Delayed choice for process algebra with abstraction

P. R. D'Argenio[1] and S. Mauw[2]

[1] Depto. de Informática, Fac. de Cs. Exactas, Universidad Nacional de La Plata. CC 11 (1900) La Plata. Buenos Aires. Argentina.
pedro@info.unlp.edu.ar

[2] Dept. of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
sjouke@win.tue.nl

Abstract. The delayed choice is an operator which serves to combine linear time and branching time within one process algebra. We study this operator in a theory with abstraction, more precisely, in a setting considering branching bisimulation. We show its use in scenario specifications and in verification to reduce irrelevant branching structure of a process.

1 Introduction

The delayed choice is an operator that allows one to express linear time aspects in a branching time process algebra. It was introduced in [3] for a basic process algebra without abstraction. The intuition behind this operator for alternative composition is the following. If two processes start with a common initial action, then the delayed choice between these alternatives consists of executing this common action before making the choice between the resulting processes. This property is best displayed in the following equation. The delayed choice is denoted by $\mp$ (for Trace-+) and the normal non-deterministic choice by $+$.

$$ab \mp ac = a(b + c)$$

If the two alternatives have no initial action in common, the delayed choice and the non-deterministic choice coincide ($a \neq c$):

$$ab \mp cd = ab + cd$$

In [3] soundness and completeness of the definition was proven and an application in the realm of Message Sequence Charts was given.

In this paper we study the delayed choice operator in a process algebra theory extended with abstraction. In this setting, the delayed choice operator should also remove non-determinism due to internal steps. This property can be expressed as follows:

$$\tau a \mp b = \tau(a + b)$$

The behaviour of the delayed choice operator with respect to internal steps compares well to the behaviour of the deterministic choice operator $\Box$ from

TCSP [8]. This operator was studied in a branching time setting in [9], where it was called τ-angelic choice.

The main purpose of this paper is to show that the definition of the delayed choice operator can be combined with the definition of the τ-angelic choice operator in order to obtain a delayed choice operator for process algebra with abstraction.

We use branching bisimulation [11] as the semantics for the silent step. We consider divergence free processes only. The case of weak bisimulation is treated in [10].

Applications of this new operator can be found in the areas of specification and verification. Using the delayed choice it is possible to make so-called *scenario specifications*. A scenario specification consists of a collection of possible behaviours of a system. If two scenarios share an initial action, it is in general not the intention to specify a non-deterministic choice between these scenarios. For example, a possible scenario for a vending machine could be the insertion of a coin followed by choosing coffee and another scenario could be the insertion of a coin followed by choosing tea. The intention is not to express that the choice between coffee and tea is made by inserting the coin, which is the interpretation when combining these scenarios with a non-deterministic choice. Rather it is to express that the selection is made after paying. This can be expressed with the delayed choice.

The second application of the delayed choice operator is in verification. A verification in process algebra in most cases consists of a proof that an abstraction of some implementation specification is equivalent to a given requirements specification. Often the structure of such a requirements specification is quite complex due to the presence of an excess of internal choices, some of which may not be relevant for the insight that the implementation is correct. These less interesting choices between internal actions can be filtered out using the delayed choice, without adopting linear time semantics for the complete system. We give an example in Sect. 4.

This paper is structured as follows. In Sect. 2 we introduce the basic theory $\mathrm{BPA}_{\delta\varepsilon}$ and extend it with the silent step τ. We consider strong bisimulation and branching bisimulation as semantics. Next, we define the delayed choice operator and give an operational semantics in Sect. 3. We prove soundness, completeness and several other properties. Finally, we give some examples in Sect. 4.

We thank Jos Baeten and Michel Reniers for their valuable comments on drafts of this paper and Rob van Glabbeek for answering some technical questions. Jan Joris Vereijken was very helpful in doing calculations on the examples.

2 Basic Process Algebra with Empty Process

The aim of this section is to introduce the algebra of sequential processes [5]. We deal with the basic process algebra with empty process for concrete processes ($\mathrm{BPA}_{\delta\varepsilon}$) [6, 15] and with abstraction in the framework of branching bisimulation ($\mathrm{BPA}^{\tau}_{\delta\varepsilon}$) [11].

2.1 The Equational Theories

The signature of the several theories is parameterized by a set of constants $A = \{a, b, \ldots\}$ called *atomic actions*. There are three distinguished constants not belonging to A. They are δ, called *deadlock* or inaction, that denotes the process that has stopped executing actions and cannot proceed; ε, the *empty process*, that denotes the process that does nothing but terminate successfully; and τ, the *silent action*, that is a special action having the meaning of internal activity. Besides, the signature has two binary operators: the *alternative composition* $(+)$ which, in $x + y$, executes process x or y, but not both; and the *sequential composition* $(\cdot)$ that, given $x \cdot y$, first executes x and, upon completion, starts with the execution of y. We generally omit this operator writing xy instead of $x \cdot y$. Besides, we assume that $\cdot$ binds stronger than all the other operators we will deal with, and $+$ binds weaker. Notice that the signature of $\mathrm{BPA}_{\delta\varepsilon}$ also includes the silent action τ. It is dealt with as any other action in $\mathrm{BPA}_{\delta\varepsilon}$. Equations A1–A9 from Table 1 define $\mathrm{BPA}_{\delta\varepsilon}$. Adding axiom BE, we obtain $\mathrm{BPA}^{\tau}_{\delta\varepsilon}$.

Table 1. Axioms for $\mathrm{BPA}_{\delta\varepsilon}$ and $\mathrm{BPA}^{\tau}_{\delta\varepsilon}$.

A1 $x + y = y + x$	A6 $x + \delta = x$
A2 $(x + y) + z = x + (y + z)$	A7 $\delta x = \delta$
A3 $x + x = x$	
A4 $(x + y)z = xz + yz$	A8 $x\varepsilon = x$
A5 $(xy)z = x(yz)$	A9 $\varepsilon x = x$
BE $a(\tau(x + y) + x) = a(x + y)$	

2.2 Structured Operational Semantics and Equivalences

Table 2 defines the operational semantics in a structured way following the style of [17]. In our system we consider two kinds of predicates, each one having its own meaning. Predicate $\downarrow$ expresses that a process may terminate successfully. For every action $a \in A \cup \{\tau\}$, predicate $\xrightarrow{a}$ expresses that the first argument can perform action a and become the second argument. In addition, we define $\Longrightarrow$ as the reflexive transitive closure of $\xrightarrow{\tau}$.

Table 2. Operational semantics for the basic operators ($a \in A \cup \{\tau\}$)

$$\varepsilon\downarrow \qquad \frac{x\downarrow}{x+y\downarrow \quad y+x\downarrow} \qquad \frac{x\downarrow \quad y\downarrow}{x\cdot y\downarrow}$$

$$a \xrightarrow{a} \varepsilon \qquad \frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x' \quad y+x \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} x'}{x\cdot y \xrightarrow{a} x'\cdot y} \qquad \frac{x\downarrow \quad y \xrightarrow{a} y'}{x\cdot y \xrightarrow{a} y'}$$

In this paper we will deal with divergence free processes only. This means that a process cannot perform an infinite sequence of τ-steps.

Let $\mathcal{T}$ be the set of all closed terms in the signature of $\mathrm{BPA}_{\delta\varepsilon}$. Next, we define two well known equivalences over $\mathcal{T}$.

Definition 1 (Bisimulation). [16] A *(strong) bisimulation* is a symmetric relation $S \subseteq \mathcal{T} \times \mathcal{T}$ satisfying, for all $a \in A \cup \{\tau\}$:

if pSq and $p \xrightarrow{a} p'$, then $\exists q' \in \mathcal{T} : q \xrightarrow{a} q'$ and $p'Sq'$; and
if pSq then $p \downarrow$ iff $q \downarrow$.

Two processes p and q are *bisimilar* (notation $p \underline{\leftrightarrow} q$), if there exists a bisimulation S with pSq.

Definition 2 (Branching bisimulation). [11] A *branching bisimulation* is a symmetric relation $S \subseteq \mathcal{T} \times \mathcal{T}$ satisfying, for all $a \in A \cup \{\tau\}$:

if pSq and $p \xrightarrow{a} p'$, then $\begin{cases} a = \tau \text{ and } p'Sq, \text{ or} \\ \exists q'', q' \in \mathcal{T} : q \Longrightarrow q'' \xrightarrow{a} q' \text{ and } pSq'' \wedge p'Sq'; \text{and} \end{cases}$
if pSq and $p \downarrow$, then $\exists q' \in \mathcal{T} : q \Longrightarrow q' \downarrow$ and pSq'.

Two processes p and q are *branching bisimilar* (notation $p \underline{\leftrightarrow}_b q$), if there exists a branching bisimulation S with pSq.

Two processes p and q are *rooted branching bisimilar*, (notation $p \underline{\leftrightarrow}_{rb} q$) if for all $a \in A \cup \{\tau\}$:

1. $p \xrightarrow{a} p'$ implies $\exists q' : q \xrightarrow{a} q'$ and $p' \underline{\leftrightarrow}_b q'$;
2. $q \xrightarrow{a} q'$ implies $\exists p' : p \xrightarrow{a} p'$ and $p' \underline{\leftrightarrow}_b q'$;
3. $p \downarrow$ iff $q \downarrow$.

The relations above are ordered by set inclusion: $\underline{\leftrightarrow} \subset \underline{\leftrightarrow}_{rb} \subset \underline{\leftrightarrow}_b$. We have:

Theorem 3 (The term models).

1. *$\mathcal{T}/\underline{\leftrightarrow}$ is a model for $\mathrm{BPA}_{\delta\varepsilon}$. $\mathrm{BPA}_{\delta\varepsilon}$ is a complete axiomatization for $\mathcal{T}/\underline{\leftrightarrow}$.*
2. *$\mathcal{T}/\underline{\leftrightarrow}_{rb}$ is a model for $\mathrm{BPA}^{\tau}_{\delta\varepsilon}$. $\mathrm{BPA}^{\tau}_{\delta\varepsilon}$ is a complete axiomatization for $\mathcal{T}/\underline{\leftrightarrow}_{rb}$.*

3 The Delayed Choice

3.1 Equational Theory

The delayed choice considered here is an extension of the operator introduced in [3]. The difference is that we also consider abstraction. The delayed choice ($\mp$) between processes x and y, is the process obtained by joining the observable common initial parts of x and y and continuing with a normal choice between the remaining parts. In case internal activity is performed, the choice is delayed in the same way the τ-angelic choice [9] does. Thus, after executing an internal step of x the alternatives from y are still enabled, and vice versa. However, the

nondeterministic choices which are internal to x or y, are not removed. This is expressed in the definition of the delayed choice in Table 3.

The definition of the delayed choice has five cases. We use three auxiliary operators. The first one is the join operator ($\between$). $x \between y$ selects exactly those summands of x and y having a common initial action which is observable (i.e. different from τ). The unless operator ($\lhd$) works exactly in the opposite way. In $x \lhd y$, only those summands of x having an initial observable action are selected for which y does not have any summand with the same initial action or with an initial silent action. Note that summands of x having an initial silent step are not selected. The τ-selecting operator ($\sqsubset$) delays the choice in case of silent actions, i.e., $x \sqsubset y$ selects the summands of x having an initial silent action.

Thus, the axioms in Table 3 extend $\mathrm{BPA}_{\delta\varepsilon}$ and $\mathrm{BPA}^{\tau}_{\delta\varepsilon}$ with the delayed choice and the auxiliary operators. We denote these extensions by $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ and $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$.

Table 3. Axioms for delayed choice ($a, b \in A$)

DC	$x \mp y = x \between y + x \lhd y + y \lhd x + x \sqsubset y + y \sqsubset x$		
J1	$\varepsilon \between x = \delta$	U1	$\varepsilon \lhd \varepsilon = \varepsilon$
J2	$x \between \varepsilon = \delta$	U2	$\varepsilon \lhd ax = \varepsilon$
J3	$\delta \between x = \delta$	U3	$ax \lhd \varepsilon = ax$
J4	$x \between \delta = \delta$	U4	$\delta \lhd x = \delta$
J5	$ax \between ay = a(x \mp y)$	U5	$\varepsilon \lhd \delta = \varepsilon$
J6	$a \neq b \Rightarrow ax \between by = \delta$	U6	$ax \lhd \delta = ax$
J7	$(x + y) \between z = x \between z + y \between z$	U7	$ax \lhd ay = \delta$
J8	$x \between (y + z) = x \between y + x \between z$	U8	$a \neq b \Rightarrow ax \lhd by = ax$
		U9	$(x + y) \lhd z = x \lhd z + y \lhd z$
		U10	$x \lhd (y + z) = (x \lhd y) \lhd z$
TJ1	$\tau x \between y = \delta$	TS1	$\varepsilon \sqsubset x = \delta$
TJ2	$x \between \tau y = \delta$	TS2	$\delta \sqsubset x = \delta$
		TS3	$ax \sqsubset y = \delta$
TU1	$\tau x \lhd y = \delta$	TS4	$\tau x \sqsubset y = \tau(x \mp y)$
TU2	$x \lhd \tau y = \delta$	TS5	$(x + y) \sqsubset z = x \sqsubset z + y \sqsubset z$

Operators $\between$, $\lhd$ and $\sqsubset$ are needed for a finite axiomatization. The unless operator $\lhd$ is quite similar to the one used in the axiomatization of the priority operator [2], but our version filters according to equality instead of an ordering on observable actions. The τ-selecting operator $\sqsubset$ works in a similar way as the left box of [9] when dealing with summands starting with τ, but instead, our operator does not select summands having an initially observable action.

Example 1. We give some simple examples in order to make clear the behaviour of the delayed choice. Suppose a, b, c, d, e and f are distinct actions in A, then:

$\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC} \vdash ab \mathbin{\mp} a(c+d) = a(b+c+d)$
$\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC} \vdash (ab + \tau ac) \mathbin{\mp} de = ab + \tau(ac+de)$
$\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC} \vdash (ab + ac) \mathbin{\mp} a(b+c) = a(b+c)$
$\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC} \vdash (ab + \tau ac) \mathbin{\mp} a(d+e) = a(b+d+e) + \tau a(c+d+e)$
$\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC} \vdash (ab + ac) \mathbin{\mp} (ad + \tau f) = a(b+d) + a(c+d) + \tau(ab+ac+f)$
$\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC} \vdash (ab + ac) \mathbin{\mp} a(\tau(b+c)+b) = a(\tau(b+c)+b)$
$\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC} \vdash (ab + ac) \mathbin{\mp} a(\tau(b+c)+b) = a(b+c)$

3.2 Structured Operational Semantics

The rules in Table 4 define the operational semantics for the delayed choice. In some rules, we make use of negative premises (see [18]). Expression $y \overset{a}{\not\longrightarrow}$ means that process y cannot execute action a. Moreover, our system is in *panth format* [18], which introduces several good properties that are useful in proving completeness of equational theories.

Our choice was to formulate the equational theory and afterwards state the operational rules which we will prove sound and complete. However, as the rule system can be simply translated into one in GSOS format [7] by changing the predicate $\downarrow$ into the action relation $\overset{\surd}{\longrightarrow}$ as done in [12], we could follow the algorithm proposed by [1] in order to help us on finding a complete axiomatization starting from the rules.

Table 4. Operational semantics for delayed choice ($a, b \in A$)

$$\frac{x \overset{a}{\longrightarrow} x' \quad y \overset{a}{\longrightarrow} y'}{x \mathbin{\mp} y \overset{a}{\longrightarrow} x' \mathbin{\mp} y' \quad x \bowtie y \overset{a}{\longrightarrow} x' \mathbin{\mp} y'} \qquad \frac{x \overset{a}{\longrightarrow} x' \quad y \overset{a}{\not\longrightarrow} \quad y \overset{\tau}{\not\longrightarrow}}{x \mathbin{\mp} y \overset{a}{\longrightarrow} x' \quad y \mathbin{\mp} x \overset{a}{\longrightarrow} x' \quad x \triangleleft y \overset{a}{\longrightarrow} x'}$$

$$\frac{x \downarrow \quad y \overset{\tau}{\not\longrightarrow}}{x \mathbin{\mp} y \downarrow \quad y \mathbin{\mp} x \downarrow \quad x \triangleleft y \downarrow} \qquad \frac{x \overset{\tau}{\longrightarrow} x'}{x \mathbin{\mp} y \overset{\tau}{\longrightarrow} x' \mathbin{\mp} y \quad y \mathbin{\mp} x \overset{\tau}{\longrightarrow} y \mathbin{\mp} x' \quad x \sqsubset y \overset{\tau}{\longrightarrow} x' \mathbin{\mp} y}$$

3.3 Soundness and Completeness

In this section we prove soundness and completeness of the term models. In order to do that, we use term rewrite techniques. Axioms A3–A9 in Table 1 and all axioms in Table 3 can be observed as rewrite rules, if they are oriented from left to right, i.e. for each axiom $s = t$ we consider the rule $s \rightarrow t$. Nevertheless, this

term rewriting system is not confluent, that is, a term may have two different normal forms. This is due to the fact that e.g. axiom A9 is sometimes needed in the opposite direction. So, we complete the term rewriting system by adding the rewrite rules in Table 5. Note that each new rewrite rule is derivable from the axioms for the delayed choice. Let TRS be the new term rewriting system.

Table 5. Additional rewrite rules ($a \neq b$)

AR1 $a \between a \to a$	AR8 $x \between \tau \to \delta$	AR15 $\varepsilon \lhd a \to \varepsilon$
AR2 $a \between ax \to a(\varepsilon \mp x)$	AR9 $a \lhd a \to \delta$	AR16 $a \lhd \varepsilon \to a$
AR3 $ax \between a \to a(x \mp \varepsilon)$	AR10 $a \lhd ax \to \delta$	AR17 $a \lhd \delta \to a$
AR4 $a \between b \to \delta$	AR11 $ax \lhd a \to \delta$	AR18 $\tau \lhd x \to \delta$
AR5 $a \between bx \to \delta$	AR12 $a \lhd b \to a$	AR19 $x \lhd \tau \to \delta$
AR6 $ax \between b \to \delta$	AR13 $a \lhd bx \to a$	AR20 $a \sqsubset x \to \delta$
AR7 $\tau \between x \to \delta$	AR14 $ax \lhd b \to ax$	AR21 $\tau \sqsubset x \to \tau(\varepsilon \mp x)$

Theorem 4. TRS *is strongly normalizing.*

Proof. This can be proved by applying the method of the lexicographical path ordering [13, 14]. The details can be found in [10]. □

Definition 5 (Basic Terms). Let B be the class of *basic terms* over the theory $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ (or $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$), defined as the smallest class satisfying:

1. $\tau, \varepsilon, \delta \in B, A \subset B$
2. $a \in A, t \in B \Rightarrow a \cdot t \in B$
3. $t \in B \Rightarrow \tau \cdot t \in B$
4. $s, t \in B \Rightarrow s + t \in B$

The next theorem states that for every closed $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ term there exists a basic term (not containing $\mp$, $\between$, $\lhd$ and $\sqsubset$) such that they can be proved equal. That is why it is called the *elimination theorem*.

Theorem 6 (Elimination Theorem in $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$). *Let t be a closed term over $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$. Then, there is a basic term s such that $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC} \vdash t = s$.*

Proof. Because of Theorem 4, t has a normal form s. We prove that such an s is a basic term. Firstly, take into account that it is well known that rules A3-A9 rewrite a closed $\mathrm{BPA}_{\delta\varepsilon}$-term into a basic one. Now, if s contains a $\mp$ then DC can be applied and so s is not in normal form which contradicts our assumption. If s contains $\between$, $\lhd$ or $\sqsubset$, take a smallest sub-term containing one of them, say $s_1 \between s_2$, now we can assume that both sub-terms s_1 and s_2 are already basic

terms, so one of the rules J1–J8, TJ1–TJ2 or AR1–AR8 can be applied. If this sub-term is $s_1 \triangleleft s_2$, one of the rules of U1–U10, TU1–TU2 or AR8–AR19 can be applied. Finally, if this sub-term is $s_1 \sqsubset s_2$, then one of the rules TS1–TS5 or AR20–AR21 can be applied. This concludes the proof. □

Corollary 7 (Elimination Theorem in $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$). *Let t be a closed term over $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$. Then, there is a basic term s such that $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC} \vdash t = s$.*

Let $\mathcal{T}^{\mp}$ be the set of all closed $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ terms. We can immediately extend the notion of the several bisimulation equivalences to $\mathcal{T}^{\mp}$. Now, we have the following results.

Theorem 8 (Congruence). *$\leftrightarroweq$ and $\leftrightarroweq_{rb}$ are congruences for the $\mp$, $\bowtie$, $\triangleleft$ and $\sqsubset$ operators.*

Proof.

($\leftrightarroweq$) The set of operational rules for $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ satisfies the *panth* format of [18] and it is also well founded. It remains to prove that it is stratifiable. As in [3], define the function S that, to each step $t \xrightarrow{a} t'$ and termination option $t\downarrow$, assigns the number of $\mp$ symbols plus the number of $\triangleleft$ symbols in t. It is now easy to prove that S is a strict stratification.

For proving that $\leftrightarroweq_{rb}$ is a congruence, we need the following four properties. Their proof is straightforward.

1. $x \Longrightarrow x' \wedge y \Longrightarrow y'$ if and only if $x \mp y \Longrightarrow x' \mp y'$.
2. $x \sqsubset y \not\xrightarrow{a}$ for all $a \neq \tau$.
3. $x \bowtie y \not\xrightarrow{\tau}$.
4. $x \triangleleft y \not\xrightarrow{\tau}$.

Now, it is tedious but not difficult to prove the following.

($\leftrightarroweq_{rb}$) Take any rooted branching bisimulation R between x and x'. Let Id be the identity relation. Then, the following relations are also rooted branching bisimulations:

$R_1 = \{(z \mp y, z' \mp y) | (z, z') \in R\} \cup R \cup Id$
$R_2 = \{(y \mp z, y \mp z') | (z, z') \in R\} \cup R \cup Id$
$R_3 = \{(x \bowtie y, x' \bowtie y)\} \cup R_1$
$R_4 = \{(y \bowtie x, y \bowtie x')\} \cup R_2$
$R_5 = \{(x \triangleleft y, x' \triangleleft y)\} \cup R$
$R_6 = \{(y \triangleleft x, y \triangleleft x')\} \cup Id$
$R_7 = \{(x \sqsubset y, x' \sqsubset y)\} \cup R_1$
$R_8 = \{(y \sqsubset x, y \sqsubset x')\} \cup R_2$ □

Notice that $\leftrightarroweq_b$ is also a congruence for $\mp$. However, this is not the case for the other operators.

Theorem 9 (Soundness).

1. $\mathcal{T}^{\mp}/\leftrightarroweq \models \mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$
2. $\mathcal{T}^{\mp}/\leftrightarroweq_{rb} \models \mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$

Proof. As usual. For every axiom $s = t$ having free variables in X, we define the relation $R = \{(\sigma(s), \sigma(t)) | \sigma \text{ substitutes variables in } X \text{ to closed terms}\} \cup Id$. It is not difficult to prove that R is a bisimulation or rooted branching bisimulation according to the soundness property we are proving. □

Theorem 10 (Equational Conservative Extension).

1. $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ *is a conservative extension of* $\mathrm{BPA}_{\delta\varepsilon}$.
2. $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$ *is a conservative extension of* $\mathrm{BPA}^{\tau}_{\delta\varepsilon}$.

Proof. The operational conservativity follows since our rules are in *panth* format, and they are pure and well-founded (see [19]). This implies operational conservativity up to $\underline{\leftrightarrow}$ and up to $\underline{\leftrightarrow}_{rb}$. Because the axiomatizations of $\mathrm{BPA}_{\delta\varepsilon}$ and $\mathrm{BPA}^{\tau}_{\delta\varepsilon}$ are sound and complete (Theorem 3), and the axiomatizations of $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ and $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$ are sound (Theorem 9), equational conservativity follows from [19, 4]. □

Theorem 11 (Completeness).

1. $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$ *is a complete axiomatization for* $\mathcal{T}^{\mp}/\underline{\leftrightarrow}$.
2. $\mathrm{BPA}^{\tau}_{\delta\varepsilon} + \mathrm{DC}$ *is a complete axiomatization for* $\mathcal{T}^{\mp}/\underline{\leftrightarrow}_{rb}$.

Proof. Again, following [19, 4] and considering Theorem 6 and Corollary 7, this theorem is a corollary of the previous one. □

3.4 Properties

In this section, we prove several properties that hold for the new operators. Mainly, we show that $\mp$ satisfies common properties of choice operators (commutativity and associativity) and that δ is the neutral element for $\mp$. Nevertheless, idempotency does not hold for $\mp$.

Lemma 12. *The following properties are derivable from* $\mathrm{BPA}_{\delta\varepsilon} + \mathrm{DC}$

1. $ax \mp ay = a(x \mp y)$
2. $a \neq b \Rightarrow ax \mp by = ax + by$

Proof.

1. $ax \mp ay = ax \between ay + ax \triangleleft ay + ay \triangleleft ax + ax \sqsubset ay + ay \sqsubset ax$
   $= ax \between ay + \delta + \delta + \delta + \delta \;=\; a(x \mp y)$
2. Let $a \neq b$. Then
   $ax \mp by = ax \between by + ax \triangleleft by + by \triangleleft ax + ax \sqsubset by + by \sqsubset ax$
   $= \delta + ax + by + \delta + \delta \;=\; ax + by$

□

Remark. From now on, we will assume

$$X = \sum_i a_i x_i + \sum_j \tau x_j + \sum_k \varepsilon \qquad Y = \sum_m b_m y_m + \sum_n \tau y_n + \sum_l \varepsilon$$

with $i \in I$, $j \in J$, $k \in K$, $m \in M$, $n \in N$ and $l \in L$; I, J, K, M, N and L are finite disjoint sets; and $a_i \neq \tau$, $b_m \neq \tau$. In particular, we consider $\sum_{h \in \emptyset} t_h = \delta$, or, by A6, we omit it.

The proof of the following lemma is by straightforward calculations.

Lemma 13. *Let X and Y be as before. Then*

1. $X \between Y = \sum_{i,m(a_i = b_m)} a_i(x_i \mp y_m)$
2. $X \lhd Y = \sum_{i(\forall m. a_i \neq b_m \wedge N = \emptyset)} a_i x_i + \sum_{k(N=\emptyset)} \varepsilon$
3. $X \sqsubset Y = \sum_j \tau(x_j \mp Y)$

Theorem 14 (Neutral Element). *Let x be a closed term. Then*

$$x \mp \delta = \delta \mp x = x$$

Proof. We prove it by induction on the number of symbols of x, say k. The base case ($k = 1$) is left to the reader. Assume X as in the previous remark. For the inductive case we have:

$$\begin{aligned} X \mp \delta &\overset{\text{DC}}{=} X \between \delta + X \lhd \delta + \delta \lhd X + X \sqsubset \delta + \delta \sqsubset X \\ &\overset{\text{J4, U4, TS2}}{=} \delta + X \lhd \delta + \delta + X \sqsubset \delta + \delta \\ &\overset{13}{=} \sum_i a_i x_i + \sum_k \varepsilon + \sum_j \tau(x_j \mp \delta) \\ &\overset{\text{IH}}{=} \sum_i a_i x_i + \sum_k \varepsilon + \sum_j \tau x_j \overset{\text{A1, A2}}{=} X \end{aligned}$$

The second part of the theorem goes analogously. □

Definition 15 (Initial Actions). Define the set of initial action of a given term x as follows:

$$\begin{array}{lll} I(\delta) = \emptyset & I(ax) = \{a\} & I(x + y) = I(x) \cup I(y) \\ I(\varepsilon) = \emptyset & I(\tau x) = \{\tau\} & \end{array}$$

Lemma 16. *Let x and y be any closed terms. Then*

1. $I(x \between y) = (I(x) \cap I(y)) \backslash \{\tau\}$
2. $I(x \lhd y) = \begin{cases} I(x) \backslash (I(y) \cup \{\tau\}) & \text{if } \tau \notin I(y) \\ \emptyset & \text{otherwise} \end{cases}$
3. $I(x \sqsubset y) = \{\tau\} \cap I(x)$
4. $I(x \mp y) = \begin{cases} I(x) \cup I(y) & \text{if } \tau \notin I(x) \cup I(y) \\ I(x) & \text{if } \tau \in I(x) \wedge \tau \notin I(y) \\ I(y) & \text{if } \tau \notin I(x) \wedge \tau \in I(y) \\ I(x) \cap I(y) & \text{if } \tau \in I(x) \cap I(y) \end{cases}$

Proof. It follows from Definition 15 and Lemma 13. □

Lemma 17. *Let x, y and z be any closed terms. Then:*

1. $I(y) = I(z) \Rightarrow x \lhd y = x \lhd z$
2. $x \lhd (y \mp z) = x \lhd (y + z) = (x \lhd y) \lhd z$

Proof. Suppose $\tau \in I(y) = I(z)$, then both y and z has a summand with τ as initial action. Hence

$$x \lhd y \overset{\text{A1, A2, A3}}{=} x \lhd (y + \tau y_j) \overset{\text{U9}}{=} (x \lhd y) \lhd \tau y_j \overset{\text{TU2}}{=} \delta$$

Analogously, $x \lhd z = \delta$. Now, suppose $\tau \notin I(y) = I(z)$. Hence, we can write $y = \sum_m b_m y_m + \sum_l \varepsilon$ and $z = \sum_h c_h z_h + \sum_f \varepsilon$. Suppose X as before. Then

$$X \lhd y \overset{13}{=} \sum_{i(\forall m.\, a_i \neq b_m)} a_i x_i + \sum_k \varepsilon \overset{\text{I(y)=I(z)}}{=} \sum_{i(\forall h.\, a_i \neq c_h)} a_i x_i + \sum_k \varepsilon \overset{13}{=} X \lhd z$$

Part (2) follows from Lemma 16, part (4) and the definition of initial actions taking into account whether τ is an initial action of y and z or not. □

Theorem 18 (Commutativity). *For all closed terms x and y, we have:*

1. $x \between y = y \between x$
2. $x \mp y = y \mp x$

Proof. By mutual induction on the sum of symbols of x and y, we can prove (1); (2) follows directly from (1) and DC. □

Theorem 19 (Associativity). *For all closed terms x, y and z, we have:*

1. $(x \between y) \sqsubset z = x \between (y \sqsubset z) = (x \sqsubset y) \between z = (x \lhd y) \sqsubset z = (x \sqsubset y) \lhd z = \delta$
2. $x \between (y \lhd z) = (x \between y) \lhd z = (x \lhd z) \between y$
3. $x \sqsubset (y \mp z) = (x \sqsubset y) \sqsubset z = (x \sqsubset z) \sqsubset y$
4. $x \between (y \between z) = (x \between y) \between z$
5. $x \mp (y \mp z) = (x \mp y) \mp z$

Proof. Identities of (1) can be deduced from Lemma 13

For (2) consider X and Y as before and $Z = \sum_h c_h z_h + \sum_g \tau z_g + \sum_f \varepsilon$. Now, we have:

$$(X \between Y) \lhd Z \overset{13(1)}{=} \left(\sum_{i,m(a_i = b_m)} a_i (x_i \mp y_m) \right) \lhd Z \overset{13(2)}{=} \sum_{\substack{i,m(a_i = b_m \\ \wedge \forall h.\, a_i \neq c_h \\ \wedge G = \emptyset)}} a_i (x_i \mp y_m)$$

$$\overset{13(1)}{=} \left(\sum_{i(\forall h.\, a_i \neq c_h \wedge G = \emptyset)} a_i x_i + \sum_{k(G = \emptyset)} \varepsilon \right) \between Y \overset{13(2)}{=} (X \lhd Z) \between Y$$

The other equation follows similarly.

Properties (3), (4) and (5) are proved by mutual induction on the sum k of symbols of x, y and z. For details we refer to [10]. □

We have already stated that $\mp$ is commutative, associative and has δ as neutral element. However, the delayed choice presented here is not idempotent and it does not satisfy the several laws of distributivity, just as the delayed choice of [3] and the τ-angelic choice [9]. We will not repeat the counter examples for the following fact given in [3].

Fact 20 *The following equations are* **not** *generally valid in the initial algebra:*

$x \mp x = x$
$(x + y) \mp z = (x \mp z) + (y \mp z)$
$(x \mp y) + z = (x + z) \mp (y + z)$
$(x \mp y)z = xz \mp yz$
$z(x \mp y) = zx \mp zy$

4 Examples

In [3] the delayed choice operator was used for the composition of Message Sequence Charts. In this section, we will show two more examples of its application.

4.1 Scenario specification

In communication protocols it is often the case that one can distinguish one main scenario and several alternative behaviours. If, e.g., the main scenario is a correct transmission, an alternative scenario could be the occurrence of a channel error followed by a retransmission. If both scenarios start with the same initial behaviour, the two alternative scenarios should not be combined with the normal non-deterministic choice (+). By using the delayed choice instead, the moment of choice is put at the point where the scenarios start to differ. In this case the benefit of using the delayed choice is not that it gives a shorter specification, but that it helps in designing and presenting the specification in a more modular way.

Next, we will give an example in which the delayed choice allows for a considerably shorter specification than without this operator. Consider an *access control* consisting of a digital key pad and a (locked) door. A user can enter any sequence of digits. The door may only be opened if the sequence ends in a special four digit code (say, 2908). Let 0–9 denote detection of the indicated key stroke and let *grantaccess* stand for offering the user the option to access, then the following is a specification of the access control.

$$AC = (0+1+2+3+4+5+6+7+8+9) \cdot AC \mp 2 \cdot 9 \cdot 0 \cdot 8 \cdot grantaccess \cdot AC$$

Please notice that this process is executed in parallel with the user behaviour. After selecting 2908, the user is not forced to take access. He can also enter another digit and lose access permission for that moment.

4.2 Requirements reduction

A verification in process algebra in general consists of proving $\tau_I(S) = R$, where S is an implementation specification and R is a requirements specification. The τ_I operator ([5]) is the abstraction operator, which renames actions from the set I into τ. It removes all internal actions, but keeps the internal branching structure. It often happens that one has a very simple requirements specification R in mind, while after calculating $\tau_I(S)$ an expression with an excess of internal choices remains. These internal choices probably represent implementation decisions. Then, there are two obvious ways to proceed. The first is to simply forget about R and consider $\tau_I(S)$ as the requirements specification, having to accept a more implementation directed requirement. The second way is to discard the branching structure and proceed in a linear time semantics, where $\tau_I(S) = R$ holds. In this case we lose all information about the branching structure of the requirements.

We propose to use the delayed choice operator. Let D be the operator which replaces all occurrences of the non-deterministic choice by the delayed choice, as defined in Table 6.

Table 6. The operator D for removing non-deterministic choices ($a \in A \cup \{\tau\}$)

DE1 $D(\varepsilon) = \varepsilon$
DE2 $D(\delta) = \delta$
DE3 $D(x + y) = D(x) \mp D(y)$
DE4 $D(ay) = a \cdot D(y)$

Now, a mixed linear time/branching time verification consists of proving

$$\tau_{I_1} \circ D \circ \tau_{I_2}(S) = R$$

The set I_2 contains all atomic actions which induce only irrelevant choices, while the choices between actions from I_1 should remain after abstraction.

We will illustrate this with parts of the verification of a *leader election protocol*. We call this protocol the *Paint Ball protocol*, because it is a formalization of the popular Paint Ball game, in which people fight each other by shooting paint balls.

Suppose that entities E_i ($i \in ID$, ID is the set of identifications $|ID| > 1$) have to elect a leader amongst themselves non-deterministically. Every entity can communicate synchronously with every other entity. Initially all entities are equal. We have the following quite simple requirements specification.

$$R = \tau \cdot \sum_{i \in ID} \tau \cdot leader(i)$$

where $leader(i)$ denotes that entity i has become leader. Notice that we prepend a silent step τ to represent some initial internal activity.

The Paint Ball protocol is specified as the parallel composition ($\|$, [5]) of all entities. The encapsulation operator ∂_H is applied to enforce successful communications only. It renames all atoms from the set H into δ.

$$S = \partial_H \left(\|_{i \in ID} \; E_i^{ID-\{i\}} \right)$$

Each entity E_i^V is indexed with a set V. This set contains all other entities that have not yet been defeated by i. If this set is empty, it means that i has defeated all other participants and that i will become the leader (L_i). If the set is not empty, a choice is made between shooting a paint ball at one of the remaining participants (s_{ij}), or receiving a paint ball (r_{ji}) and entering the failed state (F_i). If all but one of the entities have yielded, the leader informs all failed entities that the elections are finished ($sready_{ij}$) and finally executes action $leader(i)$.

$$\begin{array}{lll} E_i^{\emptyset} & = L_i & \\ E_i^V & = \sum_{j \in V} \left(s_{ij} \cdot E_i^{V-\{j\}} + r_{ji} \cdot F_i \right) & (V \neq \emptyset) \\ L_i & = \left(\|_{j \in ID-\{i\}} \; sready_{ij} \right) \cdot leader(i) & \\ F_i & = \sum_{j \in ID-\{i\}} \left(r_{ji} \cdot F_i + rready_{ji} \right) & \end{array}$$

We have the obvious communication function $(r_{ij}|s_{ij} = c_{ij}, rready_{ij}|sready_{ij} = cready_{ij})$ and encapsulation set $H = \{r_{ij}, rready_{i,j}, s_{ij}, sready_{i,j} | i, j \in ID\}$. Now let $I = \{c_{ij}, cready_{i,j} | i, j \in ID\}$ and consider $\tau_I(S)$. After several calculations we obtain a reduced specification such that $\tau \cdot \tau_I(S) = P^{ID}$.

$$\begin{array}{lll} P^V & = \tau \cdot \sum_{i \in V} \tau \cdot P^{V-\{i\}} & (|V| > 1) \\ P^{\{i\}} & = \tau \cdot leader(i) & \end{array}$$

The specification of P shows that during the execution of the protocol some internal choices are made, which denote that some entity i is removed from the list of candidates. This continues until one candidate remains. According to our requirements specification we are not interested in these implementation details. Using our proposed strategy we calculate (for $I_2 = \{c_{ij} | i, j \in ID\}$)

$$D \circ \tau_{I_2}(S) = \tau \cdot \sum_{i \in ID} \left(\left(\|_{j \in ID-\{i\}} \; cready_{ij} \right) \cdot leader(i) \right)$$

And if we set $I_1 = \{cready_{ij} | i, j \in ID\}$ then we get the desired equality.

$$\tau_{I_1} \circ D \circ \tau_{I_2}(S) = \tau \cdot \sum_{i \in ID} \tau \cdot leader(i) = R$$

5 Conclusion

We have defined the delayed choice operator in process algebra with abstraction. Using this operator we can express linear time specifications in a branching time setting. We have shown two applications of this operator, namely scenario specification and requirements reduction. A sound and complete axiomatization with respect to branching bisimulation was obtained.

References

1. L. Aceto, B. Bloom, and F.W. Vaandrager. Turning SOS rules into equations. *Information and Computation*, 111(1):1–52, 1994.
2. J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fund. Inf.*, IX(2):127–168, 1986.
3. J.C.M. Baeten and S. Mauw. Delayed choice: an operator for joining Message Sequence Charts. In D. Hogrefe and S. Leue, editors, *Formal Description Techniques, VII*, pages 340–354. Chapman & Hall, 1995.
4. J.C.M. Baeten and C. Verhoef. *Concrete process algebra*, pages 149–268. Handbook of logic in computer science (Vol 4, Semantic modelling), eds. S. Abramsky, Dov. M. Gabbay and T.S.E. Maibaum. Clarendon press, Oxford, 1995.
5. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.
6. J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information & Control*, 60:109–137, 1984.
7. B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced: preliminary report. In *Proc. 15th ACM symposium on Principles of Programming Languages*, pages 229–239. San Diego, California, 1988.
8. S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.
9. P. D'Argenio. τ-angelic choice for process algebra. Technical report, LIFIA, Dpto. de Informàtica, Fac. Cs. Exactas, UNLP, 1994.
10. P. D'Argenio and S. Mauw. Delayed choice for process algebra with abstraction. Report, Department of Computer Science, Eindhoven University of Technology, 1995. To appear.
11. R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, 1989.
12. J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100:202–260, 1992.
13. S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering. Unpublished manuscript, 1980.
14. J.W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 1–116. Oxford University Press, 1992.
15. C.P.J. Koymans and J.L.M. Vrancken. Extending process algebra with the empty process. Report LGPS 1, Dept. of Philosophy, University of Utrecht, 1985.
16. D.M.R. Park. Concurrency and automata on infinite sequence. In P. Deussen, editor, *Proc. 5th. GI Conference*, pages 167–183. LNCS 104, Springer-Verlag, 1981.
17. G.D. Plotkin. A structural approach to operational semantics. Report DAIMI-FN-19, Computer Science Department, University of Århus, 1981.
18. C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. In B. Jonsson and J. Parrow, editors, *Proc. CONCUR '94*, pages 433–448. Uppsala, Springer Verlag, 1994. LNCS 836.
19. C. Verhoef. A general conservative extension theorem in process algebra. In E.-R. Olderog, editor, *Proc. PROCOMET'94, IFIP 2 Working Conference*, pages 149–168. San Miniato, North-Holland, 1994.

This article was processed using the LaTeX macro package with LLNCS style