Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.F. Groote, A. Ponse

Process algebra with guards
Combining Hoare logic with process algebra

Computer Science/Department of Software Technology Report CS-R9069 December

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Process Algebra with Guards

Combining Hoare Logic with Process Algebra

Jan Friso Groote
Alban Ponse
Department of Software Technology, CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
e-mail: jfg@cwi.nl - alban@cwi.nl

Abstract

We extend process algebra with guards, comparable to the guards in guarded commands or conditions in common programming constructs such as 'if – then – else – fi' and 'while – do – od'.

The extended language is provided with an operational semantics based on transitions between pairs of a process and a (data-)state. The data-states are given by a data environment that also defines in which data-states guards hold and how actions (non-deterministically) transform these states. The operational semantics is studied modulo strong bisimulation equivalence. For basic process algebra (without operators for parallelism) we present a small axiom system that is complete with respect to a general class of data environments. Given a particular data environment $\mathcal{S}$ we add three axioms to this system, which is then again complete, provided weakest preconditions are expressible and $\mathcal{S}$ is sufficiently deterministic.

Then we study process algebra with parallelism and guards. A two phase-calculus is provided that makes it possible to prove identities between parallel processes. Also this calculus is complete. In the last section we show that partial correctness formulas can easily be expressed in this setting. We use process algebra with guards to prove the soundness of a Hoare logic for linear processes by translating proofs in Hoare logic into proofs in process algebra.

Key Words & Phrases: process algebra, Hoare logic, guards, structural operational semantics, bisimulation, completeness, soundness, partial correctness, conditionals.
1985 Mathematics Subject Classification: 68Q55, 68Q60.
1982 CR Categories: D2.4, D.3.1, F.3.1, F.3.2.
Note: The first author is supported under ESPRIT Basic Research Action no. 3006 (CONCUR) and both authors are supported by the European Communities under RACE project no. 1046 (SPECS). This document does not necessarily reflect the view of the SPECS consortium.

1 Introduction

Hoare logic has been introduced in 1969 as a proof system for the correctness of programs [Hoa69]. Since then it has been applied to many problems, and it has been thoroughly studied (see [Apt81, Apt84] for an overview). In Hoare logic a program is considered to be a state transformer; the initial state is transformed to a final state. The correctness of a program is expressed by pre- and post-conditions.

More recently processes, i.e. the behaviour of systems, have attracted attention. This has led to several process calculi (CCS [Mil80, Mil89], CSP [Hoa85], ACP [BK84a, BW90] and MEIJE [AB84]). In these calculi correctness is often expressed by equations saying that a specification and an implementation are equivalent in some sense. These equivalences are mainly based on observations: two processes are equal if some observer cannot distinguish between the two. A classification of process equivalences has been described in [Gla90].

It seems a natural and useful question how Hoare logic and process algebra can be integrated. In this paper we provide an answer in two steps. First we extend process algebra with *guards*. Depending on the state, a guard can either be transparent such that it can be passed, or it can block and prevent subsequent processes from being executed. Typical for our approach is that a guard *itself* represents a process. With this construct we can easily express the guarded commands of DIJKSTRA [Dij76] and the guards occurring in several languages such as LOTOS [ISO87] and CRL [Ss90]. A nice property of the guards in our framework is that they constitute a Boolean algebra.

Using guards a partial correctness formula

$$\{\alpha\}\, p\, \{\beta\}$$

with α, β guards and p representing some process can be expressed by the algebraic equation

$$\alpha p = \alpha p \beta$$

saying that if process p starts in a state where the guard α holds, then it follows that the guard β holds when p terminates. Such equations modelling partial correctness formulas were first given by MANES and ARBIB [MA86].

We provide process algebra with guards with an operational semantics involving state transformations. This semantics is based on transitions between configurations (p, s) where p is a process expression and s is the 'state'. To avoid confusion between the 'state' and configuration (also often called state) we will consequently use the term *data-state*. We assume that data-states are given by some *data environment* that also prescribes in which data-states guards hold and how atomic actions (non-deterministically) transform data-states.

We consider the processes modulo strong bisimulation equivalence [Mil80, Par81] and we come up with several axiomatisations. In the case of Basic Process Algebra (BPA) with the standard operators $+$ (choice) and $\cdot$ (sequential composition), termination constants and guards we present two axiom systems, BPA^4_G and $\mathrm{BPA}_G(\mathcal{S})$. The system BPA^4_G is complete for finite processes with respect to a general class of data environments. It contains three simple and one somewhat more involved axiom besides the nine that are standard for BPA with termination constants. BPA^4_G enables us to derive general facts about processes with guards that do not depend on a particular data environment.

The axiom system $\mathrm{BPA}_G(\mathcal{S})$ applies when one wants to prove equivalences between processes if a data environment $\mathcal{S}$ has been determined. This axiom system is defined only if *weakest preconditions* are expressible and $\mathcal{S}$ is *sufficiently deterministic*. It contains the axioms of BPA^4_G together with three new axiom schemes that depend on $\mathcal{S}$. We use $\mathrm{BPA}_G(\mathcal{S})$ to prove the correctness of a well-known small program in a completely algebraic manner.

Parallel operators fit easily in the process algebra framework. In Hoare logic, however, parallelism turns out to be rather intricate; proof rules for parallel operators are often substantial [OG76, Lam80, Sti88]. In our setup we cannot completely avoid the difficulties caused

by parallel operators in Hoare logic, but we can deal with them in a simple algebraic way. We introduce a new set of axioms, called ACP_G that enables us to rewrite every process term to a term without parallel operators. Then using BPA_G^4 or $\mathrm{BPA}_G(\mathcal{S})$ we can verify the equivalences we are interested in. We apply these techniques to an example.

In the last section of this paper we show that process algebra with guards can indeed be used to verify partial correctness formulas, even in a setting with parallelism. Furthermore we apply $\mathrm{BPA}_G(\mathcal{S})$ to show soundness of a Hoare logic for process algebra with linear processes [Pon89, PV89]. The proof uses a canonical translation of proofs in Hoare logic into proofs in process algebra.

Acknowledgement

We thank Jos Baeten, Jan Bergstra, Frank de Boer, Tony Hoare, Catuscia Palamidessi, Frits Vaandrager and Fer-Jan de Vries for their constructive and helpful comments.

Contents

2 Basic Process Algebra with δ and ϵ

In this section we introduce the basic theory $\mathrm{BPA}_{\delta\epsilon}$ (Basic Process Algebra with δ and ϵ) that forms the basis for "Process algebra with guards". Contrary to the traditional approach, we provide $\mathrm{BPA}_{\delta\epsilon}$ with an operational semantics that is based on data-state transformations. The operational meaning of a process term is defined by a transition system, where the states of the transition system are *configurations*, i.e. pairs of a process term and a data-state. We study this behaviour in the setting of bisimulation semantics. The section is concluded with a short treatment of processes defined by recursive equations.

2.1 Signature and axioms

We start off in the well developed setting of $\mathrm{BPA}_{\delta\epsilon}$ which is an extension of BPA (Basic Process Algebra, see for instance [BK84b]) with two special constants δ and ϵ. The constant δ represents the absence of the possibility to perform any activity and is called *inaction*. The constant ϵ denotes a process that can do nothing but terminate and is called the *empty process* [KV85, Vra86, BG87]. The theory $\mathrm{BPA}_{\delta\epsilon}$ is parameterised with a set A of atomic actions with typical elements $a, b, \ldots$ These atomic actions represent the basic activities that processes can perform, such as reading input, incrementing counters and so forth. For each atomic action a the signature of $\mathrm{BPA}_{\delta\epsilon}$, denoted as $\Sigma(\mathrm{BPA}_{\delta\epsilon})$, contains an identically named constant a. We also have the binary infix operators $+$ (alternative composition) and $\cdot$ (sequential composition) available. We summarise the signature $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ in table 1.

constants:	a	for any atomic action $a \in A$
	δ	inaction ($\delta \notin A$)
	ϵ	empty process ($\epsilon \notin A$)
binary operators :	$+$	alternative composition (sum)
	$\cdot$	sequential composition (product)

Table 1: The signature $\Sigma(\mathrm{BPA}_{\delta\epsilon})$

Throughout this text let $V = \{x, y, z, \ldots\}$ be a set of variables. Process terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$, or shortly terms, are constructed from the variables in V and the elements of $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. We use letters $t, t', \ldots$ to denote terms. Any term not containing variables is called *closed*. We let $p, q, \ldots$ range over closed terms. In terms we generally omit the function symbol $\cdot$ and, like in regular algebra, we adopt the convention that $\cdot$ binds stronger than $+$.

We define the *depth* of a closed term over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ as the maximal number of consecutive atomic actions that can be performed. In the sequel it will play a role as a criterion for induction in proofs.

Definition 2.1.1 The *depth* of a closed term p over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$, written as $|p|$, is some element of $\mathbb{N}$, defined inductively as follows ($a \in A$):

$$
\begin{aligned}
&|\delta| \stackrel{\text{def}}{=} |\epsilon| \stackrel{\text{def}}{=} 0,\\
&|a| \stackrel{\text{def}}{=} 1,\\
&|pq| \stackrel{\text{def}}{=} |p| + |q|,\\
&|p + q| \stackrel{\text{def}}{=} max(|p|, |q|).
\end{aligned}
$$

□

The axioms presented in table 2 describe the basic identities between terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. The operator $+$ is commutative, associative and idempotent. The operator $\cdot$ is associative and right-distributes over $+$. Note that left distributivity of $\cdot$ over $+$ is absent. Furthermore δ behaves as the neutral element for $+$, and ϵ as the neutral element for $\cdot$. We leave out the brackets in terms whenever this is allowed by associativity. The symbol $\equiv$ is used to denote syntactic equivalence (modulo associativity) between terms.

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

Table 2: The axioms of $\mathrm{BPA}_{\delta\epsilon}$

In the sequel results are often proved by reasoning on the structure of process terms. In order to give some general definitions, let the symbol Σ range over all signatures we consider in this paper. Any such a signature Σ always extends the signature $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ defined above. Terms over Σ are constructed in the usual way and may contain variables from V. For any term t over Σ, $Var(t)$ denotes the set of variables occurring in the term t and $\Gamma(\Sigma)$ is used for axiom systems over the signature Σ.

We introduce the following elementary notions.

Definition 2.1.2 Let t_1, t_2 be terms over Σ. We call t_1 a *syntactic summand* of t_2, notation $t_1 \sqsubseteq t_2$, iff

1. $t_1 \not\equiv t + t'$ for any t, t' over Σ, and
2. $t_1 \equiv t_2$, or there are t, t' over Σ such that $t_2 \equiv t + t'$ and $t_1 \equiv t$ or $t_1 \equiv t'$.

□

So eg. $x(y + y) + z + z$ has $x(y + y)$ and z as its only syntactic summands and $(x + y)z$ has no other syntactic summand than itself.

Definition 2.1.3 Two process terms t and t' over Σ are *provably equal* in $\Gamma(\Sigma)$, notation

$$\Gamma(\Sigma) \vdash t = t',$$

iff there exists a proof of $t = t'$ using the axioms of $\Gamma(\Sigma)$, and the usual inference rules for equality (stating that '=' is a congruence relation). □

In proofs we sometimes write $t = t'$ instead of $\Gamma(\Sigma) \vdash t = t'$ and $t \neq t'$ instead of $\Gamma(\Sigma) \nvdash t = t'$.

Example 2.1.4 We illustrate the use of the $\mathrm{BPA}_{\delta\epsilon}$-axioms by showing that if for two terms t and t' over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ we have $\mathrm{BPA}_{\delta\epsilon} \vdash t + t' = \delta$, then also $\mathrm{BPA}_{\delta\epsilon} \vdash t = \delta$:

$$\begin{aligned} \mathrm{BPA}_{\delta\epsilon} \vdash t &= t + \delta \\ &= t + t + t' \\ &= t + t' \\ &= \delta. \end{aligned}$$

(End example.)

For notational convenience we introduce the notation $\subseteq$, called *summand inclusion*: for any two terms t, t' over Σ we write $t \subseteq t'$ for $t + t' = t'$ and $t' \supseteq t$ for $t' = t + t'$. In both cases we say that t is a *summand* of t'.

2.2 Operational semantics and soundness

In process algebra process terms are often related to *(labelled) transition systems*, modelling their possible behaviour.

Definition 2.2.1 A *labelled transition system* $\mathcal{A}$ is a tuple $\langle S_{\mathcal{A}}, A_{\mathcal{A}}, \longrightarrow_{\mathcal{A}}, s_{\mathcal{A}} \rangle$, where

- $S_{\mathcal{A}}$ is a set of *states*,
- $A_{\mathcal{A}}$ is a set of *labels*,
- $\longrightarrow_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times A_{\mathcal{A}} \times S_{\mathcal{A}}$ is the *transition relation*, and
- $s_{\mathcal{A}} \in S_{\mathcal{A}}$ is the *initial state*.

Elements $(s, a, t) \in \longrightarrow_{\mathcal{A}}$ are generally written as $s \xrightarrow{a} t$. □

In this paper we let $\mathrm{BPA}_{\delta\epsilon}$-processes operate on *data-states*. We adopt an abstract view and assume that data-states are given by a set S. Atomic actions are considered as non-deterministic data-state *transformers*. This is modelled by a function *effect* that, given some atomic action a and a data-state s, returns the data-states which may result from the execution of a in s (see also [BKT85, BB88]; in [BB88] the *state operator* is introduced which provides an alternative way to handle processes operating on data-states). We demand that the function *effect* never returns the empty set, ensuring that an atomic action can always be executed. We will use the guards introduced in the next section to prevent actions from happening in certain data-states.

