \( p \) and \( q \) in parallel;
- \( p \parallel q \) behaves the same as \( p \parallel q \), except that it starts with performing an action of \( p \);  
- \( p \parallel q \) behaves the same as \( p \parallel q \), except that it starts with performing an action of \( p \) and an action of \( q \) synchronously;
- \( \partial_H(p) \) behaves the same as \( p \), except that it does not perform actions in \( H \).

The axioms of ACP are the axioms of BPA and the additional axioms given in Table 4. CM2–CM3, CM5–CM7, C1–C3 and D1–D2 are actually axiom schemas in which \( a, b \) and \( c \) stand for arbitrary constants of ACP (i.e. \( a, b, c \in \mathbb{A}_3 \)). In D1–D4, \( H \) stands for an arbitrary subset of \( \mathbb{A} \). So, D3 and D4 are axiom schemas as well.
possibility that actions are performed synchronously is not covered by PA with axiom CM1 replaced by the communication merge operator, without axioms CM5–CM9 and C1–C3, andwise, we have a subtheory of ACP.

In this section, we expand the full splitting bisimulation models of BPA

7 Full Splitting Bisimulation Models of ACP

Table 5. Additional transition rules for ACP

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \parallel y \vdash a$</td>
<td>$y \vdash a$</td>
<td>$x \vdash a$</td>
<td>$y \vdash a, y'$</td>
<td></td>
</tr>
<tr>
<td>$x \parallel y \vdash a$</td>
<td>$y \vdash b$</td>
<td>$x \vdash b$</td>
<td>$y \vdash b, y'$</td>
<td></td>
</tr>
<tr>
<td>$x \parallel y \vdash a$</td>
<td>$y \vdash b$, $z'$</td>
<td>$x \vdash b$, $z'$</td>
<td>$y \vdash b$, $z'$, $y'$</td>
<td></td>
</tr>
<tr>
<td>$x \vdash a$</td>
<td>$y \vdash b$, $z'$</td>
<td>$x \vdash b$, $z'$</td>
<td>$y \vdash b$, $z'$, $y'$</td>
<td></td>
</tr>
<tr>
<td>$\partial_H(x) \vdash a \notin H$</td>
<td>$x \vdash a$</td>
<td>$x \vdash a, x'$</td>
<td>$\partial_H(x') \vdash a \notin H$</td>
<td></td>
</tr>
</tbody>
</table>

Axioms A1–A7, CM1–CM9, C1–C3 and D1–D4 are the axioms of ACP. So ACP$^c$ imports the axioms of ACP.

A well-known subtheory of ACP is PA, ACP without communication. Likewise, we have a subtheory of ACP$^c$, to wit PA$^c$. The theory PA$^c$ is ACP$^c$ without the communication merge operator, without axioms CM5–CM9 and C1–C3, and with axiom CM1 replaced by $x \parallel y = x \parallel y + y \parallel x$ (M1). In other words, the possibility that actions are performed synchronously is not covered by PA$^c$.

The structural operational semantics of ACP$^c$ is described by the transition rules for BPA$^c_3$ and the additional transition rules given in Table 5.

7 Full Splitting Bisimulation Models of ACP

In this section, we expand the full splitting bisimulation models of BPA$^c_3$ to ACP$^c$. We will use the abbreviation $s \equiv s'$ for $s \equiv s'$.

First of all, we associate with each additional operator $f$ of ACP$^c$ an operation $\hat{f}$ on $\mathcal{CTS}_\kappa$ as follows.

- Let $T_i = (S_i, \rightarrow_i, \rightarrow_i \sqrt{v}, s_i^0)$, $\in \mathcal{CTS}_\kappa$ for $i = 1, 2$. Then
  
  $T_1 \parallel T_2 = (S, \rightarrow, \rightarrow \sqrt{v}, s^0)$,

where
$s^0 = (s^0_1, s^0_2)$,

$s^\prime = \text{ch}_c(S_\kappa \setminus (S_1 \cup S_2))$,

$S = ((S_1 \cup \{s^\prime\}) \times (S_2 \cup \{s^\prime\})) \setminus \{(s^\prime, s^\prime)\}$,

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

$\frac{(a, a)\cdot}{(a, a)} = \{(s_1, s^\prime) | s_1 \in S \land \frac{[a]}{[a]}_\kappa s^\prime \} \cup \{(s_1, s^\prime) | s_2 \in S \land \frac{[a]}{[a]}_\kappa s^\prime \} \cup \{(s_1, s^\prime) | (s_1, s_2^\prime) \in S \land \sqrt{\frac{[\beta]}{[\beta]}_\kappa s^\prime \wedge \sqrt{\frac{[\beta]}{[\beta]}_\kappa s^\prime \wedge \frac{[\beta]}{[\beta]}_\kappa s^\prime}

\frac{(a, a)\cdot}{(a, a)} \sqrt{\kappa} = \{(s_1, s^\prime) | s_1 \in S \land \frac{[a]}{[a]}_\kappa s^\prime \} \cup \{(s_1, s^\prime) | s_2 \in S \land \frac{[a]}{[a]}_\kappa s^\prime \} \cup \{(s_1, s^\prime) | (s_1, s_2^\prime) \in S \land \sqrt{\frac{[\beta]}{[\beta]}_\kappa s^\prime \wedge \sqrt{\frac{[\beta]}{[\beta]}_\kappa s^\prime \wedge \sqrt{\frac{[\beta]}{[\beta]}_\kappa s^\prime}}

- Let $T_i = (S_i, \rightarrow, \rightarrow, \rightarrow, s^0_i) \in \text{CTS}_\kappa$ for $i = 1, 2$. Suppose that $T_1 \equiv T_2 = (S, \rightarrow, \rightarrow, \rightarrow, s^0)$ where $S = ((S_1 \cup \{s^\prime\}) \times (S_2 \cup \{s^\prime\})) \setminus \{(s^\prime, s^\prime)\}$ and $s^\prime = \text{ch}_c(S_\kappa \setminus (S_1 \cup S_2))$. Then

$T_1 \equiv T_2 = \Gamma(S^\prime, \rightarrow, \rightarrow, \rightarrow, s^{0^\prime})$,

where

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

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

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

$\frac{(a, a)\cdot}{(a, a)} \equiv = \{(s_1, s^{0^\prime}) | s_1 \in S \land \frac{[a]}{[a]}_\kappa s^{0^\prime} \} \cup \{(s_1, s^{0^\prime}) | s_2 \in S \land \frac{[a]}{[a]}_\kappa s^{0^\prime} \}$

- Let $T_i = (S_i, \rightarrow, \rightarrow, \rightarrow, s^0_i) \in \text{CTS}_\kappa$ for $i = 1, 2$. Suppose that $T_1 \equiv T_2 = (S, \rightarrow, \rightarrow, \rightarrow, s^0)$ where $S = ((S_1 \cup \{s^\prime\}) \times (S_2 \cup \{s^\prime\})) \setminus \{(s^\prime, s^\prime)\}$ and $s^\prime = \text{ch}_c(S_\kappa \setminus (S_1 \cup S_2))$. Then

$T_1 \equiv T_2 = \Gamma(S^\prime, \rightarrow, \rightarrow, \rightarrow, s^{0^\prime})$,

where

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

$S^\prime = \{s^{0^\prime}\} \cup S$,
and for every \((\alpha, a) \in C^-_\kappa \times A^\ast\):
\[
\frac{\langle \alpha, a \rangle}{\overrightarrow{\times}} = \{ (s^0', (s_1, s_2)) \mid (s_1, s_2) \in S \land \exists \alpha' \in C^-_\kappa, a' \in A \land (s^0, s_1, s_2) \stackrel{\alpha'}{\rightarrow} s'_1 \land s'_2 \land \exists \beta' \in C^-_\kappa, b' \in A \land (s^0, s'_1, s'_2) \stackrel{\beta'}{\rightarrow} s'_2 \land\}
\cup \frac{\langle \alpha, a \rangle}{\overrightarrow{\gamma}} = \{ (s^0|, (s_1, s_2)) \mid (s_1, s_2) \in S \land \exists \alpha' \in C^-_\kappa, a' \in A \land (s^0, s_1, s_2) \stackrel{\alpha'}{\rightarrow} s'_1 \land s'_2 \land \exists \beta' \in C^-_\kappa, b' \in A \land (s^0, s'_1, s'_2) \stackrel{\beta'}{\rightarrow} s'_2 \land\}
\cup \frac{\langle \alpha, a \rangle}{\overrightarrow{\gamma}'} = \emptyset ,
\]
\[
\frac{\langle \alpha, a \rangle}{\overrightarrow{\gamma}} = \emptyset .
\]

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

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

**Proof.** Let \(R_1\) and \(R_2\) be splitting bisimulations witnessing \(T_1 \cong T'_1\) and \(T_2 \cong T'_2\), respectively. Then we construct relations \(R^\parallel, R^\parallel_1, R^\parallel_2\) and \(R^\delta_H\) as follows:

- \(R^\parallel = \{ ((s_1, s_2), (s'_1, s'_2)) \mid (s_1, s_2) \in R_1, (s'_1, s'_2) \in R_2 \}\);

- \(R^\parallel_1 = \{ (s^0, s^0') \} \cup R^\parallel\) \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;

- \(R^\parallel_2 = \{ (s^0, s^0') \} \cup R^\parallel\) \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^\delta_H = R_1 \cap (S \times S'), where \(S\) and \(S'\) are the sets of states of \(\delta_H(T_1)\) and \(\delta_H(T'_1)\), respectively.
Given the definitions of parallel composition, left merge, communication merge and encapsulation, it is easy to see that $R_{\parallel}, R_{\mid}, R_{\hat{\mid}}$ and $R_{\hat{\delta}}$ are splitting bisimulations witnessing $T_1 \hat{\mid} T_2 \equiv T_1 \hat{\mid} T_2, T_1 \hat{\mid} T_2 \equiv T_1 \hat{\mid} T_2, T_1 \hat{\mid} T_2 \equiv T_1 \hat{\mid} T_2$ and $\hat{\delta}_H(T_1) \equiv \hat{\delta}_H(T_1')$, respectively.

The full splitting bisimulation models $\Psi'_\kappa$ of ACP\(^c\), one for each infinite cardinal $\kappa$, are the expansions of the full splitting bisimulation models $\Psi_\kappa$ of BPA\(^\kappa\) with an $n$-ary operation $f$ on the domain of $\Psi'_\kappa (\text{CTS}_\kappa/\equiv)$ for each additional $n$-ary operator $f$ of ACP\(^c\). Those additional operations are defined as follows:

\[
[T_1]_{\equiv} \hat{\mid} [T_2]_{\equiv} = [T_1 \hat{\mid} T_2]_{\equiv}, \\
[T_1]_{\equiv} \parallel [T_2]_{\equiv} = [T_1 \parallel T_2]_{\equiv}, \\
[T_1]_{\equiv} \downarrow [T_2]_{\equiv} = [T_1 \downarrow T_2]_{\equiv}, \\
\hat{\delta}_H([T_1]_{\equiv}) = [\hat{\delta}_H(T_1)]_{\equiv}.
\]

Parallel composition, left merge, communication merge and encapsulation on CTS\(\kappa/\equiv\) are well-defined because $\equiv$ is a congruence with respect to the corresponding operations on CTS\(\kappa\).

The structures $\Psi'_\kappa$ are models of ACP\(^c\).

**Theorem 7.1 (Soundness of ACP\(^c\)).** For each infinite cardinal $\kappa$, we have $\Psi'_\kappa \models \text{ACP}^c$.

*Proof.* Because $\equiv$ is a congruence, it is sufficient to show that all additional axioms are sound. The soundness of all additional axioms follows easily from the definitions of the ingredients of $\Psi'_\kappa$.

It is easy to see that Theorem 4.2 goes through for $\Psi'_\kappa$.

In this section, the full splitting bisimulation models $\Psi'_\kappa$ of BPA\(^\kappa\) have been expanded to obtain the full splitting bisimulation models $\Psi'_\kappa$ of ACP\(^c\). Henceforth, we will loosely write $\Psi'_\kappa$ for $\Psi'_\kappa$. It is always made sure that no confusion between the original model and its expansion may arise.

## 8 Guarded Recursion

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

A recursive specification over ACP\(^c\) is a set of recursive equations $E = \{X = t_X \mid X \in V\}$ where $V$ is a set of variables and each $t_X$ is a term of ACP\(^c\) 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 ACP\(^c\)) $\{P_X \mid X \in V(E)\}$ 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 ACP\(^c\) containing a variable $X$. We call an occurrence of $X$ in $t$ guarded if $t$ has a subterm of the form $a \cdot t'$, where $a \in A$ and $t'$ a term

\[18\]
of $\text{ACP}^\kappa$, with $t'$ containing this occurrence of $X$. A recursive specification over $\text{ACP}^\kappa$ 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 $\text{ACP}^\kappa$ and the equations of the recursive specification. We are only interested in models of $\text{ACP}^\kappa$ 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 $\langle X | E \rangle$. We often write $X$ for $\langle X | E \rangle$ 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 $\text{ACP}^\kappa$ and $E$ be a guarded recursive specification over $\text{ACP}^\kappa$. Then we write $\langle t | E \rangle$ for $t$ with, for all $X \in V(E)$, all occurrences of $X$ in $t$ replaced by $\langle X | E \rangle$.

The additional axioms for recursion are the equations given in Table 6. Both RDP and RSP are axiom schemas. A side condition is added to restrict the variables, terms and guarded recursive specifications for which $X$, $t_X$ and $E$ stand. The additional axioms for recursion are known as the recursive definition principle (RDP) and the recursive specification principle (RSP). The equations $\langle X | E \rangle = \langle t_X | E \rangle$ for a fixed $E$ express that the constants $\langle X | E \rangle$ make up a solution of $E$. The conditional equations $E \Rightarrow X = \langle X | E \rangle$ express that this solution is the only one.

The structural operational semantics for the constants $\langle X | E \rangle$ is described by the transition rules given in Table 7.

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

**Theorem 8.1 (Unique solutions in $\Psi^\kappa_\kappa$).** For each infinite cardinal $\kappa$, guarded recursive specifications over $\text{ACP}^\kappa$ have unique solutions in $\Psi^\kappa_\kappa$.

**Proof.** In [5], a proof of uniqueness of solutions of guarded recursive specifications in the graph models of $\text{ACP}_{\kappa}$ is given. That proof can easily be adapted to the full bisimulation models of $\text{ACP}$ introduced in [10]. The proof consists of the following three steps: (i) proving that two transition systems are bisimilar if at least one of them is finitely branching and all their finite projections are...
Thus, the full splitting bisimulation models $\mathfrak{P}_\kappa''$ of $\ACP^\kappa$ with guarded recursion are simply the expansions of the full splitting bisimulation models $\mathfrak{P}_\kappa$ of $\ACP^\kappa$ 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 Evaluation of Conditions

Guarded commands cannot always be eliminated from closed terms of $\ACP^\kappa$ because conditions different from both $\bot$ and $\top$ may be involved. The condition evaluation operators introduced below, can be brought into action in such cases. These operators require to fix an infinite cardinal $\lambda$. By doing so, full splitting bisimulation models with domain $\operatorname{CTS}_\kappa/\equiv_\lambda$ for $\kappa > \lambda$ are excluded.

There are unary $\lambda$-complete condition evaluation operators $\operatorname{CE}_h: \mathcal{P} \to \mathcal{P}$ and $\operatorname{CE}_h: \mathcal{C} \to \mathcal{C}$ for each $\lambda$-complete endomorphisms $h$ of $\mathcal{C}_\lambda$.

These operators can be explained as follows: $\operatorname{CE}_h(p)$ behaves as $p$ with each condition $\zeta$ occurring in $p$ replaced according to $h$. If the image of $\mathcal{C}_\lambda$ under $h$ is $\mathbb{B}$, i.e. the Boolean algebra with domain $\{\bot, \top\}$, then guarded commands can be eliminated from $\operatorname{CE}_h(p)$. In the case where the image of $\mathcal{C}_\lambda$ under $h$ is not $\mathbb{B}$, $\operatorname{CE}_h$ can be regarded to evaluate the conditions only partially.

Henceforth, we write $\mathcal{H}_\lambda$ for the set of all $\lambda$-complete endomorphisms of $\mathcal{C}_\lambda$.

The additional axioms for $\operatorname{CE}_h$, where $h \in \mathcal{H}_\lambda$, are the axioms given in Table 8.

### Table 8. Axioms for condition evaluation ($a \in \mathcal{A}$, $\eta \in \mathcal{C}_\kappa$, $\eta' \in \mathcal{C}_\kappa \cup \{\bot, \top\}$)

<table>
<thead>
<tr>
<th>CE axioms</th>
<th>CE numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\operatorname{CE}_h(a) = a$</td>
<td>CE1</td>
</tr>
<tr>
<td>$\operatorname{CE}_h(a \cdot x) = a \cdot \operatorname{CE}_h(x)$</td>
<td>CE2</td>
</tr>
<tr>
<td>$\operatorname{CE}_h(x + y) = \operatorname{CE}_h(x) + \operatorname{CE}_h(y)$</td>
<td>CE3</td>
</tr>
<tr>
<td>$\operatorname{CE}_h(\phi :\rightarrow x) = \operatorname{CE}_h(\phi) :\rightarrow \operatorname{CE}_h(x)$</td>
<td>CE4</td>
</tr>
<tr>
<td>$\operatorname{CE}<em>h(\mathcal{C}</em>\kappa(x)) = \operatorname{CE}<em>h</em>{\triangleright h}(x)$</td>
<td>CE5</td>
</tr>
<tr>
<td>$\operatorname{CE}_h(\phi \sqcap \psi) = \operatorname{CE}_h(\phi) \sqcap \operatorname{CE}_h(\psi)$</td>
<td>CE11</td>
</tr>
</tbody>
</table>

5 For a definition of $\kappa$-complete endomorphisms, see e.g. [21].
Table 9. Transition rules for condition evaluation

| x \[φ \] a \[ h(φ) \] \[ CE \] h(x) | h(φ) \neq ⊥ \[ x \[ h(φ) \] a \[ h(φ) \] \[ CE \] h(x') | h(φ) \neq ⊥ |
|-----------------------------------|--------------------------------------------------|

Example 9.1. We return to Example 2.1, which is concerned with a pedestrian who uses a crossing with traffic lights to cross a road with busy traffic safely. Recall that the description of the behaviour of the pedestrian given in Example 2.1 is as follows:

\[ PED = \text{arrive} \cdot (\text{green} \rightarrow \text{cross} + \text{red} \rightarrow (\text{make-req} \cdot (\text{green} \rightarrow \text{cross}))) \, . \]

Let \( h_g \) be such that \( h_g(\text{green}) = \top \) and \( h_g(\text{red}) = ⊥ \); and let \( h_r \) be such that \( h_r(\text{green}) = ⊥ \) and \( h_r(\text{red}) = \top \). Then we can derive the following:

\[ \text{CE}_{h_g}(PED) = \text{arrive} \cdot \text{cross} \quad \text{and} \quad \text{CE}_{h_r}(PED) = \text{arrive} \cdot \text{make-req} \cdot δ \, . \]

So in a world where the traffic light for pedestrians is green he or she will cross the street without making a request for green light; and in a world where the traffic light for pedestrians is red he or she will become completely inactive after making a request for green light. In reality, the request would cause a change from red to green, but the condition evaluation operators \( \text{CE} \) cannot deal with that. We will return to this issue in Example 9.2.

The structural operational semantics of \( \text{ACP}^c \) extended with condition evaluation is described by the transition rules for \( \text{ACP}^c \) and the transition rules given in Table 9.

If \( λ \) is a regular infinite cardinal, the elements of \( C_λ \) can be used to represent equivalence classes with respect to logical equivalence of the set of all propositions with elements of \( C_λ \) as propositional variables and with conjunctions and disjunctions of less than \( λ \) propositions. We write \( P_λ \) for this set of propositions. If \( λ \) is a regular infinite cardinal, it is likely that there is a theory \( Φ \) about the atomic conditions in the shape of a set of propositions. Let \( Φ \subset P_λ \), and let \( h_Φ \in H_λ \) be such that for all \( α, β \in C_λ \):

\[ Φ \vdash \langle \langle h_Φ(α) \rangle \rangle \iff ⟨ ⟨ α ⟩ ⟩ \quad \text{and} \quad h_Φ(α) = h_Φ(β) \iff Φ \vdash \langle ⟨ α ⟩ ⟩ \iff ⟨ ⟨ β ⟩ ⟩ \]  \hspace{1cm} (1)

where \( \langle ⟨ α ⟩ ⟩ \) is a representative of the equivalence class of propositions isomorphic to \( α \). Then we have \( h_Φ(α) = \top \) iff \( \langle ⟨ α ⟩ ⟩ \) is derivable from \( Φ \) and \( h_Φ(α) = ⊥ \) iff \( \neg \langle ⟨ α ⟩ ⟩ \) is derivable from \( Φ \). The image of \( C_λ \) under \( h_Φ \) is \( B \) iff \( Φ \) is a complete theory. If \( Φ \) is not a complete theory, then \( h_Φ \) is not uniquely determined by (1). However, the images of \( C_λ \) under the different endomorphisms satisfying (1) are isomorphic subalgebras of \( C_λ \). Moreover, if both \( h \) and \( h' \) satisfy (1), then \( Φ \vdash \langle ⟨ h(α) \rangle \rangle \iff ⟨ h'(α) ⟩ ⟩ \) for all \( α \in C_λ \).

Below, we show that condition evaluation on the basis of a complete theory can be viewed as substitution on the basis of the theory. That leads us to the use of the following convention: for \( α \in C \), \( α \) stands for an arbitrary closed term of sort \( C \) of which the value in \( C \) is \( α \).
Proposition 9.1 (Condition evaluation on the basis of a theory). Assume that \( \lambda \) is a regular infinite cardinal. Let \( \Phi \subset \mathcal{P}_\lambda \) be a complete theory and let \( p \) be a closed term of \( \text{ACP}^c \). Then \( \text{CE}_{h_{\Phi}}(p) = p' \) where \( p' \) is \( p \) with, for all \( \alpha \in \mathcal{C} \), in all subterms of the form \( \alpha \rightarrow q \), \( \alpha \) replaced by \( \top \) if \( \Phi \vdash \langle \langle \alpha \rangle \rangle \) and \( \alpha \) replaced by \( \bot \) if \( \Phi \vdash \neg \langle \langle \alpha \rangle \rangle \).

Proof. This result follows immediately from the definition of \( h_{\Phi} \) and the distributivity of \( \text{CE}_{h_{\Phi}} \) over all operators of \( \text{ACP}^c \). \( \square \)

In \( \mu\text{CRL} \ [17] \), an extension of ACP which includes conditional expressions, we find a formalization of the substitution-based alternative for \( \text{CE}_{h_{\Phi}} \).

The substitution-based alternative works properly because condition evaluation by means of a \( \lambda \)-complete condition evaluation operator is not dependent on process behaviour. Hence, the result of condition evaluation is globally valid. Below, we will generalize the condition evaluation operators introduced above in such a way that condition evaluation may be dependent on process behaviour. In that case, the result of condition evaluation is in general not globally valid.

Remark 9.1. Assume that \( \lambda \) is a regular infinite cardinal. Let \( h \in \mathcal{H}_{\lambda} \). Then \( h \) induces a theory \( \Phi \subset \mathcal{P}_\lambda \) such that \( h = h_{\Phi} \), viz. the theory \( \Phi \) defined by

\[
\Phi = \{ \langle \langle h(\alpha) \rangle \rangle \leftrightarrow \langle \langle \alpha \rangle \rangle \mid \alpha \in \mathcal{C}_{\lambda} \} \cup \{ \langle \langle \alpha \rangle \rangle \leftrightarrow \langle \langle \beta \rangle \rangle \mid h(\alpha) = h(\beta) \} .
\]

Consequently, if \( \lambda \) is a regular infinite cardinal, condition evaluation by means of the \( \lambda \)-complete condition evaluation operators introduced above is always condition evaluation of which the result can be determined from a set of propositions. We will return to this observation in Section 11.

We proceed with generalizing the condition evaluation operators introduced above. It is assumed that a fixed but arbitrary function \( \text{eff} : \mathcal{A} \times \mathcal{H}_\lambda \rightarrow \mathcal{H}_\lambda \) has been given.

There is a unary generalized \( \lambda \)-complete condition evaluation operator \( \text{GCE}_{h} : \mathcal{P} \rightarrow \mathcal{P} \) for each \( h \in \mathcal{H}_\lambda \); and there is again the unary operator \( \text{CE}_{h} : \mathcal{C} \rightarrow \mathcal{C} \) for each \( h \in \mathcal{H}_\lambda \).

The \( \lambda \)-complete generalized condition evaluation operator \( \text{GCE}_{h} \) allows, given the function \( \text{eff} \), to evaluate conditions dependent of process behaviour. The function \( \text{eff} \) gives, for each action \( a \) and \( \lambda \)-complete endomorphism \( h \), the \( \lambda \)-complete endomorphism \( h' \) that represents the changed results of condition evaluation due to performing \( a \). The function \( \text{eff} \) is extended to \( \mathcal{A}_\delta \) such that \( \text{eff}(\delta, h) = h \) for all \( h \in \mathcal{H}_\lambda \).

The additional axioms for \( \text{GCE}_{h} \), where \( h \in \mathcal{H}_\lambda \), are the axioms given in Table 10 and axioms CE6–CE11 from Table 8.

Example 9.2. We return to Example 2.1, which is concerned with a pedestrian who uses a crossing with traffic lights to cross a road with busy traffic safely. In Example 9.1, we illustrated that the condition evaluation operators \( \text{CE}_{h} \) cannot deal with the change from red light to green light caused by a request for green light. Here, we illustrate that the generalized condition evaluation operators
Table 10. Axioms for generalized condition evaluation ($a \in A_h$)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{GCE}_h(a) = a$</td>
<td>GCE1</td>
</tr>
<tr>
<td>$\text{GCE}<em>h(a \cdot x) = a \cdot \text{GCE}</em>{\text{eff}(a,h)}(x)$</td>
<td>GCE2</td>
</tr>
<tr>
<td>$\text{GCE}_h(x + y) = \text{GCE}_h(x) + \text{GCE}_h(y)$</td>
<td>GCE3</td>
</tr>
<tr>
<td>$\text{GCE}_h(\phi \rightarrow x) = \text{CE}_h(\phi) \rightarrow \text{GCE}_h(x)$</td>
<td>GCE4</td>
</tr>
</tbody>
</table>

Table 11. Transition rules for generalized condition evaluation

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \xrightarrow{[\phi]_a} x'$</td>
<td>$h(\phi) \neq \bot$</td>
</tr>
<tr>
<td>$\text{GCE}_h(x) \xrightarrow{[h(\phi)]<em>a} \text{GCE}</em>{\text{eff}(a,h)}(x')$</td>
<td>$h(\phi) \neq \bot$</td>
</tr>
</tbody>
</table>

$\text{GCE}_h$ can deal with such a change. Let $h_g$ and $h_r$ be as in Example 9.1; and let $\text{eff}$ be such that $\text{eff}(\text{make-req}, h_r) = h_g$ and $\text{eff}(a, h) = h$ otherwise. Then we can derive the following:

$\text{GCE}_{h_g}(\text{PED}) = \text{arrive} \cdot \text{cross}$ ,  
$\text{GCE}_{h_r}(\text{PED}) = \text{arrive} \cdot \text{make-req} \cdot \text{cross}$ .

The change from red light to green light is due to interaction between the pedestrian and the traffic lights. It is clear that this interaction is poorly represented by a generalized condition evaluation operator. We will return to this issue in Example 11.1.

The structural operational semantics of ACP$^c$ extended with generalized condition evaluation is described by the transition rules for ACP$^c$ and the transition rules given in Table 11.

We can add both the $\lambda$-complete condition evaluation operators and the generalized $\lambda$-complete condition evaluation operators to ACP$^c$. However, Proposition 9.2 stated below makes it clear that the latter operators supersede the former operators.

The full splitting bisimulation models of ACP$^c$ with condition evaluation and/or generalized condition evaluation are simply the expansions of the full splitting bisimulation models $\Psi^c_\kappa$ of ACP$^c$, for infinite cardinals $\kappa \leq \lambda$, obtained by associating with each operator $\text{CE}_h$ and/or $\text{GCE}_h$ the corresponding re-labeling operation on conditional transition systems. As mentioned before, full splitting bisimulation models with domain $\text{CTS}_{\kappa/\Xi}$ for $\kappa > \lambda$ are excluded.

The equation $\text{CE}_h(\text{GCE}_{h'}(x)) = \text{GCE}_{h \circ h'}(x)$ is an axiom, but the equation $\text{GCE}_h(\text{GCE}_{h'}(x)) = \text{GCE}_{h \circ h'}(x)$ is not an axiom. The reason is that the latter equation is only valid if $\text{eff}$ satisfies $\text{eff}(a, h \circ h') = \text{eff}((a, h) \circ \text{eff}(a, h'))$ for all $a \in A$ and $h, h' \in H_{\lambda}$.

As their name suggests, the generalized $\lambda$-complete condition evaluation operators are generalizations of the $\lambda$-complete condition evaluation operators.

**Proposition 9.2 (Generalization).** We can fix the function $\text{eff}$ such that $\text{GCE}_h(x) = \text{CE}_h(x)$ for all $h \in H_{\lambda}$.
Proof. Clearly, if $\text{eff}(a,h') = h'$ for all $a \in A$ and $h' \in H_\Lambda$, then $\text{GCE}_h(x) = \text{CE}_h(x)$ for all $h \in H_\Lambda$. \hfill \Box

The $\lambda$-complete state operators that are added to ACP$^c$ in Section 10 are in their turn generalizations of the generalized $\lambda$-complete condition evaluation operators.

We come back to the $\lambda$-complete condition evaluation operators $\text{CE}_h$ for $h \in H_\lambda$. The image of $C_\lambda$ under the $\lambda$-complete endomorphism $h$ is a subalgebra of $C_\lambda$ that is $\lambda$-complete too. For that reason, we could have used $\lambda$-complete homomorphisms to subalgebras that are $\lambda$-complete instead of $\lambda$-complete endomorphisms. It would go beyond the models of the theory developed so far to generalize this in such a way that $\lambda$-complete homomorphisms to $\lambda$-complete Boolean algebras different from subalgebras of $C_\lambda$ are also included.

However, in the case where we consider $\lambda$-complete homomorphisms between free $\lambda$-complete Boolean algebras over different sets of generators, we can relate the models for different choices for $C$ at.

Let $C$ and $C'$ be different choices for $C_{\ast \lambda}$, and let $\Psi^C_k(C)$ and $\Psi^C_k(C')$, for $\kappa \leq \lambda$, be the full splitting bisimulation models $\Psi^C_k$ of ACP$^c$ for the different choices for $C_{\ast \lambda}$. Moreover, let $h$ be a $\lambda$-complete homomorphism from the free $\lambda$-complete Boolean algebra over $C$ to the free $\lambda$-complete Boolean algebra over $C'$. Then $h$ can be extended to a homomorphism $h^*$ from $\Psi^C_k(C)$ to $\Psi^C_k(C')$. This homomorphism is defined by

$$ h^*([((S, \rightarrow, \rightarrow\sqrt{\cdot}, s^0)]_{\omega}) = [[\Gamma(S, \rightarrow', \rightarrow\sqrt{\cdot}', s^0)]_{\omega}, $$

where for every $(\alpha, a) \in C_\omega^C \times A$:

$$ \frac{(\alpha, a)}{\omega}^* = \{(s, s') \mid \exists \beta \bullet s \xrightarrow{[\beta]a} s' \land \alpha = h(\beta)\}, $$

$$ \frac{(\alpha, a)}{\omega}^* = \{s \mid \exists \beta \bullet s \xrightarrow{[\beta]a} s' \land \alpha = h(\beta)\}. $$

It is easy to see that $h^*$ is well-defined and a homomorphism indeed.

Thus, a $\lambda$-complete homomorphism between $\lambda$-complete Boolean algebras over different sets of generators can be used to translate conditions throughout a full splitting bisimulation model for one choice of $C_{\ast \lambda}$ in such a way that a full splitting bisimulation model for a different choice of $C_{\ast \lambda}$ is obtained.

10 State Operators

The state operators make it easy to represent the execution of a process in a state. The basic idea is that the execution of an action in a state has effect on the state, i.e. it causes a change of state. Besides, there is an action left when an action is executed in a state. The operators introduced here generalize the state operators added to ACP in [1]. The main difference with those operators is

\footnote{The interesting cases are those where the cardinalities of $C$ and $C'$ are different. Otherwise, the homomorphisms are isomorphisms.}
Table 12. Axioms for state operators \((a \in A_s, \eta \in C_s, \eta' \in C_s \cup \{\bot, \top}\))

| \(\lambda_s(a) = \text{act}(a, s)\) | SO1 | \(\lambda_s(\bot) = \bot\) | SO5 |
| \(\lambda_s(a \cdot x) = \text{act}(a, s) \cdot \lambda_{\text{eff}}(a, s)(x)\) | SO2 | \(\lambda_s(\top) = \top\) | SO6 |
| \(\lambda_s(x + y) = \lambda_s(x) + \lambda_s(y)\) | SO3 | \(\lambda_s(\eta) = \eta'\) if \(\text{eval}(\eta, s) = \eta'\) | SO7 |
| \(\lambda_s(\phi \cdot x) = \lambda_s(\phi) \rightarrow \lambda_s(x)\) | SO4 | \(\lambda_s(-\phi) = -\lambda_s(\phi)\) | SO8 |
| \(\lambda_s(\phi \sqcup \psi) = \lambda_s(\phi) \sqcup \lambda_s(\psi)\) | SO9 | \(\lambda_s(\phi \sqcap \psi) = \lambda_s(\phi) \sqcap \lambda_s(\psi)\) | SO10 |

Table 13. Transition rules for state operators

\[
\begin{array}{ccl}
\frac{\lambda_s(x)}{\text{act}(a, s) \neq \delta, \text{eval}(\phi, s) \neq \bot} \\
\frac{\lambda_s(x)}{\delta \cdot x} \\
\frac{\lambda_s(x)}{\eta \cdot x} \\
\frac{\lambda_s(x)}{\lambda_{\text{eff}}(a, s)(x')} & \text{act}(a, s) \neq \delta, \text{eval}(\phi, s) \neq \bot}
\end{array}
\]

that guarded commands are taken into account. As in the case of the condition evaluation operators and the generalized condition evaluation operators, these state operators require to fix an infinite cardinal \(\lambda\). By doing so, full splitting bisimulation models with domain \(\text{CTS}_\kappa/\mathbb{Z}\) for \(\kappa > \lambda\) are excluded.

It is assumed that a fixed but arbitrary set \(S\) of states has been given, together with functions \(\text{act} : A \times S \to A_s\), \(\text{eff} : A \times S \to S\) and \(\text{eval} : C_s \times S \to C_s\), where, for each \(s \in S\), the function \(h_s : C_s \to C_s\) defined by \(h_s(\alpha) = \text{eval}(\alpha, s)\) is a \(\lambda\)-complete endomorphism of \(C_s\).

There are unary \(\lambda\)-complete state operators \(\lambda_s : P \to P\) and \(\lambda_s : C \to C\) for each \(s \in S\).

The \(\lambda\)-complete state operator \(\lambda_s\) allows, given the above-mentioned functions, processes to interact with a state. Let \(p\) be a process. Then \(\lambda_s(p)\) is the process \(p\) executed in state \(s\). The function \(\text{act}\) gives, for each action \(a\) and state \(s\), the action that results from executing \(a\) in state \(s\). The function \(\text{eff}\) gives, for each action \(a\) and state \(s\), the state that results from executing \(a\) in state \(s\). The function \(\text{eval}\) gives, for each condition \(\alpha\) and state \(s\), the condition that results from evaluating \(\alpha\) in state \(s\). The functions \(\text{act}\) and \(\text{eff}\) are extended to \(A_s\) such that \(\text{act}(\delta, s) = \delta\) and \(\text{eff}(\delta, s) = s\) for all \(s \in S\).

The additional axioms for \(\lambda_s\), where \(s \in S\), are the axioms given in Table 12. Axioms SO1–SO3 are the axioms for the state operators added to ACP in [1].

The structural operational semantics of ACP\(^c\) extended with state operators is described by the transition rules for ACP\(^c\) and the transition rules given in Table 13.

---

\(^7\) Holding on to the usual conventions leads to the double use of the symbol \(\lambda\): without subscript it stands for an infinite cardinal, and with subscript it stands for a state operator.
The full splitting bisimulation models of $\mathsf{ACP}^c$ with state operators are simply the expansions of the full splitting bisimulation models $\mathcal{P}_\kappa^c$ of $\mathsf{ACP}^c$ obtained by associating with each operator $\lambda$ the corresponding re-labeling operation on conditional transition systems.

We can add, in addition to the $\lambda$-complete state operators, the $\lambda$-complete condition evaluation operators and/or the generalized $\lambda$-complete condition evaluation operators from Section 9 to $\mathsf{ACP}^c$.

We write $\mathcal{P}_\kappa^{c\text{ext}}$ for the expansion of $\mathcal{P}_\kappa^c$ for the $\lambda$-complete condition evaluation operators, the generalized $\lambda$-complete condition evaluation operators and the $\lambda$-complete state operators.

The $\lambda$-complete state operators are generalizations of the generalized $\lambda$-complete condition evaluation operators from Section 9.

**Proposition 10.1 (Generalization).** We can fix $S$, act, eff and eval such that, for some $f : \mathcal{H}_\lambda \rightarrow S$, $\lambda f(h) = \mathsf{GCE}_h(x)$ holds for all $h \in \mathcal{H}_\lambda$ in all full splitting bisimulation models $\mathcal{P}_\kappa^{c\text{ext}}$ with $\kappa \leq \lambda$.

**Proof.** Clearly, if $S = \mathcal{H}_\lambda$, $f$ is the identity function on $\mathcal{H}_\lambda$, and $\text{act}(a,s) = a$, $\text{eff}(a,s) = \mathsf{GCE}_h(x)$ and $\text{eval}(\alpha,s) = f^{-1}(s)(\alpha)$ for all $a \in A$, $s \in S$ and $\alpha \in C$, then $\lambda f(h) = \mathsf{GCE}_h(x)$ holds for all $h \in \mathcal{H}_\lambda$ in all full splitting bisimulation models $\mathcal{P}_\kappa^{c\text{ext}}$ with $\kappa \leq \lambda$. $\Box$

**11 Signal Emission**

In Section 9, we made the observation that, if $\lambda$ is a regular infinite cardinal, condition evaluation by means of the $\lambda$-complete condition evaluation operators $\mathsf{CE}_h$ from that section is always condition evaluation of which the result can be determined from a set of propositions (see Remark 9.1). A similar observation can be made about condition evaluation by means of the generalized $\lambda$-complete condition evaluation operators $\mathsf{GCE}_h$ from that section. In the case of condition evaluation by means of $\mathsf{CE}_h$, the set of propositions determining the result of condition evaluation does not change as a process proceeds. In the case of condition evaluation by means of $\mathsf{GCE}_h$, it may happen that the set of propositions determining the result of condition evaluation changes as a process proceeds. That is, the sets of propositions relevant to a process and its subprocesses may differ. This suggest that condition evaluation can also be dealt with by explicitly associating sets of propositions with processes. The intuition is, then, that all propositions from the set of propositions associated with a process holds at the start of the process.

Clearly, if we restrict ourselves to sets of propositions of cardinality less than a regular infinite cardinal $\lambda$, we can associate elements of $C_\lambda$ with processes instead. In line with [2], the element of $C_\lambda$ associated with a process is called the signal emitted by the process. Because $\bot$ represents the proposition $F$, the proposition that cannot hold at the start of any process, we regard a process with which $\bot$ is associated as an inconsistency. However, in an algebraic setting,
we cannot exclude this inconsistency. Therefore, we consider it to be a special process, which is called the inaccessible process.\textsuperscript{8}

The idea to associate elements of $C^\lambda$ with processes naturally suggests itself in the case where $\lambda$ is a regular infinite cardinal. However, there are no trammels to drop the restriction that $\lambda$ is regular.

All this leads us to an extension of ACP$^c$, called ACP$^{cs}$, with the following additional constants and operators:

- the inaccessible process constant $\perp : P$;
- the binary signal emission operator $\triangledown : C \times P \rightarrow P$.

The axioms of ACP$^{cs}$ are the axioms of ACP$^c$ with axioms CM2–CM3 and GC8–GC10 replaced by axioms CM2S–CM3S and GC8S–GC10S from Table 14, and the additional axioms given in Table 15. Axioms NE1–NE3 and SE1–SE11 have been taken from [3] and axioms GC9S and GC10S have been taken from [3] with subterms of the form $s(x) \triangledown \delta$ replaced by $\partial_\lambda(x)$. Axioms CM2S, CM3S and GC8S differ really from the corresponding axioms in [3] due to the choice of having as the signal emitted by the left merge of two processes, as in the case of the communication merge, always the meet of the signals emitted by the two processes.

Example 11.1. We return to Example 2.1, which is concerned with a pedestrian who uses a crossing with traffic lights to cross a road with busy traffic safely. In

\textsuperscript{8} In [11, 9], this process is rather contradictory called the non-existent process. Its new name was prompted by the fact that after performing an action no process will ever proceed as this process.

### Table 14. Axioms adapted to signal emission ($a \in A_\delta$)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \parallel x = a \cdot x + \partial_\lambda(x)$</td>
<td>CM2S</td>
</tr>
<tr>
<td>$a \cdot x \parallel y = a \cdot (x \parallel y) + \partial_\lambda(y)$</td>
<td>CM3S</td>
</tr>
<tr>
<td>$(\phi \rightarrow x) \parallel y = \phi \rightarrow (x \parallel y) + \partial_\lambda(y)$</td>
<td>GC8S</td>
</tr>
<tr>
<td>$(\phi \rightarrow x) \lvert y = \phi \rightarrow (x \lvert y) + \partial_\lambda(y)$</td>
<td>GC9S</td>
</tr>
<tr>
<td>$x \lvert (\phi \rightarrow y) = \phi \rightarrow (x \lvert y) + \partial_\lambda(x)$</td>
<td>GC10S</td>
</tr>
</tbody>
</table>

### Table 15. Additional axioms for signal emission ($a \in A_\delta$)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + \perp = \perp$</td>
<td>NE1</td>
</tr>
<tr>
<td>$\perp \cdot x = \perp$</td>
<td>NE2</td>
</tr>
<tr>
<td>$a \cdot \perp = \delta$</td>
<td>NE3</td>
</tr>
<tr>
<td>$\top \triangledown x = x$</td>
<td>SE1</td>
</tr>
<tr>
<td>$\perp \triangledown x = \perp$</td>
<td>SE2</td>
</tr>
<tr>
<td>$\phi \triangledown x + y = \phi \triangledown (x + y)$</td>
<td>SE3</td>
</tr>
<tr>
<td>$(\phi \triangledown x) \cdot y = \phi \triangledown x \cdot y$</td>
<td>SE4</td>
</tr>
<tr>
<td>$\partial_H(\phi \triangledown x) = \phi \triangledown \partial_H(x)$</td>
<td>SE11</td>
</tr>
</tbody>
</table>

\textsuperscript{8} In [11, 9], this process is rather contradictory called the non-existent process. Its new name was prompted by the fact that after performing an action no process will ever proceed as this process.
Example 9.2, we illustrated that the generalized condition evaluation operators \( \text{GCE}_h \) poorly represent the interaction between the pedestrian and the traffic lights. Here, we illustrate that such interaction can be better represented with the signal emission operator \( \bullet \). Recall that the description of the behaviour of the pedestrian is as follows:

\[
\text{PED} = \text{arrive} \cdot (\text{green} :\Rightarrow \text{cross} + \text{red} :\Rightarrow (\text{make-req} \cdot (\text{green} :\Rightarrow \text{cross})))
\]

In the case where the light for pedestrians is red, the traffic lights will grant a request for green light. The next action of the traffic lights is to revert to red light. Suppose that initially the light for pedestrians is red. Then the behaviour of the traffic lights can be described as follows:

\[
\text{TL} = (\text{red} \cap -\text{green}) \bullet \text{grant-req} \cdot ((\text{red} \cap -\text{green}) \bullet \text{revert} \cdot \text{TL})
\]

Let the communication function \( | \) be such that \( \text{make-req} | \text{grant-req} = \text{grant-req} \) and \( a | b = \delta \) otherwise. Then we can derive the following about the combined behaviour of the pedestrian and the traffic lights:

\[
\partial_{\{\text{make-req}, \text{grant-req}\}}(\text{PED} \parallel \text{TL}) =
\]

\[
(\text{red} \cap -\text{green}) \bullet \text{arrive} \cdot ((\text{red} \cap -\text{green}) \bullet \text{request} \cdot ((\text{red} \cap -\text{green}) \bullet \text{revert} \cdot \text{TL}))) +
\]

\[
(\text{red} \cap -\text{green}) \bullet \text{arrive} \cdot ((\text{red} \cap -\text{green}) \bullet \text{request} \cdot ((\text{red} \cap -\text{green}) \bullet \text{revert} \cdot ((\text{red} \cap -\text{green}) \bullet \delta))).
\]

The possibility that the combined behaviour ends in deadlock shows that we have actually described a rather simple-minded pedestrian. If he or she has not started crossing the road before the traffic lights revert to red light, the pedestrian takes no action; and consequently the light remains red. This remains unobserved in the case where the interaction between the pedestrian and the traffic lights is represented by the generalized condition evaluation operators \( \text{GCE}_h \).

In the structural operational semantics of ACP\( ^{cs} \), unary relations \( s^\alpha \), one for each \( \alpha \in \mathcal{C} \setminus \{\bot\} \), are used in addition to the relations \( \ell \rightarrow \sqrt{\cdot} \) and \( \ell \rightarrow \cdot \). We write \( s(p) = \alpha \) instead of \( p \in s^\alpha \). The relation \( s^\alpha \) can be explained as follows:

- \( s(p) = \alpha \): \( p \) emits the signal \( \alpha \).

The structural operational semantics of ACP\( ^{cs} \) is described by the transition rules given in Tables 16 and 17. These transition rules include all transition rules from Tables 3 and 5 with additional premises to exclude transitions from or to processes that emit the signal \( \bot \). There are additional transition rules describing the signals emitted by the processes. The transition rules for signal emission are new as well.

The following gives a good picture of the nature of signals and conditions.

**Proposition 11.1 (Signals and conditions).** If \( \llangle \alpha \rrangle \vdash \llangle \beta \rrangle \Rightarrow \llangle \beta' \rrangle \), then \( \alpha \bullet (\beta :\Rightarrow x) = \alpha \bullet (\beta' :\Rightarrow x) \).
The axioms of BA, that (\(\alpha\)) of all finite propositions with elements of algebra of equivalence classes with respect to logical equivalence of the set theorem of propositional calculus and the isomorphism of C (\(\langle \langle \alpha \rangle \rangle \)).

Proof. From these two results, it follows immediately that (\(\alpha\)).

Corollary 11.1. We have the following corollaries from Proposition 11.1.

Table 16. Transition rules for BPA

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s(\bot) = \bot)</td>
<td>(s(\top) = \top)</td>
</tr>
<tr>
<td>(s(\alpha) = \top)</td>
<td>(s(\alpha) = \top)</td>
</tr>
<tr>
<td>(s(x) = \phi)</td>
<td>(s(x + y) = \psi)</td>
</tr>
<tr>
<td>(s(x + y) = \psi)</td>
<td>(s(x) = \phi)</td>
</tr>
<tr>
<td>(s(x + y) = \phi)</td>
<td>(s(x) = \phi)</td>
</tr>
<tr>
<td>(s(\neg \psi \bullet \phi) = \psi \bullet \phi)</td>
<td>(s(\psi \bullet y) = \psi \bullet \phi)</td>
</tr>
</tbody>
</table>

\(\text{Proof.}\) It follows immediately from (\(\langle \alpha \rangle\) \(\vdash\) \(\langle \beta \rangle\) \(\iff\) \(\langle \beta' \rangle\)), using the deduction theorem of propositional calculus and the isomorphism of C and the Boolean algebra of equivalence classes with respect to logical equivalence of the set of all finite propositions with elements of \(\mathcal{C}_\text{at}\) as propositional variables, that \((- (\alpha \land \beta) \land \beta') \land (- (\alpha \land \beta') \land \beta) = \top\). It follows easily from this equation, using the axioms of BA, that \((\alpha \land \beta) \land \beta' = \beta'\) (\(\ast\)) and \((\alpha \land \beta') \land \beta = \beta\) (\(\ast\ast\)). From (\(\ast\)), using the axioms of ACP, we can derive \((\alpha \land \beta' \vdash x) + \alpha \land \beta' \vdash x = \alpha \land \beta' \vdash x\) as follows:

\[
\begin{align*}
\alpha \land \beta' \vdash x & + \alpha \land \beta' \vdash x \\
\text{S6, GC6} & \Rightarrow \alpha \land \beta' \vdash x + \alpha \land \beta' \vdash x \\
\text{SE\& BA, GC7} & \Rightarrow \alpha \land (\alpha \land \beta' \vdash x) \\
\text{\(\equiv\)} & \alpha \land \beta' \vdash x .
\end{align*}
\]

From (\(\ast\ast\)), we can derive analogously \((\alpha \land \beta' \vdash x) + \alpha \land \beta' \vdash x = \alpha \land \beta' \vdash x\).

From these two results, it follows immediately that \((\alpha \land \beta' \vdash x) = \alpha \land \beta' \vdash x\).

We have the following corollaries from Proposition 11.1.

Corollary 11.1. If \(\langle \alpha \rangle\) \(\vdash\) \(\langle \beta \rangle\), then \((\alpha \land \beta' \vdash x) = \alpha \land \beta' \vdash x\). If \(\langle \alpha \rangle\) \(\vdash\) \(\neg \langle \beta \rangle\), then \((\alpha \land \beta' \vdash x) = \alpha \land \beta' \vdash \delta\).
Table 17. Additional transition rules for $\text{ACP}^a$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \vdash x' \rightarrow \top$, $s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
<tr>
<td>$x \vdash x' \rightarrow \top$, $y \vdash s(x</td>
<td></td>
</tr>
</tbody>
</table>

**Corollary 11.2.** If $\text{eff}(h, a)$ is the identity endomorphism on $C$ for all endomorphisms $h$ on $C$ and $a \in A$, then we have $\text{GCE}_h(\langle \alpha \rangle) (\langle \beta \rangle : x) = \beta' : \text{GCE}_h(\langle \alpha \rangle) (x)$ implies $\alpha \times (\beta : x) = \alpha \times (\beta' : x)$. 

30
12 Full Signal-Observing Splitting Bisimulation Models of ACPcs

In this section, we introduce conditional transition systems with signals, signal-observing splitting bisimilarity of conditional transition systems with signals, and the full signal-observing splitting bisimulation models of ACPcs.

Conditional transition systems with signals generalize conditional transition systems. Let $\kappa$ be an infinite cardinal. Then a $\kappa$-conditional transition system with signals $T$ is a tuple $(S, \to, \to\sqrt{}, s, s_0)$ where

- $(S, \to, \to\sqrt{}, s, s_0)$ is a $\kappa$-conditional transition system;
- $s$ is a function from $S$ to $C_{\kappa}$;
- and for all $\ell \in C_{\kappa} \times A$:
  - $\{ (s, s') \in \ell | s(s) = \bot \lor s(s') = \bot \} = \emptyset$;
  - $\{ s \in \ell | s(s) = \bot \} = \emptyset$.

We say that $s(s)$ is the signal emitted by the state $s$.

For conditional transition systems with signals, reachability and connectedness are defined exactly as for conditional transition systems.

Let $(S, \to, \to\sqrt{}, s, s_0)$ be a $\kappa$-conditional transition system with signals (for an infinite cardinal $\kappa$) that is not necessarily connected. Then the connected part of $T$, written $\Gamma(T)$, is simply defined as follows:

$$\Gamma(T) = (S', \to', \to\sqrt{}, s', s_0'),$$

where

$$(S', \to', \to\sqrt{}, s_0') = \Gamma(S, \to, \to\sqrt{}, s_0'),$$

$s'$ is the restriction of $s$ to $S'$.

Let $\kappa$ be an infinite cardinal. Then $\text{CTS}_\kappa^s$ is the set of all $\kappa$-conditional transition systems with signals $(S, \to, \to\sqrt{}, s, s_0)$ for which $(S, \to, \to\sqrt{}, s, s_0) \in \text{CTS}_\kappa$.

Isomorphism between conditional transition systems with signals is defined as between conditional transition systems, but with the additional condition that $s_1(s) = s_2(b(s))$. Splitting bisimilarity has to be adapted to the setting with signals.

Let $T_1 = (S_1, \to_1, \to\sqrt{1}, s_1, s_0^1)$ $\in \text{CTS}_\kappa^s$, $T_2 = (S_2, \to_2, \to\sqrt{2}, s_2, s_0^2) \in \text{CTS}_\kappa^s$ (for an infinite cardinal $\kappa$). Then a signal-observing splitting bisimulation $B$ between $T_1$ and $T_2$ is a binary relation $B \subseteq S_1 \times S_2$ such that $B(s_1^0, s_0^2)$ and for all $s_1, s_2$ such that $B(s_1, s_2)$:

- $s_1(s_1) = s_2(s_2)$;
- if $s_1 \frac{[\alpha]}{s_1} s_1'$, then there is a set $CS_2' \subseteq C_{\kappa} \times S_2$ of cardinality less than $\kappa$ such that $s_1(s_1) \cap \alpha \subseteq \bigcup \text{dom}(CS_2')$ and for all $(\alpha', s_2') \in CS_2'$, $s_2 \frac{[\alpha']}{s_2} s_2'$ and $B(s_1', s_2')$.
and encapsulation. mand, signal emission, parallel composition, left merge, communication merge with respect to alternative composition, sequential composition, guarded composition, serving splitting bisimilar, written witnessing bisimulation between splitting bisimulation \( B \) of \( T_1 \), if there exists a signal-observing bisimulation \( B \) between \( T_1 \) and \( T_2 \). Let \( B \) be a signal-observing splitting bisimulation between \( T_1 \) and \( T_2 \). Then we say that \( B \) is a splitting signal-observing bisimulation witnessing splitting bisimulation \( T_1 \triangleleft B T_2 \).

It is straightforward to see that \( \triangleleft B \) is an equivalence on \( \text{CTS}_\kappa^\varepsilon \). Let \( T \in \text{CTS}_\kappa^\varepsilon \). Then we write \([T]_\triangleleft B\) for \( \{T' \in \text{CTS}_\kappa^\varepsilon \mid T \triangleleft B T'\} \), i.e. the \( \triangleleft B \)-equivalence class of \( T \). We write \( \text{CTS}_\kappa^\varepsilon / \triangleleft B \) for the set of equivalence classes \( \{[T]_\triangleleft B \mid T \in \text{CTS}_\kappa^\varepsilon \} \).

The elements of \( \text{CTS}_\kappa^\varepsilon \) and operations on \( \text{CTS}_\kappa^\varepsilon \) to be associated with the constants and operators of \( \text{ACP}^\varepsilon \) are as the elements of \( \text{CTS}_\kappa \) and operations on \( \text{CTS}_\kappa \) associated with them, but with all signals \( \triangleright \) and \( \triangleleft \) restricted to states that emit a signal different from \( \perp \) and with the additional function \( s \) as suggested by the structural operational semantics of \( \text{ACP}^\varepsilon \).

We associate with the additional constant \( \perp \) an element \( \hat{\perp} \) of \( \text{CTS}_\kappa^\varepsilon \) and with the additional operator \( \hat{\wedge} \) an operation \( \hat{\wedge} \) on \( \text{CTS}_\kappa^\varepsilon \) as follows.

- \( \hat{\perp} = (\{s_0\}, \emptyset, \emptyset, s_0) \), where \( s(s_0) = \perp \).
- Let \( T = (S, \rightarrow, \rightarrow \triangleright, s, s^0) \in \text{CTS}_\kappa^\varepsilon \). Then
  \[
  \alpha \hat{\wedge} T = \Gamma(S, \rightarrow', \rightarrow \triangleright', s', s^0),
  \]
  where
  \[
  s'(s) = s(s), \quad \text{for } s \in S \setminus \{s_0\},
  \\
  s'(s_0) = \alpha \cap s(s_0),
  \]
  and for every \( (\alpha, a) \in C_\kappa^\varepsilon \times A \):
  \[
  \frac{(\alpha, a)}{\rightarrow} = \{(s, s') \mid s \triangleright a s' \wedge s'(s) \neq \perp \wedge s'(s') \neq \perp\},
  \\
  \frac{(\alpha, a)}{\triangleright} = \{s \mid s \triangleright a \triangleright s' \neq \perp\}.
  \]

We can easily show that signal-observing splitting bisimilarity is a congruence with respect to alternative composition, sequential composition, guarded command, signal emission, parallel composition, left merge, communication merge and encapsulation.
Proposition 12.1 (Congruence). Let \( \kappa \) be an infinite cardinal. Then for all \( T_1, T_2, T'_1, T'_2 \in \text{CTS}_\kappa \), \( T_1 \cong^s T'_1 \) and \( T_2 \cong^s T'_2 \) imply \( T_1 \parallel T_2 \cong^s T'_1 \parallel T'_2 \), \( T_1 \triangleleft T_2 \cong^s T'_1 \triangleleft T'_2 \), \( \alpha \vdash T_1 \cong^s \alpha \vdash T'_1 \), \( \alpha \triangleleft^s T_1 \cong^s \alpha \triangleleft^s T'_1 \), \( T_1 \parallel T_2 \cong^s T'_1 \parallel T'_2 \). The ingredients of the full signal-observing splitting bisimulation models \( \Psi^s_\kappa \) of ACP\( ^s_\kappa \), one for each infinite cardinal \( \kappa \), are defined as follows:

\[
\begin{align*}
\mathcal{P} &= \text{CTS}_\kappa / \cong^s, \\
\parallel^s &= [\parallel^s]_{\cong^s}, \\
\triangleleft^s &= [\triangleleft^s]_{\cong^s}, \\
\triangleleft_{\kappa}^s &= [\triangleleft_{\kappa}^s]_{\cong^s}, \\
\alpha \vdash^s T_1 &= [\alpha \vdash^s T_1]_{\cong^s}, \\
\alpha \triangleleft_{\kappa}^s T_1 &= [\alpha \triangleleft_{\kappa}^s T_1]_{\cong^s}, \\
\alpha \oplus^s T_1 &= [\alpha \oplus^s T_1]_{\cong^s}, \\
\alpha \circ^s T_1 &= [\alpha \circ^s T_1]_{\cong^s}, \\
\alpha \mu^s T_1 &= [\alpha \mu^s T_1]_{\cong^s}, \\
\alpha \vartheta^s T_1 &= [\alpha \vartheta^s T_1]_{\cong^s}, \\
\alpha \chi^s T_1 &= [\alpha \chi^s T_1]_{\cong^s}.
\end{align*}
\]

Alternative composition, sequential composition, guarded command, parallel composition, left merge, communication merge and encapsulation on \( \text{CTS}_\kappa / \cong^s \) are well-defined because \( \cong^s \) is a congruence with respect to the corresponding operations on \( \text{CTS}_\kappa \).

The structures \( \Psi^s_\kappa \) are models of ACP\( ^s_\kappa \).

Theorem 12.1 (Soundness of ACP\( ^s_\kappa \)). For each infinite cardinal \( \kappa \), we have \( \Psi^s_\kappa \models \text{ACP}^s_\kappa \).

Proof. Because \( \cong^s \) is a congruence, it is sufficient to show that all axioms are sound. The soundness of all axioms follows straightforwardly from the definitions of the ingredients of \( \Psi^s_\kappa \).  

\( \square \)
Table 18. Additional axioms for retrospection operator \((a \in A)\)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sim \bot = \bot)</td>
<td>R1</td>
</tr>
<tr>
<td>(\sim \top = \top)</td>
<td>R2</td>
</tr>
<tr>
<td>(\sim (-\phi) = -(-\phi))</td>
<td>R3</td>
</tr>
<tr>
<td>(\sim (\phi \uplus \psi) = \sim \phi \uplus \sim \psi)</td>
<td>R4</td>
</tr>
<tr>
<td>(\sim (\phi \sqcap \psi) = \sim \phi \sqcap \sim \psi)</td>
<td>R5</td>
</tr>
<tr>
<td>(a \cdot (\sim \phi :\rightarrow x) = \phi :\rightarrow a \cdot x + -\phi :\rightarrow a \cdot \delta)</td>
<td>R6</td>
</tr>
</tbody>
</table>

13 BPA with Retrospective Conditions

In this section, we present an extension of BPA\(_c\)\(_\delta\) with a retrospection operator on conditions. The retrospection operator allows for looking back on conditions under which preceding actions have been performed. The extension of BPA\(_c\)\(_\delta\) with the retrospection operator is called BPA\(_c\)\(_r\)\(_\delta\). In Section 15, we will add parallel composition and encapsulation to BPA\(_c\)\(_r\)\(_\delta\).

BPA\(_c\)\(_r\)\(_\delta\) has the constants and operators of BPA\(_c\)\(_\delta\) and in addition:

- the unary retrospection operator \(\sim: C \rightarrow C\).

The axioms of BPA\(_c\)\(_r\)\(_\delta\) are the axioms of BPA\(_c\)\(_\delta\) and the additional axioms given in Table 18. The crucial axiom is R6, which shows that a conditional expression of the form \(\sim \zeta :\rightarrow p\) gives a retrospection at the condition under which the immediately preceding action has been performed.

Recall that we write \(p \triangleleft \zeta \triangleright q\) for \(\zeta :\rightarrow p + -\zeta :\rightarrow q\). An interesting equation is \(a \cdot (x \triangleleft \sim \phi \triangleright y) = a \cdot x \triangleleft \phi \triangleright a \cdot y\). This equation is a generalization of axiom R6: axiom R6 is derivable from the other axioms of BPA\(_c\)\(_r\)\(_\delta\) and this equation by substituting \(\delta\) for \(y\) and applying axioms GC3 and A6. It is not immediately clear that this equation is derivable from the axioms of BPA\(_c\)\(_r\)\(_\delta\).

Proposition 13.1 (Derivability Generalization Axiom R6). The equation \(a \cdot (x \triangleleft \sim \phi \triangleright y) = a \cdot x \triangleleft \phi \triangleright a \cdot y\) (R6’\) is derivable from the axioms of BPA\(_c\)\(_r\)\(_\delta\).

Proof. We can make the following derivation:

\[
\begin{align*}
a \cdot (x \triangleleft \sim \phi \triangleright y) &\quad \text{BA2,GC1} \\
(\phi \uplus -\phi) :\rightarrow a \cdot (x \triangleleft \sim \phi \triangleright y) &\quad \text{GC7} \\
\phi :\rightarrow a \cdot (x \triangleleft \sim \phi \triangleright y) + -\phi :\rightarrow a \cdot (x \triangleleft \sim \phi \triangleright y) &\quad \text{A6,GC2,BAG,GA,GCS,GC4} \\
\phi :\rightarrow (\phi :\rightarrow a \cdot (x \triangleleft \sim \phi \triangleright y) + -\phi :\rightarrow a \cdot \delta) + &\quad \phi :\rightarrow (\sim \phi :\rightarrow a \cdot (x \triangleleft \sim \phi \triangleright y) + \phi :\rightarrow a \cdot \delta).
\end{align*}
\]

Hence, if we can derive \(\phi :\rightarrow (\phi :\rightarrow a \cdot (x \triangleleft \sim \phi \triangleright y) + -\phi :\rightarrow a \cdot \delta) = \phi :\rightarrow a \cdot x\) (*) and \(-\phi :\rightarrow (\sim \phi :\rightarrow a \cdot (x \triangleleft \sim \phi \triangleright y) + \phi :\rightarrow a \cdot \delta) = -\phi :\rightarrow a \cdot y\) (**), then it follows immediately that we can derive R6’. We can derive (*) as follows:
\[ \phi \rightarrow (\phi \rightarrow a \cdot (x \triangleleft \neg \phi \triangleright y) + \neg \phi \rightarrow a \cdot \delta) \]

\[ \frac{G \phi, G \phi, \Gamma}{R_6} \]

\[ \phi \rightarrow a \cdot ((\neg \phi \rightarrow (x \triangleleft \neg \phi \triangleright y)) \]

\[ \frac{G \phi, G \phi, \Gamma}{R_6} \]

\[ \phi \rightarrow (\phi \rightarrow a \cdot x + \neg \phi \rightarrow a \cdot \delta) \]

\[ \frac{G \phi, G \phi, \Gamma}{R_6} \]

\[ \phi \rightarrow a \cdot x \]

We can derive (**) analogously. \[ \square \]

**Example 13.1.** We return to Example 2.1, which is concerned with a pedestrian who uses a crossing with traffic lights to cross a road with busy traffic safely. Recall that the description of the behavior of the pedestrian given in Example 2.1 is as follows:

\[ \text{PED} = \text{arrive} \cdot (\text{green} \rightarrow \text{cross} + \text{red} \rightarrow (\text{make-req} \cdot (\text{green} \rightarrow \text{cross}))) \]

This description concerns a pedestrian who does not act unthinkingly. Now consider a pedestrian who does act unthinkingly. When this pedestrian arrives at the crossing, he or she first makes a request for green light and then crosses the street unconditionally if the light for pedestrians was green on arrival and crosses the street when the light for pedestrians has changed if it was red on arrival. This behavior can be described in BPA\(_{cr}\delta\) as follows:

\[ \text{PED}' = \text{arrive} \cdot \text{make-req} \cdot (\neg \text{green} \rightarrow \text{cross} + \neg \text{red} \rightarrow (\text{green} \rightarrow \text{cross})) \]

Because of the addition of the retrospection operator, we cannot use the Boolean algebras \(C_\kappa\) here. The algebras \(C'_\kappa\) that we use here can be characterized as the free \(\kappa\)-complete algebras over \(C_{at}\) from the class of algebras with interpretations for the constants and operators of Boolean algebras and the retrospection operator that satisfy the axioms of Boolean algebras (Table 1) and axioms R1–R5 from Table 18. We do not make this fully precise, but give an explicit construction of the algebras \(C'_\kappa\) instead. Important to bear in mind is that not only the atomic conditions, but also the results of applying the operation associated with the retrospection operator a finite number of times to atomic conditions, should not satisfy any equations except those derivable from the axioms.

Let \(C'_{at} = \bigcup \{C_{at} \times \{i\} \mid i \in \omega\}\) and define prev : \(C'_{at} \rightarrow C'_{at}\) by prev\((\eta, i) = (\eta, i + 1)\). For any infinite cardinal \(\kappa\), let \(C'_\kappa\) be the free \(\kappa\)-complete Boolean algebra over \(C'_{at}\). Then the function prev extends to a unique \(\kappa\)-complete endomorphism prev* of \(C'_\kappa\). This endomorphism is a unary operation on \(C'_\kappa\) that satisfies axioms R1–R5 from Table 18 and preserves \(\bigcup C''\) for every \(C'' \subseteq C'_\kappa\) of cardinality less than \(\kappa\). The algebra \(C''_{\kappa}\) is the expansion of \(C'_\kappa\) obtained by associating the operation prev* with the operator \(\sim\). We write \(C''\) for \(C''_{\kappa}\).

The structural operational semantics of BPA\(_{cr}\delta\) differs only from the structural operational semantics of BPA\(_{\kappa}\) in the conditions involved. They are now taken from \(C''\) instead of \(C\).
14 Full Retrospective Splitting Bisimulation Models of BPA^{cr}_δ

The construction of the full splitting bisimulation models of BPA^{cr}_δ differs from the construction of the full splitting bisimulation models of BPA^{cr}_δ in the conditions involved and in the notion of splitting bisimulation used. The conditions are now taken from C^κ_δ instead of C^κ. Henceforth, we write C^κ_δ for C^κ \ \{\{\perp}\}. Let κ be an infinite cardinal. Then a κ-conditional transition system with retrospection T consists of the following:

- a set S of states;
- a set $\frac{\ell}{\sqrt{\kappa}} \subseteq S \times S$, for each $\ell \in C^κ_δ \times A$;
- a set $\frac{\ell}{\sqrt{\kappa}} \subseteq S$, for each $\ell \in C^κ_δ \times A$;
- an initial state $s^0 \in S$.

For conditional transition systems with retrospection, reachability, connectedness and connected part are defined exactly as for conditional transition systems.

Let κ be an infinite cardinal. Then CTS^κ BPA is the set of all connected κ-conditional transition systems with retrospection $T = (S, \rightarrow, \rightarrow, s^0)$ such that $S \subseteq S_κ$ and the branching degree of $T$ is less than κ.

Isomorphism between conditional transition systems with retrospection is defined exactly as for conditional transition systems. Splitting bisimilarity has to be adapted to the setting with retrospection.

Let $T_1 = (S_1, \rightarrow_1, \rightarrow, s_1^0) \in \text{CTS}^κ BPA$ and $T_2 = (S_2, \rightarrow_2, \rightarrow, s_2^0) \in \text{CTS}^κ BPA$ (for an infinite cardinal κ). Then a retrospective splitting bisimulation $B$ between $T_1$ and $T_2$ is a ternary relation $B \subseteq S_1 \times C^κ_δ \times S_2$ such that $B(s_1^0, T, s_2^0)$ and for all $s_1, β, s_2$ such that $B(s_1, β, s_2)$:

- if $s_1 \xrightarrow{α} a s_1'$, then there is a set $CS'_1 \subseteq C^κ_δ \times S_2$ of cardinality less than κ such that $α \cap \beta \subseteq \bigcup \text{dom}(CS'_2)$ and for all $(α', s_2') ∈ CS'_2$, $s_2 \xrightarrow{[α'] a} s_2'$ and $B(s_1', α', s_2')$;
- if $s_2 \xrightarrow{α} a s_2'$, then there is a set $CS'_1 \subseteq C^κ_δ \times S_1$ of cardinality less than κ such that $α \cap \beta \subseteq \bigcup \text{dom}(CS'_1)$ and for all $(α', s_1') ∈ CS'_1$, $s_1 \xrightarrow{[α'] a} s_1'$ and $B(s_1', α', s_2')$;
- if $s_1 \xrightarrow{α} a s_1'$, then there is a set $C' \subseteq C^κ_δ$ of cardinality less than κ such that $α \cap \beta \subseteq \bigcup C'$ and for all $α' ∈ C'$, $s_2 \xrightarrow{[α'] a} s_2'$;
- if $s_2 \xrightarrow{α} a s_2'$, then there is a set $C' \subseteq C^κ_δ$ of cardinality less than κ such that $α \cap \beta \subseteq \bigcup C'$ and for all $α' ∈ C'$, $s_1 \xrightarrow{[α'] a} s_1'$.

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


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

The elements of \( \text{CTS}_k \) and operations on \( \text{CTS}_k \) to be associated with the constants and operators of \( \text{BPA}_s^k \) are defined exactly as the elements of \( \text{CTS}_k \) and operations on \( \text{CTS}_k \) associated with them.

Below, we show that retrospective splitting bisimilarity is a congruence with respect to alternative composition, sequential composition and guarded command. That leads us to the use of the notion of a layered retrospective splitting bisimulation.

Let \( T = (S, \rightarrow, \rightarrow^\vee, s^0) \in \text{CTS}_k \) (for an infinite cardinal \( \kappa \)). Then the reachability in \( n \) steps relations of \( T \), one for each \( n \in \mathbb{N} \), are the smallest relations \( \overset{n}{\rightarrow} \subseteq S \times S \) such that:

\[
\begin{align*}
- & s \overset{0}{\rightarrow} s; \\
- & \text{if } s \overset{\hat{I}}{\rightarrow} s' \text{ and } s' \overset{\hat{I}}{\rightarrow} s'' \text{, then } s \overset{n+1}{\rightarrow} s''.
\end{align*}
\]

Let \( T_1 = (S_1, \rightarrow_1, \rightarrow_1^\vee, s_1^0) \in \text{CTS}_k \) and \( T_2 = (S_2, \rightarrow_2, \rightarrow_2^\vee, s_2^0) \in \text{CTS}_k \); and let \( B \) be a retrospective splitting bisimulation between \( T_1 \) and \( T_2 \). Then \( B \) is called a layered retrospective splitting bisimulation if for all \( s_1 \in S_1, s_2 \in S_2 \) and \( \alpha \in C^*, B(s_1, \alpha, s_2) \) implies that \( s_1^1 \overset{\hat{I}}{\rightarrow} s_1 \text{ iff } s_2^1 \overset{n_2}{\rightarrow} s_2 \) for all \( n \in \mathbb{N} \).

**Lemma 14.1 (Existence layered retrospective splitting bisimulation).** Let \( T_1 = (S_1, \rightarrow_1, \rightarrow_1^\vee, s_1^0) \in \text{CTS}_k \) and \( T_2 = (S_2, \rightarrow_2, \rightarrow_2^\vee, s_2^0) \in \text{CTS}_k \) (for an infinite cardinal \( \kappa \)). Then \( T_1 \overset{\sim'}{\rightarrow} T_2 \) implies that there exists a layered retrospective splitting bisimulation witnessing \( T_1 \overset{\sim'}{\rightarrow} T_2 \).

**Proof.** Let \( B \) be a retrospective splitting bisimulation witnessing \( T_1 \overset{\sim'}{\rightarrow} T_2 \). Then we construct a relation \( B' \) as follows: \( B' = \{ (s_1, \alpha, s_2) \mid B(s_1, \alpha, s_2) \land \forall n \in \mathbb{N} \cdot s_1^1 \overset{n_1}{\rightarrow} s_1 \Leftrightarrow s_2^1 \overset{n_2}{\rightarrow} s_2 \} \). It is easy to see that \( B' \) is a retrospective splitting bisimulation witnessing \( T_1 \overset{\sim'}{\rightarrow} T_2 \) as well.

**Proposition 14.1 (Congruence).** Let \( \kappa \) be an infinite cardinal. Then for all \( T_1, T_2, T'_1, T'_2 \in \text{CTS}_k \), \( T_1 \overset{\sim'}{\rightarrow} T'_1 \) and \( T_2 \overset{\sim'}{\rightarrow} T'_2 \) imply \( T_1 \overset{\sim'}{\rightarrow} T'_1 \overset{\sim'}{\rightarrow} T'_2 \), \( T_1 \overset{\sim'}{\rightarrow} T'_2 \overset{\sim'}{\rightarrow} T'_1 \overset{\sim'}{\rightarrow} T'_2 \), \( T_1 \overset{\sim'}{\rightarrow} T'_2 \overset{\sim'}{\rightarrow} T'_1 \overset{\sim'}{\rightarrow} T'_2 \), \( T_1 \overset{\sim'}{\rightarrow} T'_2 \overset{\sim'}{\rightarrow} T'_1 \overset{\sim'}{\rightarrow} T'_2 \), \( \alpha \overset{\sim'}{\rightarrow} T_1 \overset{\sim'}{\rightarrow} T_2 \), \( \alpha \overset{\sim'}{\rightarrow} T_2 \overset{\sim'}{\rightarrow} T_1 \). Let \( R_1 \) and \( R_2 \) be layered retrospective splitting bisimulations witnessing \( T_1 \overset{\sim'}{\rightarrow} T'_1 \) and \( T_2 \overset{\sim'}{\rightarrow} T'_2 \), respectively. Then we construct relations \( R_{+} \), \( R_{\cdot} \) and \( R_{-\cdot} \) as follows:

\[
\begin{align*}
- & R_{+} = \{ (s^0, T_1, s^{0'}) \} \cup \{ \mu_1(R_1) \cup \mu_2(R_2) \} \cap (S \times C^*_k \times S'), \\
- & R_{\cdot} = \{ \mu_1(R_1) \cup \mu_2(R_2) \} \cap (S \times C^*_k \times S'), \\
- & R_{-\cdot} = \{ \mu_1(R_1) \cup \mu_2(R_2) \} \cap (S \times C^*_k \times S'), \\
\end{align*}
\]

\( R_2(s, \alpha', s') \land \exists s_1 \in S_1, a \in A \cdot s_1 \overset{[a]}{\rightarrow} S \land s_2 \overset{n}{\rightarrow} s \);
splitting bisimulations witnessing $T_R$ command, it is straightforward to see that the definitions of alternative composition, sequential composition and guarded notation denote both the injection of $\mu_i$. Because $P$ processes is performed, the retrospections of the other process that are not internal in the parallel composition of two processes, when an action of one of the processes is performed, the retrospections of the other process that are not internal

Here, we write $\mu_i(R_i)$ for $\{(\mu_i(s), \alpha, \mu_i(s')) \mid R_i(s, \alpha, s')\}$, where $\mu_i$ is used to denote both the injection of $S_i$ into $S_1 \cup S_2$ and the injection of $S_i'$ into $S_1' \cup S_2'$. The notation $\sim_0(\alpha) = \alpha$ and $\sim^+(\alpha) = \sim((\sim_0(\alpha))$. Given the definitions of alternative composition, sequential composition and guarded command, it is straightforward to see that $R_{\sim^r}$, $R_{\sim^r}$ and $R_{\sim^r}$ are retrospective splitting bisimulations witnessing $T_1 \sim^r T_2 \bowtie T_1' \sim^r T_2'$, $T_1 \sim^r T_2, T_1' \sim^r T_2'$ and $\alpha \bowtie_1 T_1 \bowtie_1 T_1'$, respectively.

Note that, in the proof of Proposition 14.1, showing that $R_{\sim^r}$ and $R_{\sim^r}$ are witnesses needs the assumption that $R_1$ and $R_2$ are layered.

The ingredients of the full retrospective splitting bisimulation models $\Psi_\kappa^{cr}$ of $BPA^{cr}_\kappa$, one for each infinite cardinal $\kappa$, are defined as follows:

- $P = \mathbb{CTS}_\kappa / \bowtie_1$,
- $[T_1]_\bowtie_1 \bowtie_1 [T_2]_\bowtie_1 = [T_1 \bowtie_1 T_2]_\bowtie_1$,
- $\delta' = [\delta']_\bowtie_1$,
- $\alpha \bowtie_1 [T_1]_\bowtie_1 = [\alpha \bowtie_1 T_1]_\bowtie_1$.

The operations alternative composition, sequential composition and guarded command on $\mathbb{CTS}_\kappa / \bowtie_1$ are well-defined because $\bowtie_1$ is a congruence with respect to the corresponding operations on $\mathbb{CTS}_\kappa$.

The structures $\Psi_\kappa^{cr}$ are models of $BPA^{cr}_\kappa$.

**Theorem 14.1 (Soundness of $BPA^{cr}_\kappa$).** For each infinite cardinal $\kappa$, we have $\Psi_\kappa^{cr} \models BPA^{cr}_\kappa$.

**Proof.** Because $\bowtie_1$ is a congruence, it is sufficient to show that all axioms are sound. The soundness of all axioms follows straightforwardly from the definitions of the ingredients of $\Psi_\kappa^{cr}$. $\square$

## 15 ACP with Retrospective Conditions

We proceed with adding parallel composition and encapsulation operators to $BPA^{cr}_\kappa$, resulting in ACP^{cr}.

For ACP^{cr}, we need the following auxiliary operators:

- the unary retrospection shift operator $\Pi^+ : P \rightarrow P$;
- for each $n \in \mathbb{N}$, the unary restricted retrospection shift operator $\Pi^+_n : P \rightarrow P$;
- for each $n \in \mathbb{N}$, the unary restricted retrospection shift operator $\Pi^+_n : C \rightarrow C$.

In the parallel composition of two processes, when an action of one of the processes is performed, the retrospection of the other process that are not internal
Table 19. Axioms adapted to retrospection \((a \in A_δ)\)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a \parallel x = a \cdot \Pi^+(x))</td>
<td>CM2R</td>
</tr>
<tr>
<td>(a \cdot x \parallel y = a \cdot (x \parallel \Pi^+(y)))</td>
<td>CM3R</td>
</tr>
</tbody>
</table>

Table 20. Additional axioms for retrospection \((a \in A_δ, \eta \in C_η)\)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Pi^+(x) = \Pi^+_{&gt;0}(x))</td>
<td>RS0</td>
</tr>
<tr>
<td>(\Pi^+_{&gt;n}(a) = a)</td>
<td>RS1</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(a \cdot x) = a \cdot \Pi^+</em>{&gt;n+1}(x))</td>
<td>RS2</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(x + y) = \Pi^+</em>{&gt;n}(x) + \Pi^+_{&gt;n}(y))</td>
<td>RS3</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\phi :\rightarrow x) = \Pi^+</em>{&gt;n}(\phi) :\rightarrow \Pi^+_{&gt;n}(x))</td>
<td>RS4</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\phi \sqcap \psi) = \Pi^+</em>{&gt;n}(\phi) \sqcap \Pi^+_{&gt;n}(\psi))</td>
<td>RS5</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\phi \sqcup \psi) = \Pi^+</em>{&gt;n}(\phi) \sqcup \Pi^+_{&gt;n}(\psi))</td>
<td>RS6</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\neg \phi) = \neg \Pi^+</em>{&gt;n}(\phi))</td>
<td>RS7</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n+1}(\neg \phi) = \neg \Pi^+</em>{&gt;n}(\phi))</td>
<td>RS8</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\phi) = \Pi^+</em>{&gt;n}(\phi))</td>
<td>RS9</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\phi) = \Pi^+</em>{&gt;n}(\phi))</td>
<td>RS10</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\phi) = \Pi^+</em>{&gt;n}(\phi))</td>
<td>RS11</td>
</tr>
<tr>
<td>(\Pi^+<em>{&gt;n}(\phi) = \Pi^+</em>{&gt;n}(\phi))</td>
<td>RS12</td>
</tr>
</tbody>
</table>

Table 21. Transition rules adapted to retrospection

\[ x \parallel y \parallel \phi \parallel a \parallel \Pi^+(y) \]
\[ x \parallel y \parallel \phi \parallel a \parallel \Pi^+(x) \]
\[ x \parallel y \parallel \phi \parallel a \parallel x' \]
\[ x \parallel y \parallel \phi \parallel a \parallel \Pi^+(y) \parallel y' \]
\[ x \parallel y \parallel \phi \parallel a \parallel x' \parallel \Pi^+(y) \]

should go one step further. This is accomplished by the retrospection shift operator. The restricted retrospection shift operators, on processes and conditions, are needed for the axiomatization of the retrospection shift operator. The retrospection shift operator \(\Pi^+\) is similar to the history pointer shift operator \(hps\) from [4].

The axioms of ACP\(_{cr}\) are the axioms of ACP\(_c\) with axioms CM2–CM3 replaced by axioms CM2R–CM3R from Table 19, and the additional axioms for retrospection given in Table 20. Axioms CM2R and CM3R show that retrospections are adapted if two processes proceed in parallel. Axioms RS0–RS12 state that this happens as explained above. By means of axioms RS5–RS12, the retrospection shift operators on conditions can be eliminated from all terms of sort C.

The structural operational semantics of ACP\(_{cr}\) is described by the transition rules for ACP\(_c\) with the first four transition rules for parallel composition and the two transition rules for left merge replaced by the transition rules given in Table 21, and the additional transition rules for retrospection given in Table 22.
Table 22. Additional transition rules for retrospection

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \xrightarrow{[\phi]} x'$</td>
<td>$\Pi^+(x) \xrightarrow{[\Pi^+_n(\phi)]} \Pi^+_1(x')$</td>
<td>(\Pi^+_n(\phi)) (a) \xrightarrow{\square} (\Pi^+_1) (a) (x') (\Pi^+_n(\phi)) (x')</td>
</tr>
<tr>
<td>$\Pi^+_n(x) \xrightarrow{[\Pi^+_n(\phi)]} \Pi^+_n(x')$</td>
<td>$x \xrightarrow{[\phi]} x'$</td>
<td>(\Pi^+_n(\phi)) (a) \xrightarrow{\square} (\Pi^+_n) (a) (x') (\Pi^+_n) (a) (x')</td>
</tr>
</tbody>
</table>

16 Full Retrospective Splitting Bisimulation Models of ACP\(\text{cr}\)

In this section, we expand the full retrospective splitting bisimulation models of BPA\(\text{cr}\) to ACP\(\text{cr}\). The operations on CTS\(\text{cr}\) that we associate with most of the additional operators of ACP\(\text{cr}\) call for unfolding of transition systems from CTS\(\text{cr}\).

For the sake of unfolding, it is assumed that, for each infinite cardinal \(\kappa\), \(S_{\kappa}\) has the following closure property:

\[
\forall S \subseteq S_{\kappa}, \{\pi \bowtie \langle s \rangle \mid \pi \in (S \times (C_{\kappa} \times A))^+, s \in S\} \subseteq S_{\kappa}.
\]

We write \(P'(S)\) for the set \(\{\pi \bowtie \langle s \rangle \mid \pi \in (S \times (C_{\kappa} \times A))^+, s \in S\}\). The function \(\# : P'(S) \to \mathbb{N}\) is defined by

\[
\#(\langle s \rangle) = 0,
\]

\[
\#(\pi \bowtie \langle s, \ell, s' \rangle) = \#(\pi \bowtie \langle s \rangle) + 1.
\]

The elements of \(P'(S)\), for an \(S \subseteq S_{\kappa}\), can be looked upon as potential paths of a \(\kappa\)-conditional transition system with \(S\) as set of states. A (non-terminating) path of a transition system \((S, \to, \to^\square, s_0) \in \text{CTS}_{\kappa}\) is a finite alternating sequence \(\langle s_0, \ell_1, s_1, \ldots, \ell_n, s_n \rangle\) of states from \(S\) and labels from \(C_{\kappa} \times A\) such that \(s_0 = s^0\) and \(s_i \xrightarrow{\ell_{i+1}} s_{i+1}\) for all \(i < n\). The state \(s_n\) is called the state in which the path ends.

Let \(T = (S, \to, \to^\square, s^0) \in \text{CTS}_{\kappa}\). Then the set of paths of \(T\), written \(P(T)\), is the smallest subset of \(P'(S)\) such that:

\(\langle s^0 \rangle \in P(T)\),

if \(\pi \bowtie \langle s \rangle \in P(T)\) and \(s \xrightarrow{\ell} s'\), then \(\pi \bowtie \langle s, \ell, s' \rangle \in P(T)\).

In order to unfold a transition system, we need for each state \(s\) of the original transition system, for each different path that ends in state \(s\), a different state in the unfolded transition system. The obvious choice is to take the paths concerned as states.

\[^9\] We write \(\langle \rangle\) for the empty sequence, \(\langle e \rangle\) for the sequence having \(e\) as sole element and \(\sigma \bowtie \sigma'\) for the concatenation of sequences \(\sigma\) and \(\sigma'\); and we use \(\langle e_1, \ldots, e_n \rangle\) as a shorthand for \(\langle e_1 \rangle \bowtie \ldots \bowtie \langle e_n \rangle\).
Let \( T = (S, \rightarrow, \rightarrow^\vee, s^0) \in \text{CTS}_\kappa^r \). Then the unfolding of \( T \), written \( \Upsilon(T) \), is defined as follows:

\[ \Upsilon(T) = (S', \rightarrow', \rightarrow^\vee', s'^0) \]

where

\[ S' = P(T), \]

and for every \( \ell \in C_{\kappa}^- \times A \):

\[
\ell' = \{(\pi \cap (s), \pi \cap (s, \ell, s')) | \pi \cap (s) \in P(T), s \xrightarrow{\ell} s'\}, \]

\[
\ell, \sqrt{'} = \{\pi \cap (s) | \pi \cap (s) \in P(T), s \xrightarrow{\ell} \sqrt{'}\}, \]

\[ s'^0 = (s^0). \]

The functions \( \text{upd}_1 \) and \( \text{upd}_2 \) defined next will be used in the definition of parallel composition on \( \text{CTS}_\kappa^r \) to adapt the retrospection in steps originating from the first operand and the second operand, respectively.

Let \( S_1, S_2 \subseteq S_r \). Then the functions \( \text{upd}_i : C_{\kappa}^- \times P(S_1 \times S_2) \rightarrow C_{\kappa}^- \), for \( i = 1, 2 \), are defined by

\[
\text{upd}_i(\alpha, ((s_1, s_2))) = \alpha, \]

\[
\text{upd}_i(\alpha, ((s_1, s_2), \ell, (s'_1, s'_2))) = \text{upd}_i(\alpha, ((s'_1, s'_2))) \cap \pi' \]

if \( s_i \neq s'_i \),

\[
\text{upd}_i(\alpha, ((s_1, s_2), \ell, (s'_1, s'_2))) \cap \pi' = \text{upd}_i(\Pi_{\rightarrow^\vee, \#_i(((s'_1, s'_2))) \cap \pi'}(\alpha), ((s'_1, s'_2)) \cap \pi') \]

if \( s_i = s'_i \).

where

\[
\#_i(((s_1, s_2))) = 0, \]

\[
\#_i(((s_1, s_2), \ell, (s'_1, s'_2))) \cap \pi') = \#_i(((s'_1, s'_2))) \cap \pi') + 1 \]

if \( s_i \neq s'_i \),

\[
\#_i(((s_1, s_2), \ell, (s'_1, s'_2))) \cap \pi') = \#_i(((s'_1, s'_2))) \cap \pi') \]

if \( s_i = s'_i \).

Henceforth, we write \( \text{upd} (\alpha_1, \alpha_2, \pi) \) for \( \text{upd}_1 (\alpha_1, \pi) \cap \text{upd}_2 (\alpha_2, \pi) \).

We proceed with expanding the full retrospective splitting bisimulation models of \( \text{BPA}^r_\kappa \) to \( \text{ACP}^r_\kappa \).

We associate with the additional operator \( \parallel \) an operation \( \hat{\parallel} \) on \( \text{CTS}_\kappa^r \) as follows.

- Let \( T_1, T_2 \in \text{CTS}_\kappa^r \). Suppose that \( \Upsilon(T_i) = (S_i, \rightarrow_i, \rightarrow^\vee_i, s^0_i) \) for \( i = 1, 2 \), and \( \Upsilon(T_i) \parallel \Upsilon(T_2) = (S, \rightarrow, \rightarrow^\vee, s^0) \). Then

\[
T_1 \parallel T_2 = (S, \rightarrow', \rightarrow^\vee', s'^0), \]

where for every \( (\alpha, a) \in C_{\kappa}^- \times A \):

\[ 41 \]
cannot be determined uniquely.

unfolded before the actual composition takes place, in general, those conditions

are performed synchronously, the condition under which the action of

composition takes place. In a step where an action of

The basic idea behind this definition is twofold:

Remark 16.1. The operation \(\parallel\) on \(\mathcal{CT}_{\infty}\) is defined above in a step-by-step way. The basic idea behind this definition is twofold:

- \(T_1 \parallel T_2\) can be obtained by first composing \(T_1\) and \(T_2\) to \(T_1 \parallel T_2\) and then adapting the retrospections in steps of \(T_1 \parallel T_2\);
- unfolding of \(T_1 \parallel T_2\) is needed before the actual adaptations can take place because the adaptation of the retrospection in a step may be different for the different paths that end in the state from which the step starts.

Somewhat surprisingly, in addition, \(T_1\) and \(T_2\) must be unfolded before the actual composition takes place. In a step where an action of \(T_1\) and an action of \(T_2\) are performed synchronously, the condition under which the action of \(T_1\) can be performed and the condition under which the action of \(T_2\) can be performed are needed to adapt the retrospection in that step correctly. If \(T_1\) and \(T_2\) are not unfolded before the actual composition takes place, in general, those conditions cannot be determined uniquely.
The operations on $\text{CTS}_k^+$ to be associated with the additional operators $\parallel$ and $|$ are defined analogously. The operations on $\text{CTS}_k^\kappa$ to be associated with the additional operators $\partial_H$ are defined exactly as the operations on $\text{CTS}_k$ associated with them. We associate with the additional operators $\Pi_{\geq n}^+$ operations $\Pi_{\geq n}^+$ on $\text{CTS}_k^\kappa$ as follows.

- Let $T \in \text{CTS}_k^\kappa$. Suppose that $\Upsilon(T) = (T, \rightarrow, \rightarrow, s_0)$. Then

$$\Pi_{\geq n}^+(T) = (S, \rightarrow', \rightarrow', s_0),$$

where for every $(\alpha, a) \in C_r^\kappa \times A$:

$$\frac{(\alpha, a)}{0} = \{(\pi \circ (s), \pi' \circ (s')) \mid \exists \alpha' \in C_r^\kappa : \Pi_{\geq n}^+_{\#(\pi) + n}(\alpha') = \alpha\},$$

$$\frac{(\alpha, a)}{1} = \{(\pi \circ (s), \pi' \circ (s')) \mid \Pi_{\geq n}^+_{\#(\pi) + n}(\alpha') = \alpha\}.$$

The operation on $\text{CTS}_k^\kappa$ to be associated with the additional operator $\Pi^+$ is the same as the operation on $\text{CTS}_k^\kappa$ associated with $\Pi_{\geq 0}^+$.

We can also show that retrospective splitting bisimilarity is a congruence with respect to parallel composition, left merge, communication merge, encapsulation, retrospection shift and restricted retrospection shift.

**Proposition 16.1 (Congruence).** Let $\kappa$ be an infinite cardinal. Then for all $T_1, T_2, T'_1, T'_2 \in \text{CTS}_k^\kappa$, $T_1 \equiv T'_1$ and $T_2 \equiv T'_2$ imply $T_1 \parallel T'_1 \equiv T_2 \parallel T'_2$, $T_1 \parallel T_2 \equiv T'_1 \parallel T'_2$, $T_1 \parallel T'_1$, $T_2 \equiv T'_2$, $\partial_H(T_1) \equiv \partial_H(T'_1)$, $\Pi^+(T_1) \equiv \Pi^+(T'_1)$ and $\Pi_{\geq n}^+(T_1) \equiv \Pi_{\geq n}^+(T'_1)$.

**Proof.** It is easy to see that, for all $T, T' \in \text{CTS}_k^\kappa$, $T \equiv T'$ implies $\Upsilon(T) \equiv \Upsilon(T')$. Hence, $\Upsilon(T_1) \equiv \Upsilon(T'_1)$ and $\Upsilon(T_2) \equiv \Upsilon(T'_2)$. Let $R_1$ and $R_2$ be retrospective splitting bisimilarities witnessing $\Upsilon(T_1) \equiv \Upsilon(T'_1)$ and $\Upsilon(T_2) \equiv \Upsilon(T'_2)$, respectively; and let $R_1$ be a layered retrospective splitting bisimulation witnessing $T_1 \equiv T'_1$. Then we construct relations $R_1^0$, $R_1^1$, $R_1^2$, $R_2^0$, $R_2^1$ and $R_2^2$, respectively, as follows:

- Let $S$ and $S'$ be the sets of states of $T_1 \parallel T_2$ and $T'_1 \parallel T'_2$, respectively, and let $s_0^0$ and $s_0'$ be the initial states of $T_1 \parallel T_2$ and $T'_1 \parallel T'_2$, respectively. Then $R_1^0$ is the smallest subset of $S \times S' \times S'$ such that:

- $R_1^0(s_0^0, \pi, s_0')$;

- if $R_1^0(\pi \circ ((\pi_1, \pi_2)), \alpha, \pi' \circ ((\pi'_1, \pi'_2)))$,

$$R_1^0(\pi'_1 \circ (s_1), \alpha_1, \pi''_1 \circ (s'_1)), R_2^1(\pi''_2 \circ (s_2), \alpha_2, \pi''_2 \circ (s'_2)), \pi \circ ((\pi_1, \pi_2), \ell, (\pi''_1 \circ (s_1), \pi''_2 \circ (s_2))) \in S,$$

$$\pi' \circ ((\pi'_1, \pi'_2), \ell, (\pi''_1 \circ (s'_1), \pi''_2 \circ (s'_1))) \in S',$$

and
\[ (s_1 \neq s_1' \land s_2 = s_2' \land \text{upd}_1(\alpha_1, \pi \triangleright (\pi_1, \pi_2)) = \alpha') \lor \\
(s_1 = s_1' \land s_2 \neq s_2' \land \text{upd}_2(\alpha_2, \pi \triangleright (\pi_1, \pi_2)) = \alpha') \lor \\
(s_1 \neq s_1' \land s_2 \neq s_2' \land \text{upd}(\alpha_1, \alpha_2, \pi \triangleright (\pi_1, \pi_2)) = \alpha')) \]

then \( R_{\Pi^r}((\pi_1, \pi_2), \ell, (\pi_1'' \triangleright (s_1), \pi_2'' \triangleright (s_2))), \alpha', \\
\pi' \triangleright ((\pi_1', \pi_2'), \ell, (\pi_1''' \triangleright (s_1'), \pi_2''' \triangleright (s_2'))). \]

- \( R_{\Pi^{r+}} \) is constructed analogous to \( R_{\Pi^r} \).
- \( R_{\Pi^\R} \) is constructed analogous to \( R_{\Pi^r} \).
- \( R_{\Pi^{\R+}, \Pi^{\R_0}} = R_1 \cap (S \times S') \), where \( S \) and \( S' \) are the sets of states of \( \tilde{\delta}_{H'}(T_1) \) and \( \tilde{\delta}_{H'}(T_1') \), respectively.
- \( R_{\Pi^{\R+}, \Pi^{\R_0}, \Pi^{\R_1}} = ((\pi \triangleright (s), \Pi^r_{>n}(\alpha), \pi' \triangleright (s')) \in S \times C_{\kappa}^r \times S' \mid R_1(s, \alpha, s')) \), where \( S \) and \( S' \) are the sets of states of \( \Pi^{\R+}(T_1) \) and \( \Pi^{\R+}(T_1') \), respectively.
- \( R_{\Pi^{\R+}, \Pi^{\R_0}, \Pi^{\R_1}, \Pi^{\R_2}} = ((\pi \triangleright (s), \Pi^r_{>n}(\alpha), \pi' \triangleright (s')) \in S \times C_{\kappa}^r \times S' \mid R_1(s, \alpha, s')) \), where \( S \) and \( S' \) are the sets of states of \( \Pi^{\R+}(T_1) \) and \( \Pi^{\R+}(T_1') \), respectively.

Given the definitions of parallel composition, left merge, communication merge, encapsulation, retrospection shift and restricted retrospection shift, it is straightforward to see that \( R_{\Pi^r} \), \( R_{\Pi^{r+}} \), \( R_{\Pi^{\R}}, R_{\Pi^{\R+}}, \) and \( R_{\Pi^{\R}}, \Pi^{\R_1} \) are retrospective splitting bisimulations witnessing \( T_1 \bar{\equiv} T_2 \), \( T_1 \Pi^{\R+} T_2 \), \( T_1 \Pi^{\R} T_2 \), and \( T_1 \Pi^{\R+} T_2 \), \( R_{\Pi^{\R+}} \). \( T_1 \Pi^{\R+} T_2 \), \( \Pi^{\R+}(T_1) \Pi^{\R+}(T_2) \), and \( \Pi^{\R+}(T_1) \Pi^{\R+}(T_2) \), respectively. 

Note that, in the proof of Proposition 16.1, showing that \( R_{\Pi^{\R+}} \) and \( R_{\Pi^{\R+}, \Pi^{\R_1}} \) are witnesses needs the assumption that \( R_1 \) is layered.

The full retrospective splitting bisimulation models \( \mathfrak{P}_{\kappa}^{\Pi^r} \) of ACP\(^{\kappa} \), one for each infinite cardinal \( \kappa \), are the expansions of the full retrospective splitting bisimulation models \( \mathfrak{P}_{\kappa}^{\Pi^r} \) of BPA\(^{\kappa} \) with an \( n \)-ary operation \( \tilde{\delta} \) on the domain of \( \mathfrak{P}_{\kappa}^{\Pi^r} (\mathfrak{C}^{\Pi^r}_{\kappa}/\tilde{\equiv}) \) for each additional \( n \)-ary operator \( f \) of ACP\(^{\kappa} \). Those additional operations are defined as follows:

\[
\begin{align*}
[T_1]_{\tilde{\equiv}} \Pi^{\R+} [T_2]_{\tilde{\equiv}} &= [T_1 \Pi^{\R+} T_2]_{\tilde{\equiv}}, \\
[T_1]_{\tilde{\equiv}} \Pi^{\R} [T_2]_{\tilde{\equiv}} &= [T_1 \Pi^{\R} T_2]_{\tilde{\equiv}}, \\
[T_1]_{\tilde{\equiv}} \Pi^{\R+} [T_2]_{\tilde{\equiv}} &= [T_1 \Pi^{\R+} T_2]_{\tilde{\equiv}}.
\end{align*}
\]

Parallel composition, left merge, communication merge, encapsulation, retrospection shift and restricted retrospection shift on \( \mathfrak{C}^{\Pi^r}_{\kappa}/\tilde{\equiv} \) are well-defined because \( \tilde{\equiv} \) is a congruence with respect to the corresponding operations on \( \mathfrak{C}^{\Pi^r}_{\kappa} \).

The structures \( \mathfrak{P}_{\kappa}^{\Pi^r} \) are models of ACP\(^{\kappa} \).

**Theorem 16.1 (Soundness of ACP\(^{\kappa} \)).** For each infinite cardinal \( \kappa \), we have \( \mathfrak{P}_{\kappa}^{\Pi^r} \models \text{ACP}^{\kappa} \).

*Proof.* Because \( \tilde{\equiv} \) is a congruence, it is sufficient to show that all additional axioms are sound. The soundness of all additional axioms follows straightforwardly from the definitions of the ingredients of \( \mathfrak{P}_{\kappa}^{\Pi^r} \). \( \square \)
Above, the full retrospective splitting bisimulation models $\Psi^\kappa_\tau$ of BPA$^\tau$ have been expanded to obtain the full splitting bisimulation models $\Psi^\kappa_{\tau'}$ of ACP$^\tau$. We will loosely write $\Psi^\kappa_\tau$ for $\Psi^\kappa_{\tau'}$.

In the full retrospective splitting bisimulation models of ACP$^\tau$, guarded recursive specifications over ACP$^\tau$ have unique solutions.

**Theorem 16.2 (Unique solutions in $\Psi^\kappa_\tau$).** For each infinite cardinal $\kappa$, guarded recursive specifications over ACP$^\tau$ have unique solutions in $\Psi^\kappa_\tau$.

**Proof.** The proof follows the same line as the proof of Theorem 8.1. Here, it is crucial that it is straightforward to define a normal form of elements of CTS$^\kappa_\tau$ such that: (a) each element of CTS$^\kappa_\tau$ is retrospective splitting bisimilar to its normal form and (b) two elements of CTS$^\kappa_\tau$ are retrospective splitting bisimilar iff their normal forms are splitting bisimilar. $\Box$

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

### 17 Evaluation of Retrospective Conditions

In this section, we add condition evaluation and generalized condition evaluation operators to ACP$^\tau$. As in the case of ACP$^c$, these operators require to fix an infinite cardinal $\lambda$. By doing so, full retrospective splitting bisimulation models with domain CTS$^\kappa_\lambda/\Xi^\tau$ for $\kappa > \lambda$ are excluded.

In the case of ACP$^\tau$, there are $\lambda$-complete condition evaluation operators $CE_h: P \rightarrow P$ and $CE_h: C \rightarrow C$ for each $\lambda$-complete endomorphism $h$ of $C^\tau_\lambda$.

Henceforth, we write $H_\lambda^\tau$ for the set of all $\lambda$-complete endomorphisms of $C^\tau_\lambda$.

The additional axioms for $CE_h$, where $h \in H_\lambda^\tau$, are axioms $CE1$–$CE11$ given in Table 8 (see Section 9), with the understanding that all endomorphisms involved are now taken from $C^\tau_\lambda$. The structural operational semantics of ACP$^\tau$ extended with condition evaluation is described by the transition rules for ACP$^\tau$ and the transition rules given in Table 9 (see Section 9), with the understanding that all endomorphisms involved are now taken from $C^\tau_\lambda$. As in the case of ACP$^c$, the full retrospective splitting bisimulation models of ACP$^\tau$ with condition evaluation are simply the expansions of the full retrospective splitting bisimulation models $\Psi^\kappa_\tau$ of ACP$^\tau$, for infinite cardinals $\kappa \leq \lambda$, obtained by associating with each operator $CE_h$ the corresponding re-labeling operation on conditional transition systems with retrospection. As mentioned before, full retrospective splitting bisimulation models with domain CTS$^\kappa_\lambda/\Xi^\tau$ for $\kappa > \lambda$ are excluded.

In summary, the condition evaluation operators $CE_h$ are added to ACP$^\tau$ in the same way as they are added to ACP$^c$. We proceed with adding the generalized condition evaluation operators GCE$\_h$ to ACP$^\tau$. These operators cannot be added to ACP$^\tau$ in the same way as they are added to ACP$^c$. Recall that generalized condition evaluation allows the results of condition evaluation to change.
by performing an action. In the presence of retrospection, different parts of a condition may have to be evaluated differently because of such changes.

In the case of ACP\textsuperscript{ext}, it is assumed that a fixed but arbitrary function $\text{eff}: A \times \mathcal{H}_\lambda^\ell \to \mathcal{H}_\lambda^\ell$ has been given. The function $\text{eff}$ is extended to $A_\lambda$ such that $\text{eff}(\delta, h) = h$ for all $h \in \mathcal{H}_\lambda^\ell$. Moreover, there is a generalized $\lambda$-complete condition evaluation operator $GCE_h: P \to P$ for each $h \in \mathcal{H}_\lambda^\ell$. We also need the following auxiliary operators:

- for each $h \in \mathcal{H}_\lambda^\ell$, $n \in \mathbb{N}$, the unary *retrospection update* operator $\Pi^h_n: P \to P$.
- for each $h \in \mathcal{H}_\lambda^\ell$, $n \in \mathbb{N}$, the unary *retrospection update* operator $\Pi^h_n: C \to C$.

In generalized condition evaluation of a process according to some endomorphism, after an action of the process is performed, the subsequent retrospective conditions that refer back to the beginning of the process should be evaluated according to that endomorphism as well. This is accomplished by the retrospection update operators.

The additional axioms for $GCE_h$, where $h \in \mathcal{H}_\lambda^\ell$, are the axioms given in Table 23. These additional axioms differ from the additional axioms in the absence of retrospection (Table 10) in that axioms GCE2 and GCE4 have been replaced by axioms GCE2R and GCE4R, and axioms CE6–CE11 by axioms RU1–RU13. Axiom GCE2R, together with axioms RU1–RU13, shows that in generalized condition evaluation of a process according to some endomorphism, after an action of the process is performed, the subsequent retrospective conditions that refer back to the beginning of the process are evaluated according to that endomorphism as well. Axiom GCE4R, together with axioms RU1–RU13, shows that, before an action of the process is performed, retrospective conditions that refer back are not at all evaluated.

*Example 17.1.* We return to Example 13.1, which is concerned with a pedestrian who uses a crossing with traffic lights to cross a road with busy traffic.
Table 24. Transition rules for generalized retrospective condition evaluation

<table>
<thead>
<tr>
<th>Transition Rule</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \cdot [\phi] \cdot \sqrt{\text{GCE}_h(x)}$</td>
<td>$\Pi^h_0(\phi) \neq \bot$</td>
<td>$x'$</td>
</tr>
<tr>
<td>$x \cdot [\phi] \cdot \sqrt{\text{GCE}_h(x)}$</td>
<td>$\Pi^h_0(\phi) \neq \bot$</td>
<td>$x'$</td>
</tr>
<tr>
<td>$x \cdot [\phi] \cdot \sqrt{\text{GCE}_h(x)}$</td>
<td>$\Pi^h_0(\phi) \neq \bot$</td>
<td>$x'$</td>
</tr>
<tr>
<td>$x \cdot [\phi] \cdot \sqrt{\text{GCE}_h(x)}$</td>
<td>$\Pi^h_0(\phi) \neq \bot$</td>
<td>$x'$</td>
</tr>
</tbody>
</table>

safely, but acts unthinkingly. Recall that the description of the behaviour of the unthinkingly acting pedestrian given in Example 13.1 is as follows:

$$PED' = \text{arrive} \cdot \text{make-req} \cdot (\neg \text{green} : \rightarrow \text{cross} + \neg \text{red} : \rightarrow (\text{green} : \rightarrow \text{cross}))$$

Like in Example 9.2, let $h_g$ be such that $h_g(\text{green}) = \top$ and $h_g(\text{red}) = \bot$, let $h_r$ be such that $h_r(\text{green}) = \bot$ and $h_r(\text{red}) = \top$, and let $\text{eff}$ be such that $\text{eff}(\text{make-req}, h_r) = h_g$ and $\text{eff}(a, h) = h$ otherwise. Then we can derive the following:

$$\text{GCE}_{h_g}(PED') = \text{arrive} \cdot \text{make-req} \cdot \text{cross}$$
$$\text{GCE}_{h_r}(PED') = \text{arrive} \cdot \text{make-req} \cdot \text{cross}$$

As to be expected, the unthinkingly acting pedestrian will act the same regardless the color of the traffic light for pedestrians on arrival.

The structural operational semantics of ACP<sub>cr</sub> extended with generalized condition evaluation is described by the transition rules for ACP<sub>cr</sub> and the transition rules given in Table 24.

The full retrospective splitting bisimulation models of ACP<sub>cr</sub> with generalized condition evaluation are not simply the expansions of the full retrospective splitting bisimulation models $\mathcal{P}_{\kappa}$ of ACP<sub>cr</sub>, for infinite cardinals $\kappa \leq \lambda$, obtained by associating with each operator $\text{GCE}_h$ the corresponding re-labeling operation on conditional transition systems with retrospection. As suggested by the structural operational semantics of ACP<sub>cr</sub> extended with generalized condition evaluation, these re-labeling operations have to be adapted in a way similar to the way in which parallel composition had to be adapted to the case with retrospection in Section 16. As mentioned before, full retrospective splitting bisimulation models with domain $\mathcal{CTS}_{r, \kappa} / \mathcal{S}_r$ for $\kappa > \lambda$ are excluded.

We can add both the $\lambda$-complete condition evaluation operators and the generalized $\lambda$-complete condition evaluation operators to ACP<sup>cr</sup>. Proposition 9.2, stating that the latter operators supersede the former operators in the setting of ACP<sup>e</sup>, goes through in the setting of ACP<sup>cr</sup>.

Adding state operators to ACP<sup>cr</sup> can be done on the same lines as adding generalized evaluation operators to ACP<sup>cr</sup>, but is more complicated. Roughly speaking, signal emission can be added to ACP<sup>cr</sup> in the same way as it is added to ACP<sup>e</sup> provided that signals are taken from $C$. No adaptations like for generalized condition evaluation are needed because signal emission corresponds to condition
Table 25. Additional axioms for last action conditions ($a \in A$)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \cdot x = a \cdot (J_a :: x)$</td>
<td>(J)</td>
</tr>
</tbody>
</table>

Table 26. Axioms adapted to last action conditions ($a, b \in A, c \in A$)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \cdot x \mid b = (a \mid b) \cdot \Pi^0_a(x)$</td>
<td>CM5J  (\Pi^a_0(\bot) = \bot) LAU5</td>
</tr>
<tr>
<td>$a \mid b \cdot x = (a \mid b) \cdot \Pi^0_b(x)$</td>
<td>CM6J  (\Pi^a_0(\top) = \top) LAU6</td>
</tr>
<tr>
<td>$a \cdot x \mid b \cdot y = (a \mid b) \cdot \Pi^0_a(x) \parallel \Pi^0_b(y)$</td>
<td>CM7J  $\Pi^a_0(J_a) = \bot$ if $a \neq c$ LAU7</td>
</tr>
<tr>
<td>$\Pi^a_n(J_a) = \top$ if $a = c$ LAU8</td>
<td></td>
</tr>
<tr>
<td>$\Pi^+_{n+1}(J_a) = \sim J_a$</td>
<td>RS7Ja  $\Pi^a_{n+1}(J_a) = J_a$ LAU9</td>
</tr>
<tr>
<td>$\Pi^+_{n+1}(J_a) = J_a$</td>
<td>RS7Jb  $\Pi^a_n(-\phi) = -\Pi^a_n(\phi)$ LAU10</td>
</tr>
<tr>
<td>$\Pi^a_n(\phi \sqcup \psi) = \Pi^a_n(\phi) \sqcup \Pi^a_n(\psi)$</td>
<td>LAU11</td>
</tr>
<tr>
<td>$\Pi^a_n(\phi \cap \psi) = \Pi^a_n(\phi) \cap \Pi^a_n(\psi)$</td>
<td>LAU12</td>
</tr>
<tr>
<td>$\Pi^a_n(b) = b$</td>
<td>LAU1</td>
</tr>
<tr>
<td>$\Pi^a_n(\phi \cap \psi) = \Pi^a_n(\phi) \cap \Pi^a_n(\psi)$</td>
<td>LAU13</td>
</tr>
<tr>
<td>$\Pi^a_n(-\phi) = -\Pi^a_n(\phi)$</td>
<td>LAU14</td>
</tr>
<tr>
<td>$\Pi^a_n(\phi :: x) = \Pi^a_n(\phi) :: \Pi^a_n(x)$</td>
<td>LAU4</td>
</tr>
</tbody>
</table>

evaluation that does not persist over performing an action. This property also points at one of the differences between the signal-emission approach to condition evaluation and the other approaches treated in this paper: retrospection has to be resolved in the signal-emission approach before condition evaluation can take place. The case where signals are taken from \(C\) is expected to be too complicated to handle.

18 An Application of ACP<sub>cr</sub>

The ultimate applications of a process algebra that includes conditional expressions of some form are the ones that remain entirely within the domain of process algebra. Such applications are by their nature extensions as well. We outline one interesting application of this kind in the setting of ACP<sub>cr</sub>.

We take the set \(\{J_a \mid a \in A\}\) of last action conditions as the set of atomic conditions \(C_{at}\). The intuition is that \(J_a\) indicates that action \(a\) is performed just now. The retrospection operator now allows for using conditions which express that a certain number of steps ago a certain action must have been performed.

Because we remain entirely within the domain of process algebra some additional axioms are needed. They are given in Table 25. Moreover, axioms CM5–CM7 (Table 4) and RS7 (Table 20) must be replaced by axioms CM5J–CM7J, RS7Ja and RS7Jb from Table 26. Axioms CM5–CM7 must be replaced by axioms CM5J–CM7J because, after performing \(a \mid b\), it makes no sense to refer back to the actions performed just now by the processes originally following \(a\) and \(b\) in the process following \(a \mid b\). Retrospective conditions in the process originally following \(a\) that indicate that \(a\) is performed just now should be evaluated to
and the ones that indicate that another action is performed just now should be evaluated to ⊥. Retrospective conditions in the process originally following b should be evaluated analogously. This is accomplished by the auxiliary operators \( \Pi_n^a : P \rightarrow P \) and \( \Pi_n^b : C \rightarrow C \) (for each \( a \in A \) and \( n \in \mathbb{N} \)) of which the defining axioms are axioms LAU1–LAU14 from Table 26. Axiom RS7 must be replaced by axioms RS7Ja and RS7Jb because of the retrospective nature of last action conditions. We mean by this that \( J_a \) can be viewed as a condition of the form \( \sim \eta \), where \( \eta \) indicates that action \( a \) is performed next. We have not introduced corresponding atomic conditions because their use without restrictions would be problematic in alternative composition.

From the axioms of \( \text{BPA}_\kappa \) and the additional axiom J, we can derive the equation \( a \cdot x + b \cdot y = (a + b) \cdot (J_a :\rightarrow x + J_b :\rightarrow y) \). This equation can be used to reduce the number of subprocesses of a process. For example, the equation \( a \cdot a' + b \cdot b' = (a+b) \cdot (J_a :\rightarrow a' + J_b :\rightarrow b') \) shows a reduction from 3 subprocesses to 2 subprocesses and the equation \( a \cdot (a_1 \cdot a'_1 + a_2 \cdot a'_2) + b \cdot (b_1 \cdot b'_1 + b_2 \cdot b'_2) = (a+b) \cdot (J_a :\rightarrow (a_1 + a_2) \cdot (J_{a_1} :\rightarrow a'_1 + J_{a_2} :\rightarrow a'_2) + J_b :\rightarrow (b_1 + b_2) \cdot (J_{b_1} :\rightarrow b'_1 + J_{b_2} :\rightarrow b'_2)) \) shows a reduction from 7 subprocesses to 4 subprocesses.

In order to obtain the full retrospective splitting bisimulation models of the extension of \( \text{ACP}^{\kappa} \) with last action conditions, retrospective splitting bisimilarity has to be adapted: in the definition of retrospective splitting bisimulation (see Section 14), the two occurrences of \( B(s'_1, \sim a', s'_2) \) must be replaced by \( B(s'_1, a' \cap J_a, s'_2) \).

The operators \( \Pi_n^a \) are reminiscent of the operators \( \Pi_n^b \) from Section 17. In fact, if we would exclude full retrospective splitting bisimulation models with domain \( \text{CTS}_\kappa / \Xi^\kappa \) for \( \kappa \) greater than some infinite cardinal \( \lambda \), \( \Pi_n^a \) could have been replaced by \( \Pi_n^{\kappa^{\alpha}} \), where \( h_a \in \mathcal{H}_\lambda^\kappa \) for \( a \in A \) is defined by \( h_a(J_a) = \top \) and \( h_a(J_b) = \bot \) if \( a \neq b \) and \( h_b \in \mathcal{H}_\lambda^\kappa \) is defined by \( h_b(J_a) = \bot \).

19 Concluding Remarks

In this paper, we build on earlier work on ACP. Conditional expressions of the form \( \zeta :\rightarrow p \) were added to ACP for the first time in [6]. In [2], it was proposed to take the domain of a free Boolean algebra over a given set of generators as the set of conditions. Splitting bisimilarity is based on the variant of bisimilarity that was defined for the first time in [2]. The formulation given here is closer to the one given in [13]. State operators and signal emission were added to ACP for the first time in [1] and [3], respectively. The condition evaluation operators, the generalized evaluation operators and the retrospection operator are new. The variants of splitting bisimilarity, i.e. signal-observing splitting bisimilarity and retrospective splitting bisimilarity, are new as well.

Full bisimulation models were presented in [10] for a first-order extension of ACP. Those models are basically the graph models of ACP, which are most extensively described in [5]. The full splitting bisimulation models of ACP\(^\kappa\) presented in this paper, as well as the full signal-observing splitting bisimilation models of ACP\(^{\kappa^\alpha}\) and the full retrospective splitting bisimulation models of...
ACP_{cr}, are adaptations of the full bisimulation models from [10]. The adaptations, in particular for the models of ACP_{cs} and ACP_{cr}, are substantial.

The above-mentioned variants of full bisimulation models take into account infinitely branching processes, even in the case where the set of atomic conditions (the set of generators) is infinite. We are not aware of previous work presenting models of such generality for a process algebra with conditional expressions. We are also not aware of previous work studying condition evaluation or retrospective conditions in a process algebra with conditional expressions.

In some extensions of ACP with conditional expressions, the conditions are propositions of a three-, four- or five-valued propositional logic, see e.g. [12, 24]. It is not clear whether the work presented in this paper can be adapted to those cases, because they bring us outside the domain of Boolean algebras.

In this paper, we give a survey of algebraic theories about processes that include conditional expressions and the main models of those theories. Although our aim is to provide complete axiomatizations, we do not present completeness theorems. We conjecture that the axioms of ACP_{c}, ACP_{cs} and ACP_{cr} form complete axiomatizations of the full splitting bisimulation models of ACP_{c}, the full signal-observing splitting bisimulation models of ACP_{cs} and the full retrospective splitting bisimulation models of ACP_{cr}, respectively, with respect to equations between closed terms; and leave the proofs for future work.

Other options for future work include: development of an extension of ACP_{cr} with state operators, development of an extension of ACP_{cr} with signal emission, development of first-order extensions of ACP_{c}, ACP_{cs} and ACP_{cr} in the style of [10], and investigations into ways to deal with the history pointers from [4] in the setting of ACP_{cr}.

References