Strong Splitting Bisimulation Equivalence

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
janb@science.uva.nl

[2] Department of Philosophy, Utrecht University,
P.O. Box 80126, 3508 TC Utrecht, the Netherlands
janb@phil.uu.nl

[3] Computing Science Department, Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
keesm@win.tue.nl

Abstract. We present ACP$^{\mathrm{c}}$, a process algebra with conditional expressions in which the conditions are taken from a Boolean algebra, and extensions of this process algebra with mechanisms for condition evaluation. We confine ourselves to finitely branching processes. This restriction makes it possible to present ACP$^{\mathrm{c}}$ in a concise and intuitively clear way, and to bring the notion of splitting bisimulation equivalence and the issue of condition evaluation in process algebras with conditional expressions to the forefront.

1 Introduction

It is not unusual that process algebras include conditional expressions of some form. Several extensions of ACP [1, 2] include conditional expressions of the form $\zeta :\rightarrow p$ or $p \lhd \zeta \rhd q$ (see e.g. [3–6]). What are considered to be conditions differs from one extension to another. The set of conditions is usually one of the following: (i) a two-valued set, usually called $\mathbb{B}$; (ii) the set of all propositions with a given set of propositional variables and with finite conjunctions and disjunctions; (iii) the domain of a free Boolean algebra over a given set of generators. The third alternative generalizes the first two alternatives. In this paper, we present ACP$^{\mathrm{c}}$, an extension of ACP with conditional expressions of the form $\zeta :\rightarrow p$ in which the domain of a free Boolean algebra over a given set of generators is taken as the set of conditions. We give the axioms of ACP$^{\mathrm{c}}$, describe the structural operational semantics of ACP$^{\mathrm{c}}$, and introduce a variant of bisimulation equivalence, called splitting bisimulation equivalence, for which the axiomatization is sound. In the title, the qualification "strong" is used to indicate that splitting bisimulation equivalence does not provide for abstraction from internal actions. Outside the title, we leave out this qualification.

How conditions are evaluated is usually not considered. The state operators as introduced in [4] allow for a kind of condition evaluation. However, state operators were not especially devised for that purpose. In this paper, we extend

ACP^{c} with operators especially devised for condition evaluation and with the state operators from [4]; and show how those extensions are related. 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.

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 [7]. Adaptation to CSP [8], which is not based on bisimulation models, will be more difficult and in part perhaps even impossible. In some extensions of ACP with conditional expressions, the conditions are propositions of a three-, four- or five-valued propositional logic, see e.g. [9, 10]. Such conditions will bring us outside the domain of Boolean algebras.

In [11], we investigated conditional expressions in the setting of ACP more extensively. In that paper, we presented ACP^{c} for the first time. We also presented its main models, called full splitting bisimulation models. We extended ACP^{c} with the above-mentioned operators especially devised for condition evaluation, the state operators from [4] and the signal emission operator from [6], which like the state operators allows for a kind of condition evaluation. We also showed how those extensions are related. On purpose to incorporate the past in conditions, we also added a retrospection operator on conditions to ACP^{c}.

All this fitted in with our intention at the time: to arrive at a well-considered extension of ACP with conditional expressions in which retrospective conditions can be used. Retrospective conditions allow for looking back on conditions under which preceding actions have been performed. Their addition is considered to be a basic way to increase expressiveness. In the full splitting bisimulation models of ACP^{c}, infinitely branching processes are taken into account. Because the set of atomic conditions is not required to be finite, those models are rather complicated. Moreover, the adaptation of the full splitting bisimulation models to the retrospection operator on conditions is quite substantial. As a result, other interesting matters are pushed to the background in [11].

The current paper can be viewed as an extended abstract of some parts of [11]. Most importantly, the full splitting bisimulation models of ACP^{c} and the addition of the retrospection operator on conditions to ACP^{c} are not covered. Moreover, because we confine ourselves to finitely branching processes, the presentation of what is left over has been fairly simplified.

We do not give proofs of the theorems concerning congruence properties of splitting bisimulation equivalence, soundness of axiomatizations for splitting bisimulation equivalence, and uniqueness of solutions of guarded recursive specifications. Those theorems follow from the corresponding theorems in [11] because the structural operational semantics induces a model isomorphic to the full bisimulation model that covers only finitely branching processes.

2 BPA with Conditions

BPA_δ is a subtheory of ACP that does not support parallelism and communication (see e.g. [2]). 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 $\mathrm{BPA}_\delta^{\mathrm{c}}$. In the extension, just as in BPA_δ, it is assumed that a fixed but arbitrary finite set of *actions* A, with $\delta \notin \mathsf{A}$, has been given. Moreover it is assumed that a fixed but arbitrary set of *atomic conditions* C_{at} has been given.

In $\mathrm{BPA}_\delta^{\mathrm{c}}$, conditions are taken from the domain of the free Boolean algebra over C_{at}. We denote this algebra by $\mathcal{C}$. As usual, we identify Boolean algebras with their domain. Thus, we also write $\mathcal{C}$ for the domain of $\mathcal{C}$. It is well known that $\mathcal{C}$ is isomorphic to the Boolean algebra of equivalence classes with respect to logical equivalence of the set of all propositions with elements of C_{at} as propositional variables and with finite conjunctions and disjunctions (see e.g. [12]).

The algebraic theory $\mathrm{BPA}_\delta^{\mathrm{c}}$ has two sorts:

– the sort $\mathbf{P}$ of *processes*;
– the sort $\mathbf{C}$ of (*finite*) *conditions*.

The algebraic theory $\mathrm{BPA}_\delta^{\mathrm{c}}$ has the following constants and operators to build terms of sort $\mathbf{C}$:

– the *bottom* constant $\bot : \mathbf{C}$;
– the *top* constant $\top : \mathbf{C}$;
– for each $\eta \in \mathsf{C}_{\mathsf{at}}$, the *atomic condition* constant $\eta : \mathbf{C}$;
– the unary *complement* operator $- : \mathbf{C} \to \mathbf{C}$;
– the binary *join* operator $\sqcup : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$;
– the binary *meet* operator $\sqcap : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$.

The algebraic theory $\mathrm{BPA}_\delta^{\mathrm{c}}$ has the following constants and operators to build terms of sort $\mathbf{P}$:

– the *deadlock* constant $\delta : \mathbf{P}$;
– for each $a \in \mathsf{A}$, the *action* constant $a : \mathbf{P}$;
– the binary *alternative composition* operator $+ : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
– the binary *sequential composition* operator $\cdot : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
– the binary *guarded command* operator $:\rightarrow : \mathbf{C} \times \mathbf{P} \to \mathbf{P}$.

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 $\mathbf{C}$ bind stronger than the operators to build terms of sort $\mathbf{P}$. The operator $\cdot$ binds stronger than all other binary operators to build terms of sort $\mathbf{P}$ and the operator $+$ binds weaker than all other binary operators to build terms of sort $\mathbf{P}$.

The constants and operators of $\mathrm{BPA}_\delta^{\mathrm{c}}$ to build terms of sort $\mathbf{P}$ are the constants and operators of BPA_δ and additionally the guarded command operator. Let p and q be closed terms of sort $\mathbf{P}$ and ζ be a closed term of sort $\mathbf{C}$. Intuitively, the constants and operators to build terms of sort $\mathbf{P}$ can be explained as follows:

Table 1. Axioms of Boolean algebras

$\phi \sqcup \bot = \phi$	BA1	$\phi \sqcap \top = \phi$	BA5
$\phi \sqcup -\phi = \top$	BA2	$\phi \sqcap -\phi = \bot$	BA6
$\phi \sqcup \psi = \psi \sqcup \phi$	BA3	$\phi \sqcap \psi = \psi \sqcap \phi$	BA7
$\phi \sqcup (\psi \sqcap \chi) = (\phi \sqcup \psi) \sqcap (\phi \sqcup \chi)$	BA4	$\phi \sqcap (\psi \sqcup \chi) = (\phi \sqcap \psi) \sqcup (\phi \sqcap \chi)$	BA8

Table 2. Axioms of $\mathrm{BPA}_\delta^\mathrm{c}$

$x + y = y + x$	A1	$\top :\rightarrow x = x$	GC1
$(x + y) + z = x + (y + z)$	A2	$\bot :\rightarrow x = \delta$	GC2
$x + x = x$	A3	$\phi :\rightarrow \delta = \delta$	GC3
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4	$\phi :\rightarrow (x + y) = \phi :\rightarrow x + \phi :\rightarrow y$	GC4
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5	$\phi :\rightarrow x \cdot y = (\phi :\rightarrow x) \cdot y$	GC5
$x + \delta = x$	A6	$\phi :\rightarrow (\psi :\rightarrow x) = (\phi \sqcap \psi) :\rightarrow x$	GC6
$\delta \cdot x = \delta$	A7	$(\phi \sqcup \psi) :\rightarrow x = \phi :\rightarrow x + \psi :\rightarrow x$	GC7

– δ 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 ζ.

Some earlier extensions of ACP include conditional expressions of the form $p \lhd \zeta \rhd q$; see e.g. [4]. This notation with triangles originates from [13]. We treat conditional expressions of the form $p \lhd \zeta \rhd q$, where p and q are terms of sort **P** and ζ is a term of sort **C**, as abbreviations. That is, we write $p \lhd \zeta \rhd q$ for $\zeta :\rightarrow p + -\zeta :\rightarrow q$.

The axioms of $\mathrm{BPA}_\delta^\mathrm{c}$ 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 BPA_δ. So $\mathrm{BPA}_\delta^\mathrm{c}$ imports the (equational) axioms of both BA and BPA_δ. The axioms of BA given in Table 1 have been taken from [14]. Several alternatives for this axiomatization can be found in the literature. If we use basic laws of BA other than axioms BA1–BA8, such as $\phi \sqcap \phi = \phi$ and $-(\phi \sqcap \psi) = -\phi \sqcup -\psi$, 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 [4], but with the axiom $x \cdot z \lhd \phi \rhd y \cdot z = (x \lhd \phi \rhd y) \cdot z$ (CO5) replaced by $\phi :\rightarrow x \cdot y = (\phi :\rightarrow x) \cdot y$ (GC5).

Example 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

Table 3. Transition rules for $\mathrm{BPA}^{\mathrm{c}}_{\delta}$

$$\frac{}{a \xrightarrow{[\top]\,a} \surd}$$

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{x+y \xrightarrow{[\phi]\,a} \surd} \quad \frac{y \xrightarrow{[\phi]\,a} \surd}{x+y \xrightarrow{[\phi]\,a} \surd} \quad \frac{x \xrightarrow{[\phi]\,a} x'}{x+y \xrightarrow{[\phi]\,a} x'} \quad \frac{y \xrightarrow{[\phi]\,a} y'}{x+y \xrightarrow{[\phi]\,a} y'}$$

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{x \cdot y \xrightarrow{[\phi]\,a} y} \quad \frac{x \xrightarrow{[\phi]\,a} x'}{x \cdot y \xrightarrow{[\phi]\,a} x' \cdot y}$$

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{\psi :\rightarrow x \xrightarrow{[\phi\sqcap\psi]\,a} \surd}\ \phi \sqcap \psi \neq \bot \quad \frac{x \xrightarrow{[\phi]\,a} x'}{\psi :\rightarrow x \xrightarrow{[\phi\sqcap\psi]\,a} x'}\ \phi \sqcap \psi \neq \bot$$

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 $\mathrm{BPA}^{\mathrm{c}}_{\delta}$ as follows:

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

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.

Henceforth, we write $\mathcal{T}_{\mathbf{P}}$ for the set of all closed terms of sort $\mathbf{P}$ and $\mathcal{T}_{\mathbf{C}}$ for the set of all closed terms of sort $\mathbf{C}$. The terms of sort $\mathbf{C}$ are interpreted in $\mathcal{C}$ as usual. Henceforth, we write $\mathcal{C}^-$ for $\mathcal{C} \setminus \{\bot\}$.

We proceed to the presentation of the structural operational semantics of $\mathrm{BPA}^{\mathrm{c}}_{\delta}$. The following transition relations on $\mathcal{T}_{\mathbf{P}}$ are used:

- for each $\ell \in \mathcal{C}^- \times \mathsf{A}$, a binary relation $\xrightarrow{\ell}$;
- for each $\ell \in \mathcal{C}^- \times \mathsf{A}$, a unary relation $\xrightarrow{\ell}\surd$.

We write $p \xrightarrow{[\alpha]\,a} q$ instead of $(p,q) \in \xrightarrow{(\alpha,a)}$ and $p \xrightarrow{[\alpha]\,a} \surd$ instead of $p \in \xrightarrow{(\alpha,a)}\surd$. The relations $\xrightarrow{\ell}\surd$ and $\xrightarrow{\ell}$ can be explained as follows:

- $p \xrightarrow{[\alpha]\,a} \surd$: p is capable of performing action a under condition α and then terminating successfully;
- $p \xrightarrow{[\alpha]\,a} q$: p is capable of performing action a under condition α and then proceeding as q.

The structural operational semantics of $\mathrm{BPA}^{\mathrm{c}}_{\delta}$ is described by the transition rules given in Table 3.

Bisimilarity has to be adapted to the setting with guarded actions. In the definition given below, we use a well-known notion from the field of Boolean algebras: the partial order relation $\sqsubseteq$ on $\mathcal{C}$ defined by $\alpha \sqsubseteq \beta$ iff $\alpha \sqcup \beta = \beta$. Moreover, we use the notation $\bigsqcup A$, where $A = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathcal{C}$, for $\alpha_1 \sqcup \ldots \sqcup \alpha_n$.

A *splitting bisimulation* B between closed terms $p, q \in \mathcal{T}_{\mathbf{P}}$ is a binary relation on $\mathcal{T}_{\mathbf{P}}$ such that $B(p, q)$ and for all p_1, q_1 such that $B(p_1, q_1)$:

- if $p_1 \xrightarrow{[\alpha]\,a} p_2$, then there exists a finite set $CT' \subseteq \mathcal{C}^- \times \mathcal{T}_{\mathbf{P}}$ such that $\alpha \sqsubseteq \bigsqcup \mathrm{dom}(CT')$ and for all $(\alpha', q_2) \in CT'$, $q_1 \xrightarrow{[\alpha']\,a} q_2$ and $B(p_2, q_2)$;
- if $q_1 \xrightarrow{[\alpha]\,a} q_2$, then there exists a finite set $CT' \subseteq \mathcal{C}^- \times \mathcal{T}_{\mathbf{P}}$ such that $\alpha \sqsubseteq \bigsqcup \mathrm{dom}(CT')$ and for all $(\alpha', p_2) \in CT'$, $p_1 \xrightarrow{[\alpha']\,a} p_2$ and $B(p_2, q_2)$;
- if $p_1 \xrightarrow{[\alpha]\,a} \surd$, then there exists a finite set $C' \subseteq \mathcal{C}^-$ such that $\alpha \sqsubseteq \bigsqcup C'$ and for all $\alpha' \in C'$, $q_1 \xrightarrow{[\alpha']\,a} \surd$;
- if $q_1 \xrightarrow{[\alpha]\,a} \surd$, then there exists a finite set $C' \subseteq \mathcal{C}^-$ such that $\alpha \sqsubseteq \bigsqcup C'$ and for all $\alpha' \in C'$, $p_1 \xrightarrow{[\alpha']\,a} \surd$.

Two closed term $p, q \in \mathcal{T}_{\mathbf{P}}$ are *splitting bisimulation equivalent* or *splitting bisimilar* for short, written $p \mathrel{\underline{\leftrightarrow}} q$, if there exists a splitting bisimulation B between p and q. Let B be a splitting bisimulation between p and q. Then we say that B is a splitting bisimulation *witnessing* $p \mathrel{\underline{\leftrightarrow}} q$.

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

Splitting bisimilarity is a congruence with respect to alternative composition, sequential composition and guarded command.

Proposition 1 (Congruence). *For all $p, p', q, q' \in \mathcal{T}_{\mathbf{P}}$ and $\alpha \in \mathcal{C}$, $p \mathrel{\underline{\leftrightarrow}} q$ and $p' \mathrel{\underline{\leftrightarrow}} q'$ implies $p + p' \mathrel{\underline{\leftrightarrow}} q + q'$, $p \cdot p' \mathrel{\underline{\leftrightarrow}} q \cdot q'$ and $\alpha :\rightarrow p \mathrel{\underline{\leftrightarrow}} \alpha :\rightarrow q$.*

The axioms of $\mathrm{BPA}_\delta^{\mathrm{c}}$ constitute a sound and complete axiomatization of splitting bisimilarity.

Theorem 1 (Soundness). *For all $p, q \in \mathcal{T}_{\mathbf{P}}$, $\mathrm{BPA}_\delta^{\mathrm{c}} \vdash p = q$ implies $p \mathrel{\underline{\leftrightarrow}} q$.*

Theorem 2 (Completeness). *For all $p, q \in \mathcal{T}_{\mathbf{P}}$, $p \mathrel{\underline{\leftrightarrow}} q$ implies $\mathrm{BPA}_\delta^{\mathrm{c}} \vdash p = q$.*

Proof. The proof follows the same line as the completeness proof for BPA_δ given in [16]. □

3 ACP with Conditions

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

Like in $\mathrm{BPA}_\delta^{\mathrm{c}}$, it is assumed that a fixed but arbitrary finite set of *actions* A, with $\delta \notin \mathsf{A}$, and a fixed but arbitrary set of *atomic conditions* $\mathsf{C_{at}}$ has been given. We write A_δ for $\mathsf{A} \cup \{\delta\}$. In $\mathrm{ACP}^{\mathrm{c}}$, it is further assumed that a fixed but arbitrary commutative and associative *communication* function $\mid : \mathsf{A}_\delta \times \mathsf{A}_\delta \to \mathsf{A}_\delta$, such that $\delta \mid a = \delta$ for all $a \in \mathsf{A}_\delta$, has been given. The function $\mid$ is regarded to give the result of synchronously performing any two actions for which this is possible, and to be δ otherwise.

The theory $\mathrm{ACP}^{\mathrm{c}}$ is an extension of $\mathrm{BPA}_\delta^{\mathrm{c}}$. It has the constants and operators of $\mathrm{BPA}_\delta^{\mathrm{c}}$ and in addition:

Table 4. Additional axioms for ACP$^{\mathrm{c}}$ $(a, b, c \in \mathsf{A}_\delta)$

$x \parallel y = x \mathbin{\|\!\lfloor} y + y \mathbin{\|\!\lfloor} x + x \mid y$	CM1	$\partial_H(a) = a$	if $a \notin H$	D1
$a \mathbin{\|\!\lfloor} x = a \cdot x$	CM2	$\partial_H(a) = \delta$	if $a \in H$	D2
$a \cdot x \mathbin{\|\!\lfloor} y = a \cdot (x \parallel y)$	CM3	$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$		D3
$(x + y) \mathbin{\|\!\lfloor} z = x \mathbin{\|\!\lfloor} z + y \mathbin{\|\!\lfloor} z$	CM4	$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$		D4
$a \cdot x \mid b = (a \mid b) \cdot x$	CM5			
$a \mid b \cdot x = (a \mid b) \cdot x$	CM6	$(\phi :\rightarrow x) \mathbin{\|\!\lfloor} y = \phi :\rightarrow (x \mathbin{\|\!\lfloor} y)$		GC8
$a \cdot x \mid b \cdot y = (a \mid b) \cdot (x \parallel y)$	CM7	$(\phi :\rightarrow x) \mid y = \phi :\rightarrow (x \mid y)$		GC9
$(x + y) \mid z = x \mid z + y \mid z$	CM8	$x \mid (\phi :\rightarrow y) = \phi :\rightarrow (x \mid y)$		GC10
$x \mid (y + z) = x \mid y + x \mid z$	CM9	$\partial_H(\phi :\rightarrow x) = \phi :\rightarrow \partial_H(x)$		GC11
$a \mid b = b \mid a$	C1			
$(a \mid b) \mid c = a \mid (b \mid c)$	C2			
$\delta \mid a = \delta$	C3			

- the binary *parallel composition* operator $\parallel : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- the binary *left merge* operator $\mathbin{\|\!\lfloor} : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- the binary *communication merge* operator $\mid : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- for each $H \subseteq \mathsf{A}$, the unary *encapsulation* operator $\partial_H : \mathbf{P} \to \mathbf{P}$.

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

The constants and operators of ACP$^{\mathrm{c}}$ to build terms of sort $\mathbf{P}$ are the constants and operators of ACP and additionally the guarded command operator.

Let p and q be closed terms of ACP$^{\mathrm{c}}$. Intuitively, the additional operators can be explained as follows:

- $p \parallel q$ behaves as the process that proceeds with p and q in parallel;
- $p \mathbin{\|\!\lfloor} q$ behaves the same as $p \parallel q$, except that it starts with performing an action of p;
- $p \mid q$ 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$^{\mathrm{c}}$ are the axioms of BPA$^{\mathrm{c}}_\delta$ 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$^{\mathrm{c}}$ (i.e. $a, b, c \in \mathsf{A}_\delta$). In D1–D4, H stands for an arbitrary subset of A. So, D3 and D4 are axiom schemas as well.

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

A well-known subtheory of ACP is PA, ACP without communication. Likewise, we have a subtheory of ACP$^{\mathrm{c}}$, to wit PA$^{\mathrm{c}}$. The theory PA$^{\mathrm{c}}$ is ACP$^{\mathrm{c}}$ without the communication merge operator, without axioms CM5–CM9 and C1–C3, and

Table 5. Additional transition rules for ACP$^{\mathrm{c}}$

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{x \parallel y \xrightarrow{[\phi]\,a} y} \qquad \frac{y \xrightarrow{[\phi]\,a} \surd}{x \parallel y \xrightarrow{[\phi]\,a} x} \qquad \frac{x \xrightarrow{[\phi]\,a} x'}{x \parallel y \xrightarrow{[\phi]\,a} x' \parallel y} \qquad \frac{y \xrightarrow{[\phi]\,a} y'}{x \parallel y \xrightarrow{[\phi]\,a} x \parallel y'}$$

$$\frac{x \xrightarrow{[\phi]\,a} \surd,\; y \xrightarrow{[\psi]\,b} \surd}{x \parallel y \xrightarrow{[\phi \sqcap \psi]\,c} \surd}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot \qquad \frac{x \xrightarrow{[\phi]\,a} \surd,\; y \xrightarrow{[\psi]\,b} y'}{x \parallel y \xrightarrow{[\phi \sqcap \psi]\,c} y'}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot$$

$$\frac{x \xrightarrow{[\phi]\,a} x',\; y \xrightarrow{[\psi]\,b} \surd}{x \parallel y \xrightarrow{[\phi \sqcap \psi]\,c} x'}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot \qquad \frac{x \xrightarrow{[\phi]\,a} x',\; y \xrightarrow{[\psi]\,b} y'}{x \parallel y \xrightarrow{[\phi \sqcap \psi]\,c} x' \parallel y'}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot$$

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{x \mathbin{\|\!\!\!\lfloor} y \xrightarrow{[\phi]\,a} y} \qquad \frac{x \xrightarrow{[\phi]\,a} x'}{x \mathbin{\|\!\!\!\lfloor} y \xrightarrow{[\phi]\,a} x' \parallel y}$$

$$\frac{x \xrightarrow{[\phi]\,a} \surd,\; y \xrightarrow{[\psi]\,b} \surd}{x \mid y \xrightarrow{[\phi \sqcap \psi]\,c} \surd}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot \qquad \frac{x \xrightarrow{[\phi]\,a} \surd,\; y \xrightarrow{[\psi]\,b} y'}{x \mid y \xrightarrow{[\phi \sqcap \psi]\,c} y'}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot$$

$$\frac{x \xrightarrow{[\phi]\,a} x',\; y \xrightarrow{[\psi]\,b} \surd}{x \mid y \xrightarrow{[\phi \sqcap \psi]\,c} x'}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot \qquad \frac{x \xrightarrow{[\phi]\,a} x',\; y \xrightarrow{[\psi]\,b} y'}{x \mid y \xrightarrow{[\phi \sqcap \psi]\,c} x' \parallel y'}\; a \mid b = c,\; \phi \sqcap \psi \neq \bot$$

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{\partial_H(x) \xrightarrow{[\phi]\,a} \surd}\; a \notin H \qquad \frac{x \xrightarrow{[\phi]\,a} x'}{\partial_H(x) \xrightarrow{[\phi]\,a} \partial_H(x')}\; a \notin H$$

with axiom CM1 replaced by $x \parallel y = x \mathbin{\|\!\!\!\lfloor} y + y \mathbin{\|\!\!\!\lfloor} x$ (M1). In other words, the possibility that actions are performed synchronously is not covered by PA$^{\mathrm{c}}$.

The structural operational semantics of ACP$^{\mathrm{c}}$ is described by the transition rules for BPA$^{\mathrm{c}}_{\delta}$ and the additional transition rules given in Table 5.

Splitting bisimilarity is a congruence with respect to parallel composition, left merge, communication merge and encapsulation.

Proposition 2 (Congruence). *For all $p, p', q, q' \in \mathcal{T}_{\mathbf{P}}$, $p \mathrel{\underline{\leftrightarrow}} q$ and $p' \mathrel{\underline{\leftrightarrow}} q'$ implies $p \parallel p' \mathrel{\underline{\leftrightarrow}} q \parallel q'$, $p \mathbin{\|\!\!\!\lfloor} p' \mathrel{\underline{\leftrightarrow}} q \mathbin{\|\!\!\!\lfloor} q'$, $p \mid p' \mathrel{\underline{\leftrightarrow}} q \mid q'$ and $\partial_H(p) \mathrel{\underline{\leftrightarrow}} \partial_H(q)$.*

The axioms of ACP$^{\mathrm{c}}$ constitute a sound and complete axiomatization of splitting bisimilarity.

Theorem 3 (Soundness). *For all $p, q \in \mathcal{T}_{\mathbf{P}}$,* ACP$^{\mathrm{c}}$ $\vdash p = q$ *implies $p \mathrel{\underline{\leftrightarrow}} q$.*

Theorem 4 (Completeness). *For all $p, q \in \mathcal{T}_{\mathbf{P}}$, $p \mathrel{\underline{\leftrightarrow}} q$ implies* ACP$^{\mathrm{c}}$ $\vdash p = q$.

Proof. The proof follows the same line as the completeness proof for ACP given in [16]. □

4 Guarded Recursion

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

Table 6. Axioms for recursion

$\langle X \vert E\rangle = \langle t_X \vert E\rangle$	if $X = t_X \in E$	RDP
$E \Rightarrow X = \langle X \vert E\rangle$	if $X \in \mathrm{V}(E)$	RSP

Table 7. Transition rules for recursion

$$\frac{\langle t_X|E\rangle \xrightarrow{[\phi]\,a} \surd}{\langle X|E\rangle \xrightarrow{[\phi]\,a} \surd}\; X{=}t_X \in E \qquad \frac{\langle t_X|E\rangle \xrightarrow{[\phi]\,a} x'}{\langle X|E\rangle \xrightarrow{[\phi]\,a} x'}\; X{=}t_X \in E$$

A *recursive specification* over ACP$^{\mathrm{c}}$ is a set of 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$^{\mathrm{c}}$ that only contains variables from V. We write $\mathrm{V}(E)$ 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$^{\mathrm{c}}$) $\{P_X \mid X \in \mathrm{V}(E)\}$ such that the equations of E hold if, for all $X \in \mathrm{V}(E)$, X stands for P_X. Let t be a term of ACP$^{\mathrm{c}}$ containing a variable X. We call an occurrence of X in t *guarded* if t has a subterm of the form $a \cdot t'$ containing this occurrence of X. A recursive specification over ACP$^{\mathrm{c}}$ 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 ACP$^{\mathrm{c}}$ and the equations of the recursive specification.

For each guarded recursive specification E and each variable $X \in \mathrm{V}(E)$, we introduce a constant of sort $\mathbf{P}$ standing for the unique solution of E for X. This constant is denoted by $\langle X|E\rangle$. We will also use the following notation. Let t be a term of ACP$^{\mathrm{c}}$ and E be a guarded recursive specification over ACP$^{\mathrm{c}}$. Then we write $\langle t|E\rangle$ for t with, for all $X \in \mathrm{V}(E)$, 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.

Guarded recursive specifications over ACP$^{\mathrm{c}}$ have unique solutions in the model induced by the structural operational semantics of ACP$^{\mathrm{c}}$ extended with guarded recursion.

Theorem 5 (Unique solutions). *Let E be a guarded recursive specifications over* ACP$^{\mathrm{c}}$*. If $\{p_X \mid X \in \mathrm{V}(E)\}$ and $\{q_X \mid X \in \mathrm{V}(E)\}$ are solutions of E, then $p_X \leftrightarrows q_X$ for all $X \in \mathrm{V}(E)$.*

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

$\mathsf{CE}_h(a) = a$	CE1	$\mathsf{CE}_h(\bot) = \bot$	CE6
$\mathsf{CE}_h(a \cdot x) = a \cdot \mathsf{CE}_h(x)$	CE2	$\mathsf{CE}_h(\top) = \top$	CE7
$\mathsf{CE}_h(x + y) = \mathsf{CE}_h(x) + \mathsf{CE}_h(y)$	CE3	$\mathsf{CE}_h(\eta) = \eta' \quad \text{if } h(\eta) = \eta'$	CE8
$\mathsf{CE}_h(\phi :\to x) = \mathsf{CE}_h(\phi) :\to \mathsf{CE}_h(x)$	CE4	$\mathsf{CE}_h(-\phi) = -\mathsf{CE}_h(\phi)$	CE9
$\mathsf{CE}_h(\mathsf{CE}_{h'}(x)) = \mathsf{CE}_{h \circ h'}(x)$	CE5	$\mathsf{CE}_h(\phi \sqcup \psi) = \mathsf{CE}_h(\phi) \sqcup \mathsf{CE}_h(\psi)$	CE10
		$\mathsf{CE}_h(\phi \sqcap \psi) = \mathsf{CE}_h(\phi) \sqcap \mathsf{CE}_h(\psi)$	CE11

5 Evaluation of Conditions

Guarded commands cannot always be eliminated from closed terms of ACP$^{\mathrm{c}}$ 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.

There are unary *condition evaluation* operators $\mathsf{CE}_h{:}\mathbf{P} \to \mathbf{P}$ and $\mathsf{CE}_h{:}\mathbf{C} \to \mathbf{C}$ for each endomorphisms h of $\mathcal{C}$.

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

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

The additional axioms for CE_h, where $h \in \mathcal{H}$, are the axioms given in Table 8.

Example 2. We return to Example 1, which is concerned with a pedestrian who uses a crossing with traffic lights to cross a road with busy traffic safely. Let h_g be such that $h_g(\mathit{green}) = \top$ and $h_g(\mathit{red}) = \bot$; and let h_r be such that $h_r(\mathit{green}) = \bot$ and $h_r(\mathit{red}) = \top$. Then we can derive the following:

$$\mathsf{CE}_{h_g}(\mathit{PED}) = \mathit{arrive} \cdot \mathit{cross} \quad \text{and} \quad \mathsf{CE}_{h_r}(\mathit{PED}) = \mathit{arrive} \cdot \mathit{make\text{-}req} \cdot \delta\ .$$

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 CE_h cannot deal with that. We will return to this issue in Example 3.

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

The elements of $\mathcal{C}$ can be used to represent equivalence classes with respect to logical equivalence of the set of all propositions with elements of C_{at} as propositional variables and with finite conjunctions and disjunctions. We write $\mathcal{P}$ for this set of propositions. It is likely that there is a theory Φ about the atomic

Table 9. Transition rules for condition evaluation

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{\mathsf{CE}_h(x) \xrightarrow{[h(\phi)]\,a} \surd}\; h(\phi) \neq \bot \qquad \frac{x \xrightarrow{[\phi]\,a} x'}{\mathsf{CE}_h(x) \xrightarrow{[h(\phi)]\,a} \mathsf{CE}_h(x')}\; h(\phi) \neq \bot$$

conditions in the shape of a set of propositions. Let $\Phi \subset \mathcal{P}$, and let $h_\Phi \in \mathcal{H}$ be such that for all $\alpha, \beta \in \mathcal{C}$:

$$\Phi \vdash \langle\!\langle h_\Phi(\alpha) \rangle\!\rangle \Leftrightarrow \langle\!\langle \alpha \rangle\!\rangle \quad \text{and} \quad h_\Phi(\alpha) = h_\Phi(\beta) \text{ iff } \Phi \vdash \langle\!\langle \alpha \rangle\!\rangle \Leftrightarrow \langle\!\langle \beta \rangle\!\rangle \qquad (1)$$

where $\langle\!\langle \alpha \rangle\!\rangle$ is a representative of the equivalence class of propositions isomorphic to α. Then we have $h_\Phi(\alpha) = \top$ iff $\langle\!\langle \alpha \rangle\!\rangle$ is derivable from Φ and $h_\Phi(\alpha) = \bot$ iff $\neg \langle\!\langle \alpha \rangle\!\rangle$ is derivable from Φ. The image of $\mathcal{C}$ under h_Φ is $\mathbb{B}$ 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 $\mathcal{C}$ under the different endomorphisms satisfying (1) are isomorphic subalgebras of $\mathcal{C}$. Moreover, if both h and h' satisfy (1), then $\Phi \vdash \langle\!\langle h(\alpha) \rangle\!\rangle \Leftrightarrow \langle\!\langle h'(\alpha) \rangle\!\rangle$ for all $\alpha \in \mathcal{C}$.

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 $\alpha \in \mathcal{C}$, $\underline{\alpha}$ stands for an arbitrary closed term of sort $\mathbf{C}$ of which the value in $\mathcal{C}$ is α.

Proposition 3 (Condition evaluation on the basis of a theory). *Let $\Phi \subset \mathcal{P}$ be a complete theory and let p be a closed term of* ACP$^\mathrm{c}$*. Then $\mathsf{CE}_{h_\Phi}(p) = p'$ where p' is p with, for all $\alpha \in \mathcal{C}$, in all subterms of the form $\underline{\alpha} :\rightarrow q$, $\underline{\alpha}$ replaced by $\top$ if $\Phi \vdash \langle\!\langle \alpha \rangle\!\rangle$ and $\underline{\alpha}$ replaced by $\bot$ if $\Phi \vdash \neg \langle\!\langle \alpha \rangle\!\rangle$.*

Proof. This result follows immediately from the definition of h_Φ and the distributivity of CE_{h_Φ} over all operators of ACP$^\mathrm{c}$. □

In μCRL [17], an extension of ACP which includes conditional expressions, we find a formalization of the substitution-based alternative for CE_{h_Φ}.

The substitution-based alternative works properly because condition evaluation by means of a 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 1. Let $h \in \mathcal{H}$. Then h induces a theory $\Phi \subset \mathcal{P}$ such that $h = h_\Phi$, viz. the theory Φ defined by

$$\Phi = \{\langle\!\langle h(\alpha) \rangle\!\rangle \Leftrightarrow \langle\!\langle \alpha \rangle\!\rangle \mid \alpha \in \mathcal{C}\} \cup \{\langle\!\langle \alpha \rangle\!\rangle \Leftrightarrow \langle\!\langle \beta \rangle\!\rangle \mid h(\alpha) = h(\beta)\} \,.$$

Consequently, condition evaluation by means of the condition evaluation operators introduced above is always condition evaluation of which the result can be determined from a set of propositions.

Table 10. Axioms for generalized condition evaluation ($a \in \mathsf{A}_\delta$)

$\mathsf{GCE}_h(a) = a$	GCE1
$\mathsf{GCE}_h(a \cdot x) = a \cdot \mathsf{GCE}_{\mathsf{eff}(a,h)}(x)$	GCE2
$\mathsf{GCE}_h(x + y) = \mathsf{GCE}_h(x) + \mathsf{GCE}_h(y)$	GCE3
$\mathsf{GCE}_h(\phi :\rightarrow x) = \mathsf{CE}_h(\phi) :\rightarrow \mathsf{GCE}_h(x)$	GCE4

Table 11. Transition rules for generalized condition evaluation

$$\frac{x \xrightarrow{[\phi]\,a} \surd}{\mathsf{GCE}_h(x) \xrightarrow{[h(\phi)]\,a} \surd}\; h(\phi) \neq \bot \qquad \frac{x \xrightarrow{[\phi]\,a} x'}{\mathsf{GCE}_h(x) \xrightarrow{[h(\phi)]\,a} \mathsf{GCE}_{\mathsf{eff}(a,h)}(x')}\; h(\phi) \neq \bot$$

We proceed with generalizing the condition evaluation operators introduced above. It is assumed that a fixed but arbitrary function $\mathsf{eff} : \mathsf{A} \times \mathcal{H} \to \mathcal{H}$ has been given.

There is a unary *generalized condition evaluation* operator $\mathsf{GCE}_h : \mathbf{P} \to \mathbf{P}$ for each $h \in \mathcal{H}$; and there is again the unary operator $\mathsf{CE}_h : \mathbf{C} \to \mathbf{C}$ for each $h \in \mathcal{H}$.

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

The additional axioms for GCE_h, where $h \in \mathcal{H}$, are the axioms given in Table 10 and axioms CE6–CE11 from Table 8.

Example 3. We return to Example 1, which is concerned with a pedestrian who uses a crossing with traffic lights to cross a road with busy traffic safely. In Example 2, we illustrated that the condition evaluation operators 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 GCE_h can deal with such a change. Let h_g and h_r be as in Example 2; and let eff be such that $\mathsf{eff}(\mathit{make\text{-}req}, h_r) = h_g$ and $\mathsf{eff}(a, h) = h$ otherwise. Then we can derive the following:

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

The change from red light to green light is due to interaction between the pedestrian and the traffic lights.

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

We can add both the condition evaluation operators and the generalized condition evaluation operators to $\mathrm{ACP}^{\mathrm{c}}$. However, Proposition 4 stated below makes it clear that the latter operators supersede the former operators.

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

As their name suggests, the generalized condition evaluation operators are generalizations of the condition evaluation operators.

Proposition 4 (Generalization). *We can fix the function* eff *such that* $\mathsf{GCE}_h(x) = \mathsf{CE}_h(x)$ *for all* $h \in \mathcal{H}$.

Proof. Clearly, if $\mathsf{eff}(a, h') = h'$ for all $a \in \mathsf{A}$ and $h' \in \mathcal{H}$, then $\mathsf{GCE}_h(x) = \mathsf{CE}_h(x)$ for all $h \in \mathcal{H}$. $\square$

The state operators that are added to $\mathrm{ACP^c}$ in Sect. 6 are in their turn generalizations of the generalized condition evaluation operators.

6 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 [18]. The main difference with those operators is that guarded commands are taken into account.

It is assumed that a fixed but arbitrary set S of *states* has been given, together with functions $\mathsf{act} : \mathsf{A} \times S \to \mathsf{A}_\delta$, $\mathsf{eff} : \mathsf{A} \times S \to S$ and $\mathsf{eval} : \mathcal{C} \times S \to \mathcal{C}$, where, for each $s \in S$, the function $h_s : \mathcal{C} \to \mathcal{C}$ defined by $h_s(\alpha) = \mathsf{eval}(\alpha, s)$ is an endomorphism of $\mathcal{C}$.

There are unary *state operators* $\lambda_s : \mathbf{P} \to \mathbf{P}$ and $\lambda_s : \mathbf{C} \to \mathbf{C}$ for each $s \in S$.

The state operator λ_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 act gives, for each action a and state s, the action that results from executing a in state s. The function eff gives, for each action a and state s, the state that results from executing a in state s. The function eval gives, for each condition α and state s, the condition that results from evaluating α in state s. The functions act and eff are extended to A_δ such that $\mathsf{act}(\delta, s) = \delta$ and $\mathsf{eff}(\delta, s) = s$ for all $s \in S$.

The additional axioms for λ_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 [18].

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

We can add, in addition to the state operators, the condition evaluation operators and/or the generalized condition evaluation operators from Sect. 5 to $\mathrm{ACP^c}$.

The state operators are generalizations of the generalized condition evaluation operators from Sect. 5.

Table 12. Axioms for state operators ($a \in \mathsf{A}_\delta$, $\eta \in \mathsf{C}_{\mathsf{at}}$, $\eta' \in \mathsf{C}_{\mathsf{at}} \cup \{\bot, \top\}$)

$\lambda_s(a) = \mathsf{act}(a,s)$	SO1	$\lambda_s(\bot) = \bot$	SO5
$\lambda_s(a \cdot x) = \mathsf{act}(a,s) \cdot \lambda_{\mathsf{eff}(a,s)}(x)$	SO2	$\lambda_s(\top) = \top$	SO6
$\lambda_s(x + y) = \lambda_s(x) + \lambda_s(y)$	SO3	$\lambda_s(\eta) = \eta' \quad \text{if } \mathsf{eval}(\eta, s) = \eta'$	SO7
$\lambda_s(\phi :\rightarrow x) = \lambda_s(\phi) :\rightarrow \lambda_s(x)$	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

$$\frac{x \xrightarrow{[\phi]\, a} \surd}{\lambda_s(x) \xrightarrow{[\mathsf{eval}(\phi,s)]\ \mathsf{act}(a,s)} \surd}\ \mathsf{act}(a,s) \neq \delta,\ \mathsf{eval}(\phi,s) \neq \bot$$

$$\frac{x \xrightarrow{[\phi]\, a} x'}{\lambda_s(x) \xrightarrow{[\mathsf{eval}(\phi,s)]\ \mathsf{act}(a,s)} \lambda_{\mathsf{eff}(a,s)}(x')}\ \mathsf{act}(a,s) \neq \delta,\ \mathsf{eval}(\phi,s) \neq \bot$$

Proposition 5 (Generalization). *We can fix S,* act, eff *and* eval *such that, for some $f : \mathcal{H} \to S$, $\lambda_{f(h)}(x) = \mathsf{GCE}_h(x)$ holds for all $h \in \mathcal{H}$.*

Proof. Clearly, if $S = \mathcal{H}$, f is the identity function on $\mathcal{H}$, and $\mathsf{act}(a,s) = a$, $\mathsf{eff}(a,s) = \mathsf{eff}(a, f^{-1}(s))$ and $\mathsf{eval}(\alpha, s) = f^{-1}(s)(\alpha)$ for all $a \in \mathsf{A}$, $s \in S$ and $\alpha \in \mathcal{C}$, then $\lambda_{f(h)}(x) = \mathsf{GCE}_h(x)$ holds for all $h \in \mathcal{H}$. □

Notice that, in so far as condition evaluation is concerned, the state operators do not add anything to the generalized condition evaluation operators.

7 Concluding Remarks

Conditional expressions of the form $\zeta :\rightarrow p$ are not new. They were added to ACP for the first time in [3]. In [4], 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 a variant of bisimilarity that was defined for the first time in [4]. The formulation given here is closer to the one given in [5]. State operators were added to ACP for the first time in [18]. The condition evaluation operators and the generalized condition evaluation operators were introduced for the first time in [11]. We are not aware of other work studying condition evaluation in a process algebra with conditional expressions.

In ACP$^{\mathrm{c}}$, like in ACPps [6], conditional expressions give rise to the inclusion of conditional transitions in the behaviour being described, whereas in most other process algebraic formalisms that include conditional expressions, they concern the conditional inclusion of unconditional transitions (see e.g. μCRL [19]). ACP$^{\mathrm{c}}$, like ACPps, is a development following ideas from [4]. ACP$^{\mathrm{c}}$ is based on a more abstract view on conditions than ACPps, but it lacks signal emission – a mechanism from ACPps that allows for a kind of condition evaluation.

References

1. Bergstra, J.A., Klop, J.W.: Process algebra for synchronous communication. Information and Control **60** (1984) 109–137
2. Baeten, J.C.M., Weijland, W.P.: Process Algebra. Volume 18 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (1990)
3. Baeten, J.C.M., Bergstra, J.A., Mauw, S., Veltink, G.J.: A process specification formalism based on static COLD. In Bergstra, J.A., Feijs, L.M.G., eds.: Algebraic Methods II: Theory, Tools and Applications. Volume 490 of Lecture Notes in Computer Science., Springer-Verlag (1991) 303–335
4. Baeten, J.C.M., Bergstra, J.A.: Process algebra with signals and conditions. In Broy, M., ed.: Programming and Mathematical Methods. Volume F88 of NATO ASI Series., Springer-Verlag (1992) 273–323
5. Bergstra, J.A., Ponse, A., van Wamel, J.J.: Process algebra with backtracking. In de Bakker, J.W., de Roever, W.P., Rozenberg, G., eds.: A Decade of Concurrency (Reflections and Perspectives). Volume 803 of Lecture Notes in Computer Science., Springer-Verlag (1994) 46–91
6. Baeten, J.C.M., Bergstra, J.A.: Process algebra with propositional signals. Theoretical Computer Science **177** (1997) 381–405
7. Hennessy, M., Milner, R.: Algebraic laws for non-determinism and concurrency. Journal of the ACM **32** (1985) 137–161
8. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. Journal of the ACM **31** (1984) 560–599
9. Bergstra, J.A., Ponse, A.: Process algebra and conditional composition. Information Processing Letters **80** (2001) 41–49
10. van der Zwaag, M.B.: Models and Logics for Process Algebra. PhD thesis, Programming Research Group, University of Amsterdam, Amsterdam (2002)
11. Bergstra, J.A., Middelburg, C.A.: Splitting bisimulations and retrospective conditions. Computer Science Report 05-03, Department of Mathematics and Computer Science, Eindhoven University of Technology (2005)
12. Monk, J.D., Bonnet, R., eds.: Handbook of Boolean Algebras. Volume 1. Elsevier, Amsterdam (1989)
13. Hoare, C.A.R., Hayes, I.J., He Jifeng, Morgan, C.C., Roscoe, A.W., Sanders, J.W., Sorensen, I.H., Spivey, J.M., Sufrin, B.A.: Laws of programming. Communications of the ACM **30** (1987) 672–686
14. Halmos, P.R.: Lectures on Boolean Algebras. Mathematical Studies. Van Nostrand, Princeton, NJ (1963)
15. Busi, N., van Glabbeek, R.J., Gorrieri, R.: Axiomatising ST-bisimulation semantics. In Olderog, R.R., ed.: PROCOMET'94. Volume 56 of IFIP Transactions A., North-Holland (1994) 169–188
16. Baeten, J.C.M., Verhoef, C.: Concrete process algebra. In Abramsky, S., Gabbay, D.M., Maibaum, T.S.E., eds.: Handbook of Logic in Computer Science. Volume IV. Oxford University Press, Oxford (1995) 149–268
17. Groote, J.F., Ponse, A.: Proof theory for μCRL: A language for processes with data. In Andrews, D.J., Groote, J.F., Middelburg, C.A., eds.: Semantics of Specification Languages. Workshops in Computing Series, Springer-Verlag (1994) 232–251
18. Baeten, J.C.M., Bergstra, J.A.: Global renaming operators in concrete process algebra. Information and Control **78** (1988) 205–245
19. Groote, J.F., Ponse, A.: The syntax and semantics of μCRL. In Ponse, A., Verhoef, C., van Vlijmen, S.F.M., eds.: Algebra of Communicating Processes 1994. Workshops in Computing Series, Springer-Verlag (1995) 26–62