Definition 2.2.2 A *data environment* $\mathcal{S} = \langle S, \mathit{effect} \rangle$ over a set A of atomic actions is specified by

- a non-empty set S of *data-states*, with typical elements $s, s', s'', ...,$
- a function $\mathit{effect} : A \times S \to (\mathcal{P}(S) - \{\emptyset\})$.

Here $\mathcal{P}(S)$ is the power set of S. □

Let $\mathcal{S} = \langle S, \mathit{effect} \rangle$ be some data environment over A. We give an operational semantics in the style of PLOTKIN [Plo81]. The behaviour of a process p with some initial data-state $s \in S$ starts in the *configuration* (p, s).

Definition 2.2.3 Let Σ be some signature and S a set of data-states. A *configuration* (p, s) *over* (Σ, S) is a pair containing a closed term p over Σ and a data-state $s \in S$. The set of all configurations over (Σ, S) is denoted by $C(\Sigma, S)$. □

ϵ	$(\epsilon, s) \xrightarrow{\surd} (\delta, s)$	
$a \in A$	$(a, s) \xrightarrow{a} (\epsilon, s')$ if $s' \in \mathit{effect}(a, s)$	
$+$	$\dfrac{(x, s) \xrightarrow{a} (x', s')}{(x + y, s) \xrightarrow{a} (x', s')}$	$\dfrac{(y, s) \xrightarrow{a} (y', s')}{(x + y, s) \xrightarrow{a} (y', s')}$
$\cdot$	$\dfrac{(x, s) \xrightarrow{a} (x', s')}{(xy, s) \xrightarrow{a} (x'y, s')}$ $a \neq \surd$	$\dfrac{(x, s) \xrightarrow{\surd} (x', s') \quad (y, s) \xrightarrow{a} (y', s'')}{(xy, s) \xrightarrow{a} (y', s'')}$

Table 3: Transition rules for $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ $(a \in A_{\surd})$

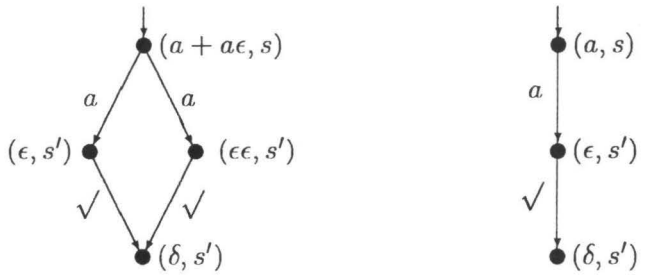
Let $\surd \notin A$ be a special symbol which we use to represent successful termination, and $A_{\surd} \stackrel{\text{def}}{=} A \cup \{\surd\}$. The rules in table 3, where the label a ranges over $A_{\surd}$, determine the transition relation $\longrightarrow_{\Sigma(\mathrm{BPA}_{\delta\epsilon}),\mathcal{S}}$ that contains exactly all derivable transitions between the configurations over $(\Sigma(\mathrm{BPA}_{\delta\epsilon}), S)$. The idea is that for $a \in A$, the transition $(p, s) \xrightarrow{a} (p', s')$ expresses that by executing a, the process p in data-state s can evolve into p' in data-state s'. In this case we have $s' \in \mathit{effect}(a, s)$ and the configuration (p', s') represents what remains to be executed. The transition $(p, s) \xrightarrow{\surd} (p', s')$ expresses that the process p in data-state s can terminate successfully. The empty process ϵ can terminate successfully in any data-state s, which is denoted by the transition $(\epsilon, s) \xrightarrow{\surd} (\delta, s)$ in table 3. The configuration (δ, s) has

no outgoing transitions, which expresses that no further activity is possible ('inaction' or 'deadlock' after successful termination).

In the case of $\mathrm{BPA}_{\delta\epsilon}$ we define

$$\mathcal{A}(p,s) \stackrel{\text{def}}{=} \langle C(\Sigma(\mathrm{BPA}_{\delta\epsilon}),S)\ ,\ A_{\surd}\ ,\ \longrightarrow_{\Sigma(\mathrm{BPA}_{\delta\epsilon}),\mathcal{S}}\ ,\ (p,s)\rangle.$$

As an example consider the following (partially depicted) transition systems $\mathcal{A}(a+a\epsilon,s)$ and $\mathcal{A}(a,s)$, where the initial states are marked with a little arrow and $\mathit{effect}(a,s)=\{s'\}$.



Observe that the transition system $\mathcal{A}(a+a\epsilon,s)$ is shaped as two transition systems for $\mathcal{A}(a,s)$. With respect to *operational* behaviour it does not matter whether the a-side or the $a\epsilon$-side is executed. Therefore we would like to consider both transition systems as equivalent. This can be achieved by identifying *bisimilar* configurations (see [Mil80, Par81]), as bisimilarity is the coarsest equivalence that respects the operational characteristics of a transition system [Vaa89]. Following the traditional approach in semantics based on data-state transformations, processes with different data-states in their configurations are not considered as equivalent (See eg. [Man74]). Therefore we adapt the standard notion of bisimilarity in the following way.

Definition 2.2.4 Let Σ be a signature, $\mathcal{S}$ a data environment with data-state space S and $\longrightarrow_{\Sigma,\mathcal{S}}$ a transition relation over $C(\Sigma,S)$.

- A binary relation $R \subseteq C(\Sigma,S)\times C(\Sigma,S)$ is an *$\mathcal{S}$-bisimulation* iff R satisfies the transfer property, i.e. for all $(p,s),(q,s)\in C(\Sigma,S)$ with $(p,s)R(q,s)$:

  1. whenever $(p,s)\stackrel{a}{\longrightarrow}_{\Sigma,\mathcal{S}}(p',s')$ for some a and (p',s'), then, for some q', also $(q,s)\stackrel{a}{\longrightarrow}_{\Sigma,\mathcal{S}}(q',s')$ and $(p',s')R(q',s')$,
  2. conversely, whenever $(q,s)\stackrel{a}{\longrightarrow}_{\Sigma,\mathcal{S}}(q',s')$ for some a and (q',s'), then, for some p', also $(p,s)\stackrel{a}{\longrightarrow}_{\Sigma,\mathcal{S}}(p',s')$ and $(p',s')R(q',s')$.

- A configuration $(p,s)\in C(\Sigma,S)$ is *$\mathcal{S}$-bisimilar* with a configuration $(q,s')\in C(\Sigma,S)$, notation $(p,s)\leftrightarrows_{\mathcal{S}}(q,s')$, iff $s=s'$ and there is an $\mathcal{S}$-bisimulation containing the pair $((p,s),(q,s'))$ (note the equality of the data-states!).

- A transition system $\mathcal{A}(p,s)=\langle C(\Sigma,S),A_{\surd},\longrightarrow_{\Sigma,\mathcal{S}},(p,s)\rangle$ is *$\mathcal{S}$-bisimilar* with a transition system $\mathcal{A}(q,s')=\langle C(\Sigma,S),A_{\surd},\longrightarrow_{\Sigma,\mathcal{S}},(q,s')\rangle$, notation $\mathcal{A}(p,s)\leftrightarrows_{\mathcal{S}}\mathcal{A}(q,s')$, iff $(p,s)\leftrightarrows_{\mathcal{S}}(q,s')$.

- Two closed terms p, q over Σ are *$\mathcal{S}$-bisimilar*, notation $p \leftrightarrows_{\mathcal{S}} q$, iff $\mathcal{A}(p, s) \leftrightarrows_{\mathcal{S}} \mathcal{A}(q, s)$ for all $s \in S$.

□

We introduced the symbol $\leftrightarrows_{\mathcal{S}}$ instead of the more consistent symbol $\leftrightarrows_{\Sigma,\mathcal{S}}$ to avoid lengthy notation. We take care that Σ is known from the context when we use $\leftrightarrows_{\mathcal{S}}$. Note that the symbol $\leftrightarrows_{\mathcal{S}}$ is also overloaded in another way. It denotes either a relation between configurations, between transition systems or between closed terms.

For any data environment $\mathcal{S}$ it follows in the standard way that the relation $\leftrightarrows_{\mathcal{S}}$ between closed terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ is a congruence relation.

Lemma 2.2.5 *For any data environment $\mathcal{S}$ the relation $\leftrightarrows_{\mathcal{S}}$ between closed terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ is a congruence with respect to the operators of $\Sigma(\mathrm{BPA}_{\delta\epsilon})$.* □

Moreover, it is not hard to prove that $\mathrm{BPA}_{\delta\epsilon}$ is a *sound* axiom system with respect to $\mathcal{S}$-bisimulation equivalence for any data environment $\mathcal{S}$.

Theorem 2.2.6 (Soundness) *Let p, q be closed terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. If $\mathrm{BPA}_{\delta\epsilon} \vdash p = q$, then $p \leftrightarrows_{\mathcal{S}} q$ for any data environment $\mathcal{S}$.*

Proof. Standard (for an idea, see the proof of theorem 3.2.5). □

2.3 Completeness

We show that the axiom system $\mathrm{BPA}_{\delta\epsilon}$ completely axiomatises $\mathcal{S}$-bisimilarity with respect to the closed terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. So, given some data environment $\mathcal{S}$, we show that for any two closed terms p, q over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$

$$p \leftrightarrows_{\mathcal{S}} q \implies \mathrm{BPA}_{\delta\epsilon} \vdash p = q.$$

The technique we use in proving this result is based on the following notion of 'normal form'.

Definition 2.3.1 A closed term p is in *prefix normal form over* $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ iff

$$p ::= \delta \mid \epsilon \mid ap \mid p + p$$

where a ranges over A. □

By induction on the structure of closed terms, the following lemma is not difficult to prove.

Lemma 2.3.2 *For any closed term p over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ there is a term q in prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ such that $\mathrm{BPA}_{\delta\epsilon} \vdash p = q$.* □

We can derive information about the actual structure of a term in prefix normal form from its operational behaviour. We will use this to conclude from the $\mathcal{S}$-bisimilarity of terms in prefix normal form their provable equality.

Lemma 2.3.3 *Let $\mathcal{S} = \langle S, effect \rangle$. For any term p in prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ the following properties hold:*

1. *If $\exists s \in S$ such that $(p, s) \xrightarrow{\surd} (r, s')$, then $\epsilon \sqsubseteq p$,*
2. *If $\exists s \in S$ such that $(p, s) \xrightarrow{a} (r, s')$ $(a \in A)$, then there is a term r' in prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ such that $ar' \sqsubseteq p$ and $\epsilon r' \equiv r$.*

Proof. We apply induction on the structure of p. For the cases $p \equiv \delta$ and $p \equiv \epsilon$ both results follow easily by inspection of the transition rules.

$p \equiv bq$ (with $b \in A$). In this case $(bq, s) \xrightarrow{\surd} (r, s')$ is not derivable. Furthermore $(bq, s) \xrightarrow{a} (r, s')$ can only be derived if $b \equiv a$ and $r \equiv \epsilon q$.

$p \equiv q + q'$. In this case $(q + q', s) \xrightarrow{b} (r, s')$ for some $b \in A_{\surd}$ can only be derived from a transition $(q, s) \xrightarrow{b} (r, s')$ or a transition $(q', s) \xrightarrow{b} (r, s')$. In both cases we can apply the induction hypothesis. By definition of the relation $\sqsubseteq$ (see 2.1.2) we are done. □

In the next lemma we can now present an intermediate result which not only implies completeness, but is also used later in the paper.

Lemma 2.3.4 *Let p_1, p_2 be terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ in prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$, and $\mathcal{S} = \langle S, effect \rangle$ a data environment. Then*

$$\exists s \in S \, ((p_1, s) \leftrightarrows_{\mathcal{S}} (p_2, s)) \implies \mathrm{BPA}_{\delta\epsilon} \vdash p_1 = p_2.$$

Proof. We show that $p_1 = p_2$ by proving that any syntactic summand of p_1 is provably equal to a syntactic summand of p_2 and vice versa. We apply induction on $|p_1| + |p_2|$. The case $|p_1| + |p_2| = 0$ is trivial, so assume $|p_1| + |p_2| > 0$. By symmetry it suffices to show that if $t \sqsubseteq p_1$ for some term t, then we can find a term t' such that $\mathrm{BPA}_{\delta\epsilon} \vdash t = t'$ and $t' \sqsubseteq p_2$. Let $s \in S$ satisfy the condition of the lemma.

Suppose $ar \sqsubseteq p_1$. For any $s' \in effect(a, s)$ we have $(p_1, s) \xrightarrow{a} (\epsilon r, s')$. By assumption $(p_2, s) \xrightarrow{a} (r', s')$ for some term r', satisfying $(\epsilon r, s') \leftrightarrows_{\mathcal{S}} (r', s')$. By lemma 2.3.3 there is a term r'' in prefix normal form such that $\epsilon r'' \equiv r'$ and $ar'' \sqsubseteq p'$. So $(\epsilon r, s') \leftrightarrows_{\mathcal{S}} (\epsilon r'', s')$, and thus $(r, s') \leftrightarrows_{\mathcal{S}} (r'', s')$. By the induction hypothesis $r = r''$, and hence $ar = ar''$.

In case $\epsilon \sqsubseteq p_1$, we can show in the same way that $\epsilon \sqsubseteq p_2$. □

This last result immediately implies that $\mathrm{BPA}_{\delta\epsilon}$ completely axiomatises the relation $\leftrightarrows_{\mathcal{S}}$ for any data environment $\mathcal{S}$.

Theorem 2.3.5 (Completeness) *Let r_1, r_2 be closed terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ and $\mathcal{S}$ be some data environment. If $r_1 \leftrightarrows_{\mathcal{S}} r_2$, then $\mathrm{BPA}_{\delta\epsilon} \vdash r_1 = r_2$.*

Proof. By lemma 2.3.2 we can find terms p_1, p_2 in prefix normal form satisfying $p_i = r_i$ $(i = 1, 2)$, and hence by soundness $p_i \leftrightarrows_{\mathcal{S}} r_i$. So we have $p_1 \leftrightarrows_{\mathcal{S}} p_2$ and using the previous lemma 2.3.4 we derive $\mathrm{BPA}_{\delta\epsilon} \vdash r_1 = p_1 = p_2 = r_2$. □

Combining lemma 2.3.4 and the soundness theorem, we obtain the following corollary that says that data environments have no particular effect with respect to bisimulation semantics.

Corollary 2.3.6 *If for some particular data environment* $\mathcal{S}_0 = \langle S_0, \mathit{effect}\rangle$, *data-state* $s_0 \in S_0$ *and closed terms* p, q *over* $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ *we have that* $(p, s_0) \leftrightarrows_{\mathcal{S}_0} (q, s_0)$, *then for any data environment* $\mathcal{S}$ *also* $p \leftrightarrows_{\mathcal{S}} q$. □

Remark that if we allow the functions *effect* to return the empty set, this corollary is not true any more. However, soundness and completeness are still provable in this case.

2.4 Recursion

We extend our process language with a mechanism that enables us to specify infinite processes by recursive equations.

Definition 2.4.1 A *recursive specification* $E = \{x = t_x \,|\, x \in V_E\}$ *over* a signature Σ is a set of equations where V_E is a (possibly infinite) set of variables and t_x a term over Σ such that $Var(t_x) \subseteq V_E$. □

A *solution* of a recursive specification $E = \{x = t_x \mid x \in V_E\}$ is an interpretation of the variables in V_E as processes, such that the equations of E are satisfied. For instance the recursive specification $\{x = x\}$ has any process as a solution for x and $\{x = ax\}$ has the infinite process "a^ω" as a solution for x. We introduce the following syntactical restriction on recursive specifications.

Definition 2.4.2 Let t be a term over a signature Σ. An occurrence of a variable x in t is *guarded* iff t has a subterm of the form $a \cdot M$ with $a \in A \cup \{\delta\}$, and this x occurs in M. Let $E = \{x = t_x \,|\, x \in V_E\}$ be a recursive specification over Σ. We say that E is a *guarded* specification iff all occurrences of variables in the terms t_x are guarded. □

Now the signature Σ_{REC}, in which we are interested, is defined by:

Definition 2.4.3 The signature Σ_{REC} is obtained by extending Σ in the following way: for each guarded specification $E = \{x = t_x \,|\, x \in V_E\}$ over Σ a set of constants $\{<x \,|\, E> \,|\, x \in V_E\}$ is added, where the construct $<x \,|\, E>$ denotes the x-component of a solution of E. □

We introduce some more notations: let $E = \{x = t_x \,|\, x \in V_E\}$ be a guarded specification over Σ, and t some term over Σ_{REC}. Then $<t \,|\, E>$ denotes the term in which each occurrence of a variable $x \in V_E$ in t is replaced by $<x \,|\, E>$, e.g. $<aax \,|\, \{x = ax\}>$ denotes the term $aa<x \,|\, \{x = ax\}>$. If we assume that the variables in recursive specifications are chosen uniquely, there is no need to repeat E in each occurrence of $<x \,|\, E>$. Variables reserved in this way are called *formal variables* and denoted by capital letters. We adopt the convention that $<x \,|\, E>$ can be abbreviated by X once E is declared. As an example consider the guarded recursive specification $\{x = ax\}$: the closed term aaX abbreviates $aa<x \,|\, \{x = ax\}>$.

For the new Σ-constants of the form $<x \,|\, E>$ there are two axioms in table 4. In these axioms the letter E ranges over guarded specifications. The axiom REC states that the constant $<x \,|\, E>$ $(x \in V_E)$ is a solution for the x-component of E. The conditional axiom RSP (Recursive Specification Principle) expresses that E has at most one solution for each of its variables: whenever we can find processes p_x $(x \in V_E)$ satisfying the equations of E, then $p_x = <x \,|\, E>$. This axiom was first formulated in [BK86]. The format adopted here stems from [GV89].

$<x \mid E> = <t_x \mid E>$ if $x = t_x \in E$ and E guarded	REC
$\dfrac{E}{x = <x \mid E>}$ if $x \in V_E$ and E guarded	RSP

Table 4: Axioms for guarded recursive specifications

Example 2.4.4 Consider the guarded specifications $E = \{x = ax\}$ and $E' = \{y = ayb\}$ over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. We can show that $\mathrm{BPA}_{\delta\epsilon} + \mathrm{REC} + \mathrm{RSP} \vdash X = Y$ in the following way:

$$Xb \overset{\mathrm{REC}}{=} aXb \quad \overset{\mathrm{RSP}}{\Longrightarrow} \quad Xb = X, \tag{1}$$

and secondly

$$Xb \overset{\mathrm{REC}}{=} aXb \overset{(1)}{=} aXbb \quad \overset{\mathrm{RSP}}{\Longrightarrow} \quad Xb = Y.$$

Hence $\mathrm{BPA}_{\delta\epsilon} + \mathrm{REC} + \mathrm{RSP} \vdash X = Xb = Y$. *(End example.)*

In order to associate transition systems with processes defined by guarded specifications, we define in the case of $E = \{x = t_x \mid x \in V_E\}$ being a guarded recursive specification over some signature Σ the general transition rule in table 5. With respect to $\mathrm{BPA}_{\delta\epsilon}$, this rule

recursion $\dfrac{(<t_x \mid E>, s) \xrightarrow{a} (y, s')}{(<x \mid E>, s) \xrightarrow{a} (y, s')}$	if $x = t_x \in E$

Table 5: Transition rule for guarded recursive specifications ($a \in A_{\surd}$)

immediately implies the soundness of REC. We state without proof that $\mathrm{BPA}_{\delta\epsilon} + \mathrm{REC} + \mathrm{RSP}$ is sound (the interested reader is referred to [BW90]).

Theorem 2.4.5 (Soundness of REC and RSP) *Let p, q be closed terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})_{\mathrm{REC}}$. If $\mathrm{BPA}_{\delta\epsilon} + \mathrm{REC} + \mathrm{RSP} \vdash p = q$, then $p \leftrightarrows_{\mathcal{S}} q$ for any data environment $\mathcal{S}$.* □

Note that RSP is not valid in the case of *un*guarded recursion: the unguarded recursive specification $\{x = x\}$ would otherwise lead to provable equality between all terms over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$.

Remark 2.4.6 The terminology *guarded* recursive specification is established and therefore we respect it. The adjective "guarded" has nothing to do with the "guards" that form the main subject of this paper. □

3 Basic Process Algebra with guards

In this section we extend $\mathrm{BPA}_{\delta\epsilon}$ with *guards*. These guards are comparable to those in the guarded commands of DIJKSTRA [Dij76], or to the conditions in programming constructs as

if - then - else - fi and **while - do - od**. Typical for our approach is that a guard itself represents a process that, depending on its current data-state, either terminates successfully (so behaves as ϵ), or can do nothing (cf. δ). The process algebra operators $+$ and $\cdot$ are used to express Boolean operations on guards. In this section we study these guards in the setting of bisimulation semantics, and present an axiomatisation that is complete with respect to $\mathcal{S}$-bisimilarity in *all* data environments $\mathcal{S}$.

3.1 Guards

Let G_{at} be a non-empty set of *atomic guards* disjoint with a fixed set A of atomic actions, and also disjoint with $\{\delta, \epsilon\}$. We extend G_{at} to the set G of *basic* guards with typical elements $\phi, \psi, ...$, where basic guards are defined by the following syntax:

$$\phi ::= \delta \mid \epsilon \mid \neg\phi \mid \psi \in G_{at}.$$

In particular the special constants δ and ϵ are considered as basic guards: δ is the guard that always blocks, and ϵ is the guard that can always be passed. Furthermore $\neg$ is the negation operator on basic guards. Now the signature $\Sigma(\mathrm{BPA}_G)$ is defined by adding all elements of $G - \{\delta, \epsilon\}$ as constants to the signature of $\mathrm{BPA}_{\delta\epsilon}$. We summarise the signature $\Sigma(\mathrm{BPA}_G)$ in table 6. The *depth* of a closed term p over $\Sigma(\mathrm{BPA}_G)$ is defined by extending definition 2.1.1 with the inductive clause $|\phi| \stackrel{\text{def}}{=} 0$ for any basic guard $\phi \in \Sigma(\mathrm{BPA}_G)$.

constants:	a	for any atomic action $a \in A$
	ϕ	for any basic guard $\phi \in G$ (recall $\delta, \epsilon \in G$)
binary operators:	$+$	alternative composition (sum)
	$\cdot$	sequential composition (product)

Table 6: The signature $\Sigma(\mathrm{BPA}_G)$

The intended meaning of a basic guard ϕ in any data environment $\mathcal{S}$ is that it behaves as δ or ϵ, depending on the data-state it is evaluated in. Either ϕ holds in a data-state s, in which case (ϕ, s) behaves as (ϵ, s), or this is not the case and then (ϕ, s) behaves as (δ, s).

The axioms in table 7 describe the basic identities between terms over $\Sigma(\mathrm{BPA}_G)$. In this table ϕ ranges over G and $a \in A$. The axioms A1 – A9 are the ordinary $\mathrm{BPA}_{\delta\epsilon}$-axioms. The axioms G1 – G3 describe the expected identities between guards. G1 and G2 express that a basic guard always behaves dually to its negation: ϕ holds in a data-state s iff $\neg\phi$ does not and vice versa. The axiom G3 states that $+$ does not change the interpretation of a basic guard ϕ. It does not matter whether the choice is exercised before or after the evaluation of ϕ. Notice the $\mathrm{BPA}_{\delta\epsilon}$-derivability for the δ and ϵ-instances of G3.

The last new axiom G4 (recall the notation $\subseteq$, introduced at the end of section 2.1) is necessary for our completeness result. It can be explained as follows: the process $a(\phi x + \neg\phi y)$, where a is an atomic action, behaves either like ax or ay, depending on the data-state resulting from the execution of a. As a consequence the process $a(\phi x + \neg\phi y)$ should be a provable

summand of $ax+ay$. The a in this axiom may not be replaced by a larger process expression. If it is for instance replaced by the expression $a \cdot b$ then after a has happened, it is in general not clear whether ϕ or $\neg\phi$ will hold after b. Hence $ab(\phi x + \neg\phi y)$ need not be a summand of $abx + aby$. Note that the axiom G4 is not derivable from the first three 'guard'-axioms.

The axioms in table 7 constitute the axiom system BPA_G^4. The superscript 4 expresses that there are four axioms referring to guards. We will not always consider all of these. In particular the system BPA_G^3, containing all BPA_G^4-axioms except G4 will play a role in this paper.

$x+y=y+x$	A1	$\phi \cdot \neg\phi = \delta$	G1
$x+(y+z)=(x+y)+z$	A2	$\phi + \neg\phi = \epsilon$	G2
$x+x=x$	A3	$\phi(x+y)=\phi x+\phi y$	G3
$(x+y)z=xz+yz$	A4		
$(xy)z=x(yz)$	A5		
$x+\delta=x$	A6		
$\delta x=\delta$	A7	$a(\phi x+\neg\phi y) \subseteq ax+ay$	G4
$\epsilon x=x$	A8		
$x\epsilon=x$	A9		

Table 7: The axioms of BPA_G^4 where $\phi \in G$ and $a \in A$

We now give a result expressing some useful properties of basic guards, in which the axiom G4 is not used. Note clause (v), which expresses that the sequential composition is commutative for basic guards.

Lemma 3.1.1 *Let $\phi, \psi \in G$. The following identities are derivable in BPA_G^3:*

(i) $\neg\delta = \epsilon$,
(ii) $\neg\epsilon = \delta$,
(iii) $\neg\neg\phi = \phi$,
(iv) $\phi\psi\neg\phi = \delta$,
(v) $\phi\psi = \psi\phi$.

Proof. In the proofs of (i) and (ii) the axiom G3 is also not used.

$$\begin{array}{lllll} (i) & \neg\delta & = & \delta+\neg\delta \\ & & = & \epsilon. \end{array} \qquad \begin{array}{lllll} (ii) & \neg\epsilon & = & \epsilon\cdot\neg\epsilon \\ & & = & \delta. \end{array}$$

In the proof of (iv) we use $t+t'=\delta \Longrightarrow t=\delta$ (see example 2.1.4). In (v) we use $\neg\phi\psi\phi=\delta$, which is a direct consequence of (iii) and (iv).

$$\begin{array}{lllll} (iii) & \neg\neg\phi & = & (\phi+\neg\phi)\neg\neg\phi \\ & & = & \phi\neg\neg\phi+\delta \\ & & = & \phi\neg\neg\phi+\phi\neg\phi \\ & & = & \phi(\neg\neg\phi+\neg\phi) \\ & & = & \phi. \end{array} \qquad \begin{array}{lllll} (iv) & \delta & = & \phi\neg\phi \\ & & = & \phi(\psi+\neg\psi)\neg\phi \\ & & = & \phi(\psi\neg\phi+\neg\psi\neg\phi) \\ & & = & \phi\psi\neg\phi+\phi\neg\psi\neg\phi \\ & & \Longrightarrow & \phi\psi\neg\phi=\delta. \end{array} \qquad \begin{array}{lllll} (v) & \phi\psi & = & \phi\psi(\phi+\neg\phi) \\ & & = & \phi\psi\phi+\phi\psi\neg\phi \\ & & = & \phi\psi\phi+\neg\phi\psi\phi \\ & & = & (\phi+\neg\phi)\psi\phi \\ & & = & \psi\phi. \end{array}$$

□

Up till now we only defined 'atomic' and 'basic' guards. We use the general name *guards* for terms over $\Sigma(\mathrm{BPA}_G)$ that contain only basic guards. Let the symbols $\alpha, \beta, \ldots$ range over guards.

Definition 3.1.2 A *guard* α over $\Sigma(\mathrm{BPA}_G)$ has the following syntax

$$\alpha ::= \phi \mid \alpha + \alpha \mid \alpha \cdot \alpha$$

where $\phi \in G$. □

On guards the operators $+$ and $\cdot$ correspond to the Boolean operators $\vee$ and $\wedge$, respectively. Let $\phi, \psi \in G$, then the guard $\phi + \psi$ holds in a data-state s whenever ϕ or ψ holds in s. The guard $\phi\psi$ holds in s iff both ϕ and ψ hold in s. In order to have the Boolean operator $\neg$ on guards, we introduce the *abbreviations*

$$\begin{array}{ll} \neg(\alpha\beta) & \text{for} \quad \neg\alpha + \neg\beta, \\ \neg(\alpha + \beta) & \text{for} \quad \neg\alpha\neg\beta. \end{array}$$

It is not hard to prove that all identities on basic guards that are derivable in BPA^3_G (or BPA^4_G), are derivable in BPA^3_G (BPA^4_G, respectively) for *all* guards:

Theorem 3.1.3 *Let α be a guard over $\Sigma(\mathrm{BPA}_G)$, then the following identities are derivable in BPA^3_G (cf. G1 – G3):*

(i) $\alpha \cdot \neg\alpha = \delta$,
(ii) $\alpha + \neg\alpha = \epsilon$,
(iii) $\alpha(x + y) = \alpha \cdot x + \alpha \cdot y$.

The following identity is derivable in BPA^4_G (cf. G4):

(iv) $a(\alpha \cdot x + \neg\alpha \cdot y) \subseteq ax + ay$.

where $a \in A$. □

Moreover, restricting the signature $\Sigma(\mathrm{BPA}_G)$ to terms without atomic actions, the axiom system BPA^3_G constitutes a Boolean algebra. According to [Sio64], the following five equations form an equational basis for a Boolean algebra $(G_{at}, +, \cdot, \neg)$:

$B1.$ $xy = yx$
$B2.$ $x(y + z) = xy + xz$
$B3.$ $x + y\neg y = x$
$B4.$ $x(y + \neg y) = x$
$B5.$ $x + (y + \neg y) = y + \neg y$.

The only equation here that does not immediately follow from BPA^3_G is B5:

$$\begin{aligned} \alpha + (\beta + \neg\beta) &= \alpha + \epsilon \\ &= \alpha + (\alpha + \neg\alpha) \\ &= (\alpha + \alpha) + \neg\alpha \\ &= \alpha + \neg\alpha \\ &= \epsilon \\ &= \beta + \neg\beta. \end{aligned}$$

To illustrate the elegance of adopting guards as a special kind of processes, we make a short comparison with the algebraic approach advocated by HOARE cs. in [HHJ+87]. There, processes ('programs') have a syntax comparable to $\Sigma(\mathrm{BPA}_{\delta\epsilon})_{\mathrm{REC}}$ with *conditionals*, the latter being constructs of the form

$$P \lhd b \rhd Q$$

where P, Q are programs, and b is a Boolean expression that evaluates the current (execution) state. When b evaluates to *true*, P is executed and otherwise Q takes place. So this conditional represents the programming construct **if** b **then** P **else** Q **fi**. Assuming that atomic guards are the atomic propositions from which Boolean expressions are constructed (and hence guards as such represent all Boolean expressions), a conditional

$$p \lhd \alpha \rhd q$$

can be represented in $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ by the closed term

$$\alpha p + \neg\alpha q.$$

As a consequence we can do with about 12 simple algebraic laws, which should be compared with the about 30 laws for recursion-free programs presented in [HHJ+87]. Of course the advantage of 'conditionals' is that they refer to a well-known and established programming construct. Nevertheless it can be argued that for analytical purposes, guards as introduced here constitute a simpler and more fundamental approach.

3.2 Operational semantics and soundness

In this section we adapt the operational semantics of section 2 to accommodate guards. We extend the definition of a data environment by adding a predicate *test* that determines whether an atomic guard holds in some data-state.

Definition 3.2.1 A data environment $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ over a set A of atomic actions and a set G_{at} of atomic guards is specified by

- A non-empty set S of data-states,
- A function $\mathit{effect} : S \times A \to (\mathcal{P}(S) - \{\emptyset\})$,
- A predicate $\mathit{test} \subseteq G_{at} \times S$.

□

Whenever $\mathit{test}(\phi, s)$ holds, this denotes that in data-state s the atomic guard ϕ may be passed. In this case we say that ϕ *holds* in s. In order to interpret basic guards, we extend the predicate *test* in the obvious way.

Definition 3.2.2 Let $\langle S, \mathit{effect}, \mathit{test}\rangle$ be some data environment. We extend the domain of *test* to $G \times S$ as follows:

- for all $s \in S$: $\mathit{test}(\epsilon, s)$ holds and $\mathit{test}(\delta, s)$ does not hold,

- for all $s \in S$ and $\phi \in G$: $test(\neg\phi, s)$ holds iff $test(\phi, s)$ does not hold.

□

This gives us the means to easily define transition rules. Let $\mathcal{S} = \langle S, effect, test \rangle$ be a data environment. In table 8 we display a set of transition rules extending those of $\mathrm{BPA}_{\delta\epsilon}$ (see table 3). The difference with the transition rules introduced previously is that we now have added axioms for the elements of G (implying the old transition scheme $(\epsilon, s) \xrightarrow{\surd} (\delta, s)$). The transition rules in table 8 and the transition rule for guarded recursive specifications (see table 5) determine the transition relation $\longrightarrow_{\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}},\mathcal{S}}$ over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$. Let p be a closed term over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$. For any $s \in S$ the transition system $\mathcal{A}(p, s)$ is defined as

$$\mathcal{A}(p, s) \stackrel{\mathrm{def}}{=} \langle C(\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}, S)\ ,\ A_{\surd}\ ,\ \longrightarrow_{\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}},\mathcal{S}}\ ,\ (p, s)\rangle.$$

$a \in A \quad (a, s) \xrightarrow{a} (\epsilon, s')$ if $s' \in effect(a, s)$

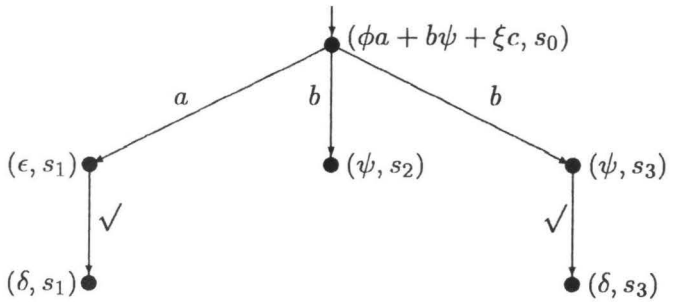
$\phi \in G \quad (\phi, s) \xrightarrow{\surd} (\delta, s)$ if $test(\phi, s)$

$$+ \quad \frac{(x, s) \xrightarrow{a} (x', s')}{(x + y, s) \xrightarrow{a} (x', s')} \qquad \frac{(y, s) \xrightarrow{a} (y', s')}{(x + y, s) \xrightarrow{a} (y', s')}$$

$$\cdot \quad \frac{(x, s) \xrightarrow{a} (x', s')}{(xy, s) \xrightarrow{a} (x'y, s')}\ a \neq \surd \qquad \frac{(x, s) \xrightarrow{\surd} (x', s') \quad (y, s) \xrightarrow{a} (y', s'')}{(xy, s) \xrightarrow{a} (y', s'')}$$

Table 8: Transition rules for $\Sigma(\mathrm{BPA}_G)$ $(a \in A_{\surd})$

Example 3.2.3 Consider the data environment $\langle \{s_0, s_1, s_2, s_3\}, effect, test \rangle$ and the following transition system $\mathcal{A}(\phi a + b\psi + \xi c, s_0)$:



The information about the function *effect* and the predicate *test* implicitly present in this transition system tells us that apparently

$$\begin{array}{l} \mathit{effect}(a, s_0) = \{s_1\} \text{ and } \mathit{effect}(b, s_0) = \{s_2, s_3\} \\ \mathit{test}(\phi, s_0), \mathit{test}(\psi, s_3) \text{ and not } \mathit{test}(\xi, s_0), \mathit{test}(\psi, s_2) \end{array}$$

and we have

$$(\phi a + b\psi + \xi c, s_0) \leftrightarrows_{\mathcal{S}} (a + b\psi, s_0)$$

in $\langle S, \mathit{effect}, \mathit{test}\rangle$. If eg. $\mathit{test}(\xi, s_1)$ holds, then

$$\phi a + b\psi + \xi c \not\leftrightarrows_{\mathcal{S}} a + b\psi$$

for only the left term has a $\xrightarrow{c}$ transition from s_1. *(End example.)*

Next we show that BPA^4_G + REC + RSP is a sound axiom system. This result says that the axioms of BPA^4_G are valid in all data environments as defined in this section, and hence express identities that are independent of a particular data environment.

Lemma 3.2.4 *For any data environment $\mathcal{S}$ the relation $\leftrightarrows_{\mathcal{S}}$ between closed terms over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ is a congruence with respect to the operators of $\Sigma(\mathrm{BPA}_G)$.*

Proof. Standard. □

Theorem 3.2.5 (Soundness) *Let p, q be closed terms over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$. If BPA^4_G+REC+ RSP $\vdash p = q$, then $p \leftrightarrows_{\mathcal{S}} q$ for any data environment $\mathcal{S}$.*

Proof. The relation $\leftrightarrows_{\mathcal{S}}$ between the closed terms over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ is a congruence and all the axioms of $\mathrm{BPA}_{\delta\epsilon}$ are sound (cf. theorem 2.2.6). Also REC and RSP are sound (cf. theorem 2.4.5). We have to show the soundness of G1 – G4. We skip the proofs of G1 – G3, which are straightforward, and only show that G4 is sound.

Assume that $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$, $a \in A$, $\phi \in G$ and p, q are closed process expressions over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$. We have to show $(ap + aq + a(\phi p + \neg\phi q), s) \leftrightarrows_{\mathcal{S}} (ap + aq, s)$ for all $s \in S$. We define the relation R as follows:

$$\begin{array}{rcl} R & \stackrel{\mathrm{def}}{=} & \mathit{Id} \cup \{((ap + aq + a(\phi p + \neg\phi q), s), (ap + aq, s)) \mid s \in S\} \\ & & \cup \{((\epsilon(\phi p + \neg\phi q), s), (\epsilon p, s)) \mid s \in S \text{ and } \mathit{test}(\phi, s)\} \\ & & \cup \{((\epsilon(\phi p + \neg\phi q), s), (\epsilon q, s)) \mid s \in S \text{ and } \mathit{test}(\neg\phi, s)\} \end{array}$$

where Id is the identity on $C(\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}, S) \times C(\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}, S)$. In the standard way it follows that R is an $\mathcal{S}$-bisimulation satisfying $(ap + aq + a(\phi p + \neg\phi q), s) R (ap + aq, s)$ for all $s \in S$. □

3.3 Completeness

In this section we prove the completeness of BPA_G^4 with respect to $\mathcal{S}$-bisimilarity. This result differs from the completeness theorem 2.3.5 in the following sense: let p, q be closed terms over $\Sigma(\mathrm{BPA}_G)$.

If for *all* data environments $\mathcal{S}$ we have $p \leftrightarrows_{\mathcal{S}} q$, then $\mathrm{BPA}_G^4 \vdash p = q$.

So completeness says that the axioms of BPA_G^4 are strong enough to prove all identities between closed terms over $\Sigma(\mathrm{BPA}_G)$ that are valid in all data environments, and that $\mathcal{S}$-bisimilarity between terms that cannot be proved in this way depends on the particular parameters of $\mathcal{S}$. If for example the atomic guard ϕ holds in all the data-states of some data environment $\mathcal{S}$, we have $\phi \leftrightarrows_{\mathcal{S}} \epsilon$. Of course we cannot derive $\mathrm{BPA}_G^4 \vdash \phi = \epsilon$, as ϕ is not interpreted as ϵ in all data environments. Proving identities that are dependent on a particular data environment is the topic of the next section.

Observe that some of the results proved in this section concern the axiom system BPA_G^3 (the system containing all axioms of BPA_G^4, except G4). In order to show that BPA_G^4 is complete we first extend the previously defined notion 'prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$' (see 2.3.1) to the closed terms over $\Sigma(\mathrm{BPA}_G)$.

Definition 3.3.1 A closed term p over $\Sigma(\mathrm{BPA}_G)$ is in *prefix normal form over* $\Sigma(\mathrm{BPA}_G)$ iff

$$p ::= \delta \mid \epsilon \mid \phi p \mid \neg\phi p \mid ap \mid p + p$$

with $\phi \in G_{at}$ and $a \in A$. □

The following lemma states that for proving the completeness of BPA_G^4 it is sufficient to restrict our attention to the terms that are in prefix normal form over $\Sigma(\mathrm{BPA}_G)$.

Lemma 3.3.2 *If p is a closed term over $\Sigma(\mathrm{BPA}_G)$, then there is a term p' in prefix normal form over $\Sigma(\mathrm{BPA}_G)$ such that $\mathrm{BPA}_G^3 \vdash p = p'$.*

Proof. By induction on the structure of closed terms. □

We proceed in a similar way as was done in proving the completeness of $\mathrm{BPA}_{\delta\epsilon}$. Because the syntactic format of prefix normal forms over $\Sigma(\mathrm{BPA}_G)$ is not yet adequate for proving the completeness of BPA_G^4, we introduce *reference* sets that will be used to define suitable normal forms.

Definition 3.3.3 *(Reference)*

1. Let p be a closed term over $\Sigma(\mathrm{BPA}_G)$. By $Ref(p)$ we denote the set of atomic guards to which p makes *reference*:

$$Ref(p) \stackrel{\text{def}}{=} \{\phi \in G_{at} \mid \phi \text{ occurs in } p\}.$$

2. Any non-empty, finite subset of G_{at} is called a *reference set*. We use symbols $R, R_1, R_2, \ldots$ to denote reference sets. For technical convenience we assume that reference sets are ordered.

3. Let $R = \{\phi_0, ..., \phi_n\}$ be some (ordered) reference set. A 'sequential' expression $\psi_0 \cdot ... \cdot \psi_n$ is called a *complete guard sequence over* R iff for $i = 0, ..., n$ we have that either $\psi_i \equiv \phi_i$ or $\psi_i \equiv \neg\phi_i$. Such sequences are abbreviated by symbols $\vec{\phi}, \vec{\psi}, ...$ and we write R^{co} for the set of all complete guard sequences over R.

□

We demonstrate two properties of reference sets by a simple observation and a lemma. Let R be some reference set. First observe that if $\vec{\phi}, \vec{\psi} \in R^{co}$, then

$$\mathrm{BPA}_G^3 \vdash \vec{\phi} \cdot \vec{\psi} = \begin{cases} \vec{\phi} & \text{if } \vec{\phi} \equiv \vec{\psi}, \\ \delta & \text{otherwise.} \end{cases}$$

This observation holds because R is ordered: if $\{\phi, \psi\}$ is an unordered reference set, we have by lemma 3.1.1 for instance $\mathrm{BPA}_G^3 \vdash (\phi\psi)(\psi\phi) = \phi\psi$.

In order to denote terms in a convenient way we further use the Σ-notation: let I be some finite index set, then

$$\sum_{i \in I} t_i \stackrel{\text{def}}{=} \begin{cases} \delta & \text{if } I = \emptyset, \\ t_{i_0} + ... + t_{i_n} & \text{if } I = \{i_0, ..., i_n\}. \end{cases}$$

Note that due to the axioms A1 and A2 the actual enumeration of the terms t_{i_j} does not matter.

The following lemma establishes a second property of reference sets that we will often use.

Lemma 3.3.4 *For any t over $\Sigma(\mathrm{BPA}_G)$ and reference set R we have*

$$\mathrm{BPA}_G^3 \vdash t = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} t.$$

Proof. By induction on the cardinality of R:

$R = \{\phi\}$. In this case $t = \epsilon t = (\phi + \neg\phi)t = \phi t + \neg\phi t = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} t$.

$R = \{\phi_0, ..., \phi_{k+1}\}$. Let $R_1 \stackrel{\text{def}}{=} R - \{\phi_0\}$. First applying the induction hypothesis we derive

$$\begin{aligned} t &= \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} t \\ &= (\phi_0 + \neg\phi_0) \cdot \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} t \\ &= \phi_0 \cdot \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} t + \neg\phi_0 \cdot \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} t \\ &= \sum_{\vec{\psi} \in R_1^{co}} \phi_0 \vec{\psi} t + \sum_{\vec{\psi} \in R_1^{co}} \neg\phi_0 \vec{\psi} t \\ &= \sum_{\vec{\phi} \in R^{co}} \vec{\phi} t. \end{aligned}$$

□

Using reference sets, we introduce the following two categories of terms over $\Sigma(\mathrm{BPA}_G)$ that constitute the normal forms we will use in the sequel.

Definition 3.3.5 Let R be some reference set.

1. A closed term p is called *G-basic over R* iff

$$p = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}}$$

where for each $\vec{\phi} \in R^{co}$ the term $q_{\vec{\phi}}$ is an A-basic term over R.

2. A closed term q is called *A-basic over R* iff

$$q = \sum_{i \in I} a_i p_i [+\epsilon]$$

where for each $i \in I$ it holds that $a_i \in A$ and the term p_i is a G-basic term over R. The notation $[+\epsilon]$ means that the occurrence of the syntactic summand ϵ is optional.

□

We show that any closed term over $\Sigma(\mathrm{BPA}_G)$ is provably equal to a G-basic term over some reference set. As a consequence we can further restrict our attention to basic terms in the forthcoming completeness proof.

Lemma 3.3.6 *If p is a closed term over $\Sigma(\mathrm{BPA}_G)$ and R some reference set satisfying $R \supseteq Ref(p)$, then there is a G-basic term p' over R such that $\mathrm{BPA}_G^3 \vdash p = p'$.*

Proof. By lemma 3.3.2 we may assume that p is in prefix normal form over $\Sigma(\mathrm{BPA}_G)$. We apply induction on the structure of such normal forms:

$p \equiv \delta$ **or** $p \equiv \epsilon$. By lemma 3.3.4 we have

$$\delta = \sum_{\vec{\phi} \in R^{co}} \vec{\phi}\delta \text{ and } \epsilon = \sum_{\vec{\phi} \in R^{co}} \vec{\phi}\epsilon, \text{ respectively,}$$

for any reference set R.

$p \equiv \phi q$. Let $R \supseteq Ref(p)$, then $R \supseteq Ref(q)$. By the induction hypothesis we have

$$q = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}}$$

with all the terms $q_{\vec{\phi}}$ A-basic over R. Let for each $\vec{\phi} \in R^{co}$

$$q'_{\vec{\phi}} \equiv \begin{cases} q_{\vec{\phi}} & \text{if } \phi \text{ occurs in } \vec{\phi}, \\ \delta & \text{otherwise,} \end{cases}$$

then

$$\sum_{\vec{\phi} \in R^{co}} \vec{\phi} q'_{\vec{\phi}}$$

is a G-basic term over R that is provably equal to p.

$p \equiv \neg\phi q$. Likewise.

$p \equiv aq$. Let $R \supseteq Ref(p)$, then $R \supseteq Ref(q)$. By the induction hypothesis we have

$$q = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}}$$

with all the terms $q_{\vec{\phi}}$ A-basic over R. By lemma 3.3.4 we have $a = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} a$ and we can take

$$\sum_{\vec{\phi} \in R^{co}} \vec{\phi} a \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}}$$

which clearly is a G-basic term over R provably equal to p.

$p \equiv q + r$. Let $R \supseteq Ref(p)$, then $R \supseteq Ref(q)$ and $R \supseteq Ref(r)$. By the induction hypothesis we have

$$q = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}} \text{ and } r = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q'_{\vec{\phi}}$$

with all the terms $q_{\vec{\phi}}, q'_{\vec{\phi}}$ A-basic over R. Hence

$$q + r = \sum_{\vec{\phi} \in R^{co}} \vec{\phi}(q_{\vec{\phi}} + q'_{\vec{\phi}}).$$

Observe that the sum of two A-basic terms over R is provably equal to an A-basic term over R: change to one index-set or remove a δ-summand and replace double occurrences of ϵ-summands. So for each $\vec{\phi} \in R^{co}$ there is an A-basic term $q''_{\vec{\phi}}$ over R such that $q_{\vec{\phi}} + q'_{\vec{\phi}} = q''_{\vec{\phi}}$. Hence

$$\sum_{\vec{\phi} \in R^{co}} \vec{\phi} q''_{\vec{\phi}}$$

is a G-basic term over R provably equal to p. □

The syntax of an A-basic term is sufficiently strict to derive information about its (syntactic) structure from its operational behaviour (cf. lemma 2.3.3).

Lemma 3.3.7 *Let $\mathcal{S} = \langle S, effect, test \rangle$, and R be some reference set. For any A-basic term q over R the following properties hold:*

1. *If $\exists s \in S$ such that $(q, s) \xrightarrow{\surd} (r, s')$, then $\epsilon \sqsubseteq q$,*
2. *If $\exists s \in S$ such that $(q, s) \xrightarrow{a} (r, s')$ $(a \in A)$, then there is a G-basic term p over R such that $ap \sqsubseteq q$ and $\epsilon p \equiv r$.*

Proof. By using representations of the form

$$\sum_{i \in I} a_i p_i [+\epsilon]$$

and applying induction on the cardinality of I. □

We will also need the following result, which is in fact a generalisation of the axiom G4.

Lemma 3.3.8 (Saturation) *Let R be some reference set. For any $a \in A$, terms $t_0, ..., t_n$ over $\Sigma(\mathrm{BPA}_G)$ and function $f : R^{co} \to \{t_0, ..., t_n\}$ we have*

$$\mathrm{BPA}^4_G \vdash \sum_{i=0}^{n} at_i \sqsupseteq a \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} \cdot f(\vec{\phi}).$$

Proof. By induction on the cardinality of R.

$R = \{\phi\}$. Then

$$a \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} \cdot f(\vec{\phi}) = a(\phi t_j + \neg\phi t_k)$$

for some $j, k \in \{0, ..., n\}$. By the axiom G4 we conclude that $at_j + at_k \sqsupseteq a(\phi t_j + \neg\phi t_k)$, hence

$$\sum_{i=0}^{n} at_i \sqsupseteq a(\phi t_j + \neg\phi t_k).$$

$R = \{\phi_0, ..., \phi_{k+1}\}$. Let $f : R^{co} \to \{t_0, ..., t_n\}$ be given, and $R_1 \stackrel{\text{def}}{=} R - \{\phi_0\}$.
Take $g^i : R_1^{co} \to \{t_0, ..., t_n\}$ $(i = 1, 2)$ such that

$$g^1(\vec{\psi}) \stackrel{\text{def}}{=} f(\phi_0\vec{\psi}) \text{ and } g^2(\vec{\psi}) \stackrel{\text{def}}{=} f(\neg\phi_0\vec{\psi}).$$

First applying the induction hypothesis two times and then the axiom G4 we derive

$$\begin{aligned}
\sum_{i=0}^{n} at_i &\sqsupseteq a \cdot \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} \cdot g^1(\vec{\psi}) + a \cdot \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} \cdot g^2(\vec{\psi}) \\
&\sqsupseteq a\big(\phi_0 \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} \cdot g^1(\vec{\psi}) + \neg\phi_0 \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} \cdot g^2(\vec{\psi})\big) \\
&= a\big(\sum_{\vec{\psi} \in R_1^{co}} \phi_0\vec{\psi} \cdot g^1(\vec{\psi}) + \sum_{\vec{\psi} \in R_1^{co}} \neg\phi_0\vec{\psi} \cdot g^2(\vec{\psi})\big) \\
&= a \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} \cdot f(\vec{\phi}).
\end{aligned}$$

□

The two previous results give us the means to prove a key lemma, in fact stating that whenever two G-basic terms over some reference set R do *not* obey certain provable characteristics, then we can find a data environment $\mathcal{S}$ such that $p \not\leftrightarrow_{\mathcal{S}} q$. Such a data environment is then defined in terms of R.

Definition 3.3.9 Let R be some reference set. We define the data environment $\mathcal{S}(R) = \langle R^{co}, \mathit{effect}, \mathit{test} \rangle$ by

$$\begin{array}{lll} a \in A & \Longrightarrow & \mathit{effect}(a, \vec{\phi}) \stackrel{\text{def}}{=} R^{co}, \\ \phi \in G_{at} & \Longrightarrow & \mathit{test}(\phi, \vec{\phi}) \text{ iff } \phi \text{ occurs in } \vec{\phi}, \text{ or if } \phi \notin R. \end{array}$$

□

The idea is that $\mathcal{S}(R)$ is sufficiently discriminating to distinguish any two G-basic terms over R that are not provably equal.

Lemma 3.3.10 *Let p_1, p_2 be G-basic terms over some reference set R. If there is a syntactic summand $\vec{\phi}q_1$ of p_1 such that for any A-basic term q' over R we have*

$$\mathrm{BPA}_G^4 \vdash q_1 = q' \implies \vec{\phi}q' \not\sqsubseteq p_2,$$

then $(p_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (p_2, \vec{\phi})$.

Proof. Apply induction on $|p_1| + |p_2|$. The case $|p_1| + |p_2| = 0$ is trivial, so let $|p_1| + |p_2| > 0$. By definition p_2 has a syntactic summand $\vec{\phi}q_2$ and by assumption $q_1 \neq q_2$. At least one of the following should hold:

1. $\epsilon \sqsubseteq q_1$ and $\epsilon \not\sqsubseteq q_2$,
2. $\epsilon \sqsubseteq q_2$ and $\epsilon \not\sqsubseteq q_1$,
3. $ar \sqsubseteq q_1$ and $ar \not\sqsubseteq q_2$ for some $a \in A$ and G-basic term r over R,
4. $ar \sqsubseteq q_2$ and $ar \not\sqsubseteq q_1$ for some $a \in A$ and G-basic term r over R.

(If not, then $q_1 \subseteq q_2$ by 1 and 3, and $q_2 \subseteq q_1$ by 2 and 4, so $q_1 = q_2$, contradicting the assumption).

In cases 1 and 2 we have that for one of $(p_1, \vec{\phi})$, $(p_2, \vec{\phi})$ there is a derivable $\surd$-transition, whereas by lemma 3.3.7 this is not the case for the other (for $\epsilon \not\sqsubseteq q_2 \Longrightarrow \epsilon \not\sqsubseteq q_2$). Hence $(p_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (p_2, \vec{\phi})$. We evaluate case 3 (the last case can be dealt with in a similar fashion):

either q_2 has no syntactic summand of the form ar'. Now $(p_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (p_2, \vec{\phi})$, for $(p_1, \vec{\phi})$ has an a-transition, whereas $(p_2, \vec{\phi})$ has no such transition by lemma 3.3.7;

or q_2 has $n+1$ syntactic summands starting with a, say $ar_0, ..., ar_n$ and $r_0, ..., r_n$ G-basic terms over R. Now there is $\vec{\psi}t_{\vec{\psi}} \sqsubseteq r$ such that for all A-basic terms t' over R we have

$$t_{\vec{\psi}} = t' \implies \forall i \in \{0, ..., n\} \ \vec{\psi}t' \not\sqsubseteq r_i$$

If this were not the case, then there would be a function $f : R^{co} \to \{r_0, ..., r_n\}$ such that for any syntactic summand $\vec{\phi}t_{\vec{\phi}}$ of r there is a $t'_{\vec{\phi}}$ satisfying $t_{\vec{\phi}} = t'_{\vec{\phi}}$ and $\vec{\phi}t'_{\vec{\phi}} \sqsubseteq f(\vec{\phi})$.

Using 'saturation' (see lemma 3.3.8) we derive

$$\begin{aligned}\sum_{i=0}^{n} ar_i &\supseteq a \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} \cdot f(\vec{\phi}) \\ &= a \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} t'_{\vec{\phi}} \quad (\vec{\psi} \not\equiv \vec{\phi} \implies \vec{\psi} \cdot \vec{\phi} = \delta) \\ &= a \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} t_{\vec{\phi}} \\ &= ar.\end{aligned}$$

We conclude $ar \subseteq q_2$, which is a contradiction in this case.

By the induction hypothesis we have for $i = 0, ..., n$ that $(r, \vec{\psi}) \not\leftrightarrows_{\mathcal{S}(R)} (r_i, \vec{\psi})$. Now $(p_1, \vec{\phi}) \xrightarrow{a} (\epsilon r, \vec{\psi})$ is a derivable transition that can only be mimicked from $(p_2, \vec{\phi})$ by a transition $(p_2, \vec{\phi}) \xrightarrow{a} (\epsilon r_i, \vec{\psi})$ for some i. As $(\epsilon r, \vec{\psi}) \leftrightarrows_{\mathcal{S}(R)} (r, \vec{\psi})$ and $(\epsilon r_i, \vec{\psi}) \leftrightarrows_{\mathcal{S}(R)} (r_i, \vec{\psi})$ it follows that $(p_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (p_2, \vec{\phi})$. □

With this key lemma on the specific data environment $\mathcal{S}(R)$, the main result of this section follows easily.

Theorem 3.3.11 (Completeness) *Let r_1, r_2 be closed terms over $\Sigma(\mathrm{BPA}_G)$. If $r_1 \leftrightarrows_{\mathcal{S}} r_2$ for all data environments $\mathcal{S}$, then $\mathrm{BPA}^4_G \vdash r_1 = r_2$.*

Proof. We prove the theorem by contraposition. Suppose $r_1 \neq r_2$. We have to find a data environment $\mathcal{S}$ such that $r_1 \not\leftrightarrows_{\mathcal{S}} r_2$.

According to lemma 3.3.6 there are G-basic terms p_1, p_2 over some reference set $R \supseteq Ref(r_1) \cup Ref(r_2)$ such that $\mathrm{BPA}^4_G \vdash r_i = p_i$ $(i = 1, 2)$. By soundness (see theorem 3.2.5) we have $r_i \leftrightarrows_{\mathcal{S}} p_i$ for all $\mathcal{S}$. Because $\mathrm{BPA}^4_G \not\vdash p_1 = p_2$, either p_1 has a syntactic summand $\vec{\phi} q$ such that for any A-basic term q' over R we have $\mathrm{BPA}^4_G \vdash q = q' \implies \vec{\phi} q' \not\sqsubseteq p_2$, or vice versa: if this were not the case, then

$$p_1 = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}} = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q'_{\vec{\phi}} = p_2.$$

This means that the previous lemma 3.3.10 can be applied, hence $(p_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (p_2, \vec{\phi})$ and therefore $(r_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (r_2, \vec{\phi})$. So $r_1 \not\leftrightarrows_{\mathcal{S}(R)} r_2$, which concludes the proof. □

4 BPA with guards in a specific data environment

Up till now we have studied process algebra with guards with respect to the general class of data environments. But often one wants to consider a data environment that is already determined, for instance in the case where actions are assignments and guards are Boolean expressions. Therefore we now investigate bisimulation semantics for $\mathrm{BPA}_{\delta\epsilon}$ with guards in a *specific* data environment. For any data environment satisfying some constraints we present a complete axiomatisation by adding some new axioms to the system BPA^4_G. Finally, we show by an example how we can prove the (partial) correctness of a small program in process algebra.

4.1 Axioms and weakest preconditions

Let A be a set of actions and G_{at} a set of atomic guards. In this section we fix a data environment $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test} \rangle$ over A and G_{at}. Now the axiom system BPA_G^4 need not be complete. Consider for instance two basic guards ϕ and ψ that satisfy $\mathit{test}(\phi, s) \iff \mathit{test}(\psi, s)$ for all $s \in S$, i.e. ϕ and ψ behave the same in all data-states. Obviously we have that $\phi \leftrightarrows_{\mathcal{S}} \psi$, but this cannot be shown using BPA_G^4 because in general $\phi \not\Leftrightarrow_{\mathcal{S}} \psi$. For another example, assume that the process a, starting in a data-state where ϕ holds, always ends in a data-state where ψ holds. In this case $\phi a \leftrightarrows_{\mathcal{S}} \phi a \psi$. Again this cannot be proved in BPA_G^4.

$x + y = y + x$	A1	$\phi \cdot \neg\phi = \delta$	G1
$x + (y + z) = (x + y) + z$	A2	$\phi + \neg\phi = \epsilon$	G2
$x + x = x$	A3	$\phi(x + y) = \phi x + \phi y$	G3
$(x + y)z = xz + yz$	A4		
$(xy)z = x(yz)$	A5	$\phi_0 \cdot \ldots \cdot \phi_n = \delta$	SI
$x + \delta = x$	A6	if $\forall s \in S\ \exists i \leq n$ not $\mathit{test}(\phi_i, s)$	
$\delta x = \delta$	A7		
$\epsilon x = x$	A8	$wp(a, \phi)a\phi = wp(a, \phi)a$	WPC1
$x\epsilon = x$	A9	$\neg wp(a, \phi)a\neg\phi = \neg wp(a, \phi)a$	WPC2

Table 9: The axioms of $\mathrm{BPA}_G(\mathcal{S})$ where $\phi, \phi_i \in G$ and $a \in A$

In table 9 we present the axiom system $\mathrm{BPA}_G(\mathcal{S})$ (implying BPA_G^4) by which we can prove these identities. It contains the first three axioms for guards and three new axioms depending on $\mathcal{S}$ (this explains the $\mathcal{S}$ in $\mathrm{BPA}_G(\mathcal{S})$).

The axiom SI (Sequence is Inaction) expresses that if a sequence of basic guards fails in each data-state, then it equals δ. Note that SI implies G1. The equivalence $\phi \leftrightarrows_{\mathcal{S}} \psi$ mentioned above implies that $\phi\neg\psi = \delta$ and $\neg\phi\psi = \delta$ are in this case instances of SI. We can prove $\mathrm{BPA}_G(\mathcal{S}) \vdash \phi = \psi$ as follows:

$$\begin{aligned} \phi &= \phi(\psi + \neg\psi) \\ &= \phi\psi + \phi\neg\psi \\ &= \phi\psi \\ &= \phi\psi + \neg\phi\psi \\ &= (\phi + \neg\phi)\psi \\ &= \psi. \end{aligned}$$

In the axioms WPC1 and WPC2 (Weakest Preconditions under some Constraints) the expression $wp(a, \phi)$ represents the basic guard that is the *weakest precondition* of an action a and an atomic guard ϕ. Weakest preconditions are semantically defined as follows:

Definition 4.1.1 Let A be a set of actions, G_{at} a set of atomic guards and $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test} \rangle$ be a data environment over A and G_{at}. A *weakest precondition* of an action $a \in A$ and an atomic guard $\phi \in G_{at}$ is a basic guard $\psi \in G$ satisfying for all $s \in S$:

$$\mathit{test}(\psi, s) \text{ iff } \forall s' \in S\ (s' \in \mathit{effect}(a, s) \Longrightarrow \mathit{test}(\phi, s')).$$

If ψ is a weakest precondition of a and ϕ, it is denoted by $wp(a, \phi)$. Weakest preconditions are *expressible* with respect to A, G_{at} and $\mathcal{S}$ iff there is a weakest precondition in G of any $a \in A$ and $\phi \in G_{at}$. □

In the remainder of this section we assume that weakest preconditions are expressible with respect to $\mathcal{S}$. The axioms WPC1 and SI can be used to prove that $\phi a = \phi a \psi$ (see above) whenever process a, starting in a data-state where ϕ holds, always ends in a data-state where ψ holds. In this case, in all data-states where $wp(a, \psi)$ holds, ϕ holds as well. So we have the axioms $\phi \cdot \neg wp(a, \psi) = \delta$ (SI) and $wp(a, \psi)a = wp(a, \psi)a\psi$ (WPC1). We derive:

$$\begin{aligned} \phi a &= \phi(wp(a, \psi) + \neg wp(a, \psi))a \\ &= \phi wp(a, \psi)a \\ &= \phi wp(a, \psi)a\psi \\ &= \phi wp(a, \psi)a\psi + \phi\neg wp(a, \psi)a\psi \\ &= \phi a\psi. \end{aligned}$$

The expressibility of weakest preconditions is not yet sufficient to give an axiomatic characterisation of their properties. For this we also need a constraint on the non-determinism defined by the function *effect*, called *sufficient determinism*.

Definition 4.1.2 Let A be a set of actions and G_{at} a set of atomic guards and let $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test} \rangle$ be a data environment over A and G_{at}. We say that $\mathcal{S}$ is *sufficiently deterministic* iff for all $a \in A$ and $\phi \in G_{at}$:

$$\forall s, s', s'' \in S\ (s', s'' \in \mathit{effect}(a, s) \implies (\mathit{test}(\phi, s') \iff \mathit{test}(\phi, s''))).$$

□

Remark that a data environment with a deterministic function *effect* is sufficiently deterministic. Now if $\mathcal{S}$ is also sufficiently deterministic, then the axioms WPC1 and WPC2 characterise (the properties of) weakest conditions in an algebraic way: WPC1 expresses that $wp(a, \phi)$ is a *precondition* of a and ϕ, and WPC2 states that $wp(a, \phi)$ is the *weakest* precondition of a and ϕ. The following lemma states that the soundness of $\mathrm{BPA}_G(\mathcal{S})$ implies sufficient determinism.

Lemma 4.1.3 *Let $\mathcal{S}$ be some data environment over a set A of atomic actions and a set G_{at} of atomic guards. If weakest preconditions are expressible and $\mathrm{BPA}_G(\mathcal{S})$ is sound, then $\mathcal{S}$ is sufficiently deterministic.*

Proof. Suppose $\mathcal{S}$ is not sufficiently deterministic. So there are $a \in A$, $\phi \in G_{at}$ and $s \in S$ such that we can find $s', s'' \in S$ with

1. $\{s', s''\} \subseteq \mathit{effect}(a, s)$, and
2. $\mathit{test}(\phi, s')$ holds and $\mathit{test}(\phi, s'')$ does *not* hold.

We derive

$$\begin{aligned} a &= wp(a, \phi)a + \neg wp(a, \phi)a \\ &= wp(a, \phi)a\phi + \neg wp(a, \phi)a\neg\phi \end{aligned}$$

but obviously $(a, s) \not\leftrightarrows_{\mathcal{S}} (wp(a,\phi)a\phi + \neg wp(a,\phi)a\neg\phi, s)$, which contradicts the supposition. □

We conclude the introduction of $\mathrm{BPA}_G(\mathcal{S})$ with some small observations. First observe that $\mathrm{BPA}_G(\mathcal{S})$ is not meaningful if weakest preconditions cannot be expressed in $\mathcal{S}$ (we cannot even read its axioms). Furthermore remark that the axiom SI cannot be replaced by the simpler axiom

$$\phi = \psi \text{ if } \forall s \in S\ (test(\phi, s) \Longleftrightarrow test(\psi, s)).$$

If eg. ϕ holds in data-states s_0, s_1 and ψ only holds in s_0, then $\phi\psi \leftrightarrows_{\mathcal{S}} \psi$, but $\phi\psi = \psi$ cannot be derived with the scheme above. Finally, note that the axiom G4 $(a(\phi x + \neg\phi y) \subseteq ax + ay)$ does not occur in table 9, as this axiom is derivable in the following way:

$$\begin{aligned}
\mathrm{BPA}_G(\mathcal{S}) \vdash ax + ay &= (wp(a,\phi) + \neg wp(a,\phi))(ax + ay) \\
&\supseteq wp(a,\phi)ax + \neg wp(a,\phi)ay \\
&= wp(a,\phi)a\phi x + \neg wp(a,\phi)a\neg\phi y \\
&= wp(a,\phi)a\phi(\phi x + \neg\phi y) + \neg wp(a,\phi)a\neg\phi(\phi x + \neg\phi y) \\
&= wp(a,\phi)a(\phi x + \neg\phi y) + wp(a,\neg\phi)a(\phi x + \neg\phi y) \\
&= (wp(a,\phi) + \neg wp(a,\phi))a(\phi x + \neg\phi y) \\
&= a(\phi x + \neg\phi y).
\end{aligned}$$

Remark 4.1.4 Weakest preconditions can be extended to guards as follows (adopting the use of $\neg$ on guards as defined in 3.1):

$wp(a, \neg\alpha)$	abbreviates	$\neg wp(a,\alpha)$
$wp(a, \alpha + \beta)$	abbreviates	$wp(a,\alpha) + wp(a,\beta)$
$wp(a, \alpha\beta)$	abbreviates	$wp(a,\alpha) \cdot wp(a,\beta)$.

Weakest preconditions of guards behave as expected: they satisfy the axiom schemes WPC1 and WPC2 of $\mathrm{BPA}_G(\mathcal{S})$, i.e. we have:

$$\mathrm{BPA}_G(\mathcal{S}) \vdash wp(a,\alpha)a\alpha = wp(a,\alpha)a$$

for any $a \in A$ and guard α over G. We show this in case $\alpha \equiv \neg\beta$:

$$\begin{aligned}
&\text{WPC1}: \quad \neg wp(a,\beta)a\neg\beta = \neg wp(a,\beta)a \quad \text{(from WPC2)} \\
&\text{WPC2}: \quad \neg\neg wp(a,\beta)a\neg\neg\beta = \neg\neg wp(a,\beta)a \quad \text{(from WPC1)}.
\end{aligned}$$

□

4.2 Soundness and completeness

In the following let $\mathcal{S}$ be a data environment over A and G_{at} such that weakest preconditions are expressible and $\mathcal{S}$ is sufficiently deterministic. As stated in lemma 3.2.4, the relation $\leftrightarrows_{\mathcal{S}}$ is a congruence. We state without proof that $\mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP}$ is sound with respect to $\mathcal{S}$ (see theorem 3.2.5, and it is easy to check that the 'new' axioms are sound).

Theorem 4.2.1 (Soundness) *Let $\mathcal{S}$ be a data environment that has weakest preconditions and that is sufficiently deterministic. Let p, q be closed terms over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$.*
If $\mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash p = q$, then $p \leftrightarrows_{\mathcal{S}} q$. □

We show that the axiom system $\mathrm{BPA}_G(\mathcal{S})$ completely axiomatises bisimulation equivalence in $\mathcal{S}$, i.e. the relation $\leftrightarrows_{\mathcal{S}}$, between the closed terms over $\Sigma(\mathrm{BPA}_G)$. In order to do so we will use some results of sections 2 and 3, though we do not need the concepts of A-basic and G-basic terms over $\Sigma(\mathrm{BPA}_G)$ from section 3. Reason for this is that weakest preconditions allow us to manipulate closed terms over $\Sigma(\mathrm{BPA}_G)$ in such a way that any basic guard different from δ, ϵ can occur only at 'head level'. This makes it possible to use a much simpler type of basic terms in proving completeness. We first illustrate what kind of manipulation we mean. As an example consider the term $a\neg\phi c(b+\epsilon)$. We derive

$$\begin{aligned} a\neg\phi c(b+\epsilon) &= wp(a,\phi)a\neg\phi c(b+\epsilon) + \neg wp(a,\phi)a\neg\phi c(b+\epsilon) \\ &= wp(a,\phi)a\phi\neg\phi c(b+\epsilon) + \neg wp(a,\phi)ac(b+\epsilon) \\ &= wp(a,\phi)a\delta + \neg wp(a,\phi)ac(b+\epsilon) \end{aligned}$$

with all basic guards different from δ, ϵ at head level. Using the possibility to push basic guards to head level as illustrated above, it suffices to slightly extend the notion of prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$ (see definition 2.3.1) to the following syntactic category.

Definition 4.2.2 A term p over $\Sigma(\mathrm{BPA}_G)$ is called *basic over* some reference set R iff the following conditions hold:

1. $\mathrm{A1}, \mathrm{A2} \vdash p = \sum_{\vec{\phi}\in R^{co}} \vec{\phi} q_{\vec{\phi}}$,
2. for all $\vec{\phi} \in R^{co}$ the term $q_{\vec{\phi}}$ is a term in prefix normal form over $\mathrm{BPA}_{\delta\epsilon}$.

□

In the following two lemma's we show that for any closed term p over $\Sigma(\mathrm{BPA}_G)$ there exists a basic term p' (over some reference set) satisfying

$$\mathrm{BPA}_G(\mathcal{S}) \vdash p = p'.$$

Hence we may restrict our attention to basic terms in proving completeness, and exploit their syntactic structure. Particularly, if two basic terms p, q are *not* provably equal, then there is a data-state s such that $(p,s) \not\leftrightarrows_{\mathcal{S}} (q,s)$.

Lemma 4.2.3 *Let $a \in A$ and R be some reference set. For any term t over $\Sigma(\mathrm{BPA}_G)$*

$$\mathrm{BPA}_G(\mathcal{S}) \vdash t = \sum_{\vec{\phi}\in R^{co}} wp(a,\vec{\phi}) \cdot t.$$

Proof. By induction on the cardinality of R. □

Lemma 4.2.4 (Basic form) *If p is a closed term over $\Sigma(\mathrm{BPA}_G)$, then there is a basic term p' over some reference set R such that $\mathrm{BPA}_G(\mathcal{S}) \vdash p = p'$.*

Proof. By lemma 3.3.2 we may assume that p is a term in prefix normal form over $\Sigma(\mathrm{BPA}_G)$ and we apply induction on the structure of p:

$p \equiv \delta$ **or** $p \equiv \epsilon$. By lemma 3.3.4 we have

$$\delta = \sum_{\vec{\phi} \in R^{co}} \vec{\phi}\delta \text{ and } \epsilon = \sum_{\vec{\phi} \in R^{co}} \vec{\phi}\epsilon, \text{ respectively,}$$

for any reference set R.

$p \equiv \phi q$. By the induction hypothesis there is a reference set R such that

$$q = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}}$$

with all the terms $q_{\vec{\phi}}$ in prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. Let $R_1 \stackrel{\text{def}}{=} \{\phi\} \cup R$. By lemma 3.3.4 we have

$$\begin{aligned} \phi q &= \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi}\phi \cdot \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}} \\ &= \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} \cdot q'_{\vec{\psi}} \end{aligned}$$

where for all $\vec{\psi} \in R_1^{co}$

$$q'_{\vec{\psi}} \equiv \begin{cases} q_{\vec{\phi}} & \text{if } \phi \text{ occurs in } \vec{\psi} \text{ and } \vec{\phi} \text{ occurs in } \vec{\psi}, \\ \delta & \text{otherwise.} \end{cases}$$

Furthermore

$$\sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} q'_{\vec{\psi}}$$

is clearly a basic term over R_1.

$p \equiv \neg\phi q$. Likewise.

$p \equiv aq$. By the induction hypothesis there is a reference set R such that

$$q = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q_{\vec{\phi}}$$

with all the terms $q_{\vec{\phi}}$ in prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. We derive

$$\begin{aligned} aq &= a \cdot \sum_{\vec{\psi} \in R^{co}} \vec{\psi} q_{\vec{\psi}} \\ &= \sum_{\vec{\phi} \in R^{co}} wp(a, \vec{\phi}) \cdot a \cdot \sum_{\vec{\psi} \in R^{co}} \vec{\psi} q_{\vec{\psi}} && \text{(by lemma 4.2.3)} \\ &= \sum_{\vec{\phi} \in R^{co}} wp(a, \vec{\phi}) \cdot a \cdot \vec{\phi} q_{\vec{\phi}} && (\vec{\psi} \not\equiv \vec{\phi} \implies \vec{\phi} \cdot \vec{\psi} = \delta) \\ &= \sum_{\vec{\phi} \in R^{co}} wp(a, \vec{\phi}) \cdot a \cdot q_{\vec{\phi}} \end{aligned}$$

Let $wp(a,R) \stackrel{\text{def}}{=} \{Ref(wp(a,\phi)) \mid \phi \in R\}$. Note that $wp(a,R)$ may be empty (for instance in case $R = \{\phi\}$ and $wp(a,\phi) = \epsilon$). Let

$$R_1 \stackrel{\text{def}}{=} \{\phi\} \cup wp(a,R)$$

for some arbitrary $\phi \in G_{at}$. Obviously R_1 is a reference set, and we derive

$$aq = \sum_{\vec{\phi} \in R^{co}} wp(a,\vec{\phi}) \cdot a \cdot q_{\vec{\phi}} = \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} \cdot \sum_{\{\vec{\phi} \in R^{co} \mid \phi \cdot wp(a,\vec{\phi}) = \vec{\psi} \vee \neg\phi \cdot wp(a,\vec{\phi}) = \vec{\psi}\}} a \cdot q_{\vec{\phi}}$$

with the rightmost term basic over R_1.

$p \equiv q + r$. By the induction hypothesis there are reference sets R_1, R_2 such that

$$q = \sum_{\vec{\psi} \in R_1^{co}} \vec{\psi} q_{\vec{\psi}} \text{ and } r = \sum_{\vec{\theta} \in R_2^{co}} \vec{\theta} r_{\vec{\theta}}$$

with all the terms $q_{\vec{\psi}}, r_{\vec{\theta}}$ in prefix normal form over $\Sigma(\mathrm{BPA}_{\delta\epsilon})$. Let $R \stackrel{\text{def}}{=} R_1 \cup R_2$. By lemma 3.3.4 we have

$$q = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} q'_{\vec{\phi}} \text{ and } r = \sum_{\vec{\phi} \in R^{co}} \vec{\phi} r'_{\vec{\phi}}.$$

where for all $\vec{\phi} \in R^{co}$ the terms $q'_{\vec{\phi}}, r'_{\vec{\phi}}$ are defined as follows:

$$\begin{aligned} q'_{\vec{\phi}} &\equiv q_{\vec{\psi}}, \quad \text{provided } \vec{\psi} \text{ occurs in } \vec{\phi}, \\ r'_{\vec{\phi}} &\equiv r_{\vec{\theta}}, \quad \text{provided } \vec{\theta} \text{ occurs in } \vec{\phi}. \end{aligned}$$

We derive

$$q + r = \sum_{\vec{\phi} \in R^{co}} \vec{\phi}(q'_{\vec{\phi}} + r'_{\vec{\phi}})$$

and clearly the right hand side term is basic over R. □

The syntax of a basic term is sufficiently strict to derive information about its (syntactic) structure from its operational behaviour. As announced before, we show that if two basic terms over some reference set R do *not* obey certain provable characteristics, then we can find a data-state $s \in S$ such that the associated transition systems with initial data-state s are *not* $\mathcal{S}$-bisimilar. The proof of this fact uses properties of the subsystem $\mathrm{BPA}_{\delta\epsilon}$ of $\mathrm{BPA}_G(\mathcal{S})$, and it is quite easy compared to the proof of the related lemma 3.3.10.

Lemma 4.2.5 *Let p_1, p_2 be basic terms over some reference set R. If there is $\vec{\phi} \in R^{co}$ such that*

1. $\vec{\phi} q^i_{\vec{\phi}} \sqsubseteq p_i$ $(i = 1,2)$,
2. $\mathrm{BPA}_G(\mathcal{S}) \not\vdash \vec{\phi} = \delta$,

3. $\mathrm{BPA}_{\delta\epsilon} \not\vdash q^1_{\vec{\phi}} = q^2_{\vec{\phi}}$,

then $\exists s \in S\ ((p_1, s) \not\leftrightarrows_{\mathcal{S}} (p_2, s))$.

Proof. Assume that $\vec{\phi}$ satisfies the condition of the lemma. So in particular we can find $s \in S$ such that $(\vec{\phi}, s) \xrightarrow{\surd} (\delta, s)$ (otherwise the axiom SI violates the second condition).

Now suppose $(p_1, s) \leftrightarrows_{\mathcal{S}} (p_2, s)$ by some $\mathcal{S}$-bisimulation B. Adding the tuple $((q^1_{\vec{\phi}}, s), (q^2_{\vec{\phi}}, s))$ to B would by the first condition result in an $\mathcal{S}$-bisimulation establishing $(q^1_{\vec{\phi}}, s) \leftrightarrows_{\mathcal{S}} (q^2_{\vec{\phi}}, s)$. But by lemma 2.3.4 this contradicts the third condition. Hence $(p_1, s) \not\leftrightarrows_{\mathcal{S}} (p_2, s)$. □

Connecting all the results proved so far, we can prove the completeness of $\mathrm{BPA}_G(\mathcal{S})$ in a simple way.

Theorem 4.2.6 (Completeness) *Let $\mathcal{S}$ be a data environment that has weakest preconditions and that is sufficiently deterministic. Let r_1, r_2 be closed terms over $\Sigma(\mathrm{BPA}_G)$. If $r_1 \leftrightarrows_{\mathcal{S}} r_2$, then $\mathrm{BPA}_G(\mathcal{S}) \vdash r_1 = r_2$.*

Proof. We prove the theorem by contraposition. Suppose $r_1 \neq r_2$. We have to show $r_1 \not\leftrightarrows_{\mathcal{S}} r_2$. According to lemma 4.2.4 there are basic terms p'_1, p'_2 over reference sets R_1, R_2, respectively, such that $r_i = p'_i$ $(i = 1, 2)$. By lemma 3.3.4 we can find basic terms p_1, p_2 over $R = R_1 \cup R_2$ such that $p_i = p'_i$, and hence $r_i = p_i$ $(i = 1, 2)$. By soundness (see theorem 3.2.5) we have that $r_i \leftrightarrows_{\mathcal{S}} p_i$. Because $p_1 \neq p_2$, there must be $\vec{\phi} \in R^{co}$ satisfying

$$\mathrm{BPA}_G(\mathcal{S}) \not\vdash \vec{\phi} = \delta \quad \text{and} \quad \mathrm{BPA}_{\delta\epsilon} \not\vdash q^1_{\vec{\phi}} = q^2_{\vec{\phi}}.$$

By the previous lemma 4.2.5 we can find some $s \in S$ such that $(p_1, s) \not\leftrightarrows_{\mathcal{S}} (p_2, s)$. As $\leftrightarrows_{\mathcal{S}}$ is an equivalence relation, we conclude $(r_1, s) \not\leftrightarrows_{\mathcal{S}} (r_2, s)$, which finishes our proof. □

4.3 An example: the process *SWAP*

Process algebra with guards can be used to express and prove partial correctness formulas in Hoare logic. In section 6 we elaborate on this idea. Here a simple example that is often used as an illustration of Hoare logic is presented and its correctness is shown.

First we transform $\mathrm{BPA}_G(\mathcal{S})$ into a small programming language with Boolean guards and assignments. Our language has the signature of $\Sigma(\mathrm{BPA}_G)$ and we have some set $\mathcal{V} = \{x, y, ...\}$ of data variables. Atomic actions have the form:

$$x := e$$

with $x \in \mathcal{V}$ a variable ranging over the set $\mathbb{Z}$ of integers and e an integer expression. We assume that some interpretation $[\![\cdot]\!]$ from closed integer expressions to integers is given. Atomic guards have the form

$$e = f$$

where e and f are both integer expressions. For readability we sometimes write angular brackets around guards and square brackets around assignments.

The components of the data environment $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ are straightforward to define:

$$S = \mathbb{Z}^{\mathcal{V}}$$

i.e. the set of mappings from $\mathcal{V}$ to the integers. We write ρ, σ for data-states in S, and we assume that the domain $\mathcal{V}$ of these mappings is extended to integer expressions in the standard way. The function *effect* is defined by:

$$\mathit{effect}(x := e, \rho) = \rho[[\![\rho(e)]\!]/x]$$

where $\rho[n/x]$ is as the mapping ρ, except that x is mapped to n. We define the predicate *test* by:

$$\mathit{test}(e = f, \rho) \Longleftrightarrow ([\![\rho(e)]\!] = [\![\rho(f)]\!]).$$

Note that the effect function is deterministic, so $\mathcal{S}$ is certainly sufficiently deterministic. Weakest preconditions can easily be expressed:

$$wp(x := e, e_1 = e_2) = \langle e_1[e/x] = e_2[e/x]\rangle.$$

The axiom SI cannot be formulated so easily, partly because we have not yet defined integer expressions very precisely. We characterise SI by the scheme:

$$\langle e_0 = f_0\rangle \cdot \ldots \cdot \langle e_n = f_n\rangle = \delta$$

if $\forall \rho \in S$ we can find $i \leq n$ such that $[\![\rho(e_i)]\!] \neq [\![\rho(f_i)]\!]$.

In this language we can express the following tiny program $SWAP$ that exchanges the initial values of x and y without using any other variables.

$$\begin{aligned} SWAP \;=\; & x := x + y \\ & y := x - y \\ & x := x - y. \end{aligned}$$

The correctness of this program can be expressed by the following equation:

$$\langle x = n\rangle\langle y = m\rangle SWAP = \langle x = n\rangle\langle y = m\rangle SWAP\langle x = m\rangle\langle y = n\rangle.$$

This equation says that if $SWAP$ is executed in an initial data-state where $x = n$ and $y = m$, then after termination of $SWAP$ it must hold, i.e. it can be derived, that $x = m$ and $y = n$. So $SWAP$ indeed exchanges the values of x and y.

The correctness of $SWAP$ can be proved as follows:

$$\begin{aligned} \langle x = n\rangle\langle y = m\rangle SWAP &\overset{\text{SI}}{=} \langle (x + y) - y = n\rangle\langle (x + y) - ((x + y) - y) = m\rangle SWAP \\ &\overset{\text{SI,WPC1}}{=} \langle x = n\rangle\langle y = m\rangle[x := x + y]\langle x - y = n\rangle\langle x - (x - y) = m\rangle[y := x - y][x := x - y] \\ &\overset{\text{WPC1}}{=} \langle x = n\rangle\langle y = m\rangle[x := x + y][y := x - y]\langle y = n\rangle\langle x - y = m\rangle[x := x - y] \\ &\overset{\text{WPC1}}{=} \langle x = n\rangle\langle y = m\rangle SWAP\langle x = m\rangle\langle y = n\rangle. \end{aligned}$$

5 Parallel processes with guards

In this section Basic Process Algebra with guards is extended with operators for parallelism. We give Plotkin-style rules to express the operational behaviour of these operators and show that $\mathcal{S}$-bisimilarity is not a congruence any longer. We deal with this problem by introducing another bisimulation equivalence, called *global $\mathcal{S}$-bisimulation equivalence* which is finer than $\mathcal{S}$-bisimilarity. Global $\mathcal{S}$-bisimulation equivalence is a congruence, but it is not so natural. Moreover, the axioms WPC1, WPC2 and G4 are not valid anymore in global $\mathcal{S}$-bisimulation.

We present the axiom system ACP_G which is based on ACP (the Algebra of Communicating Processes [BK84a]). ACP_G is sound for global $\mathcal{S}$-bisimilarity, and for finite processes also complete. This axiom system enables us to prove $\mathcal{S}$-bisimulation equivalence between processes: using ACP_G every closed process term can be proved equivalent to one without parallel operators, and then BPA^4_G or $\mathrm{BPA}_G(\mathcal{S})$ can be used to prove $\mathcal{S}$-bisimilarity. This section is concluded with an example in which the correctness of a parallel process is proved in this way.

5.1 Axioms and a two-phase calculus

We extend the language of $\Sigma(\mathrm{BPA}_G)$ to a concurrent one, suitable to describe the behaviour of parallel, communicating processes. Communication is modelled by a communication function $\gamma : A \times A \longrightarrow A_\delta$ that is commutative and associative. If $\gamma(a,b)$ is δ, then a and b cannot communicate, and if $\gamma(a,b) = c$, then c is the atomic action resulting from the communication between a and b.

Concurrency is described by three operators, the *merge* $\|$, the *left-merge* $\lfloor\!\lfloor$ and the *communication-merge* $|$.

$p \| q$ represents the parallel execution of p and q. It starts when one of its components starts, and terminates if both of them do.

$p \lfloor\!\lfloor q$ is as $p \| q$, but under the assumption that the first action that is performed comes from p (it may be the case that the behaviour of p starts with the evaluation of a guard).

$p \mid q$ is as $p \| q$, but the first action is a communication between p and q.

We present encapsulation operators ∂_H (for any $H \subseteq A$) that block actions in H by renaming them into δ. Encapsulation is used to enforce communication between processes. The signature $\Sigma(\mathrm{ACP}_G)$ is summarised in table 10.
For the terms over $\Sigma(\mathrm{ACP}_G)$ we have the axioms given in table 11, where $a, b \in A$, $H \subseteq A$ and $\phi \in G$ (note that the axiom $a(\phi x + \neg\phi y) \subseteq ax + ay$ (G4) is absent). Most of these axioms are standard for ACP (see [BK84a]), and, apart from G1, G2 and G3, only the axioms EM10, EM11 and D0 are new. The axiom EM10 (EM11) expresses that a basic guard ϕ in $\phi x \lfloor\!\lfloor y$ ($\phi x \mid y$, respectively) also may prevent that y happens.

Using ACP_G any closed term over $\Sigma(\mathrm{ACP}_G)$ can be proved equal to one without merge operators, i.e. a closed term over $\Sigma(\mathrm{BPA}_G)$.

Theorem 5.1.1 (Elimination) *Let p be a closed term over $\Sigma(\mathrm{ACP}_G)$, then there is a closed term over q over $\Sigma(\mathrm{BPA}_G)$ such that $\mathrm{ACP}_G \vdash p = q$.*

constants:	a	for any atomic action $a \in A$
	ϕ	for any basic guard $\phi \in G$
unary operators:	∂_H	encapsulation, for any $H \subseteq A$
binary operators:	$+$	alternative composition (sum)
	$\cdot$	sequential composition (product)
	$\|\|$	parallel composition (merge)
	$\lfloor\!\lfloor$	left-merge
	$\mid$	communication-merge

Table 10: The signature $\Sigma(\mathrm{ACP}_G)$

Proof. By induction on the structure of terms. □

ACP_G and BPA^4_G or $\mathrm{BPA}^3_G(\mathcal{S})$ cannot be combined in bisimulation semantics; if G4 is added to ACP_G we can derive the following:

$$\begin{aligned} \mathrm{ACP}_G + \mathrm{G4} \quad &\vdash \quad a(b \parallel d) + a(c \parallel d) + d(ab + ac) && (1)\\ &= \quad (ab + ac) \parallel d \\ &\overset{\mathrm{G4}}{=} \quad (ab + ac + a(\phi b + \neg\phi c)) \parallel d \\ &\supseteq \quad a(\phi bd + \neg\phi cd + d(\phi b + \neg\phi c)). && (2) \end{aligned}$$

So, in (2) after an a step we arrive in a state where we still can do both a b or c step, assuming that the effect of b yields data-states where ϕ holds and data-states where $\neg\phi$ holds. This cannot be mimicked in (1). Therefore, every term with (2) as a summand is not bisimilar to (1) for any reasonable form of bisimulation. So $\mathrm{ACP}_G + \mathrm{G4}$ is not sound in any bisimulation semantics.

Because we still want to derive $\mathcal{S}$-bisimilarity between closed terms containing merge operators, we introduce a *two-phase* calculus that does not have these problems. Derivability in this calculus is denoted by $\vdash_c$.

Definition 5.1.2 Let p_1, p_2 be closed terms over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$. We write

$$\mathrm{ACP}^4_G \vdash_c p_1 = p_2$$

iff there are closed terms q_1, q_2 over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ such that $\mathrm{ACP}_G \vdash p_i = q_i \quad (i = 1, 2)$ and $\mathrm{BPA}^4_G \vdash q_1 = q_2$. Furthermore, we write

$$\mathrm{ACP}_G(\mathcal{S}) \vdash_c p_1 = p_2$$

iff there are closed terms q_1, q_2 over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ such that $\mathrm{ACP}_G \vdash p_i = q_i \quad (i = 1, 2)$ and $\mathrm{BPA}_G(\mathcal{S}) \vdash q_1 = q_2$.

We sometimes put REC + RSP in front of $\vdash_c$ which means that we may use REC and RSP in proving $p_i = q_i$ $(i = 1, 2)$ and $q_1 = q_2$. □

$x + (y + z) = (x + y) + z$	A1	$\phi \cdot \neg\phi = \delta$	G1
$x + y = y + x$	A2	$\phi + \neg\phi = \epsilon$	G2
$x + x = x$	A3	$\phi(x + y) = \phi x + \phi y$	G3
$(x + y)z = xz + yz$	A4		
$(xy)z = x(yz)$	A5		
$x + \delta = x$	A6		
$\delta x = \delta$	A7		
$\epsilon x = x$	A8		
$x\epsilon = x$	A9		
$a \mid b = \gamma(a, b)$	CF		
$x \parallel y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x \mid y$	EM1	$\phi x \lfloor\!\lfloor y = \phi(x \lfloor\!\lfloor y)$	EM10
$\epsilon \lfloor\!\lfloor x = \delta$	EM2	$\phi x \mid y = \phi(x \mid y)$	EM11
$ax \lfloor\!\lfloor y = a(x \parallel y)$	EM3		
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	EM4		
$x \mid y = y \mid x$	EM5	$\partial_H(\phi) = \phi$	D0
$\epsilon \mid \epsilon = \epsilon$	EM6	$\partial_H(a) = a$ if $a \notin H$	D1
$\epsilon \mid ax = \delta$	EM7	$\partial_H(a) = \delta$ if $a \in H$	D2
$ax \mid by = (a \mid b)(x \parallel y)$	EM8	$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3
$(x + y) \mid z = x \mid z + y \mid z$	EM9	$\partial_H(xy) = \partial_H(x)\partial_H(y)$	D4

Table 11: The axioms of ACP_G where $\phi \in G$, $a, b \in A$ and $H \subseteq A$

5.2 Operational semantics and soundness

Let $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ be some data environment over a set A of atomic actions and a set G_{at} of atomic guards. The transition rules in table 12 and the transition rule for guarded recursive specifications (see table 5) determine the transition relation $\longrightarrow_{\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}},\mathcal{S}}$ over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$. Remark that these rules formalise the informal description of the new operators given earlier, and that all rules given for $\Sigma(\mathrm{BPA}_G)$ in table 8 are included. Let p be a closed term over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$. For any $s \in S$ the transition system $\mathcal{A}(p,s)$ is defined as

$$\mathcal{A}(p,s) \stackrel{\mathrm{def}}{=} \langle C(\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}, S)\ ,\ A_{\surd}\ ,\ \longrightarrow_{\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}},\mathcal{S}}\ ,\ (p,s)\rangle.$$

We first show by an example that the notion of '$\mathcal{S}$-bisimilarity' as defined in 2.2.4 for the configurations over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$ gives in general no congruence relation between the closed terms over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$.

Example 5.2.1 Consider the data environment $\langle\{s_0, s_1\}, \mathit{effect}, \mathit{test}\rangle$ in which

- $\forall s \in S\ (\mathit{effect}(a,s) = \{s_0\})$ for some $a \in A$;
- $\forall s \in S\ (\mathit{effect}(b,s) = \{s_1\})$ for some $b \in A$;
- $\mathit{test}(\phi, s_0)$ and not $\mathit{test}(\phi, s_1)$ for some $\phi \in G$.

In this case we have $a\delta \leftrightarrows_{\mathcal{S}} a\neg\phi$ but not $a\delta \parallel b \leftrightarrows_{\mathcal{S}} a\neg\phi \parallel b$, for the transition system $\mathcal{A}(a\neg\phi \parallel b, s_0)$ has an execution path

$$(a\neg\phi \parallel b, s_0) \stackrel{a}{\longrightarrow} (b\neg\phi, s_0) \stackrel{b}{\longrightarrow} (\neg\phi, s_1) \stackrel{\surd}{\longrightarrow} (\delta, s_1)$$

that is not present in $\mathcal{A}(a\delta \parallel b, s_0)$. *(End example.)*

We define a different bisimulation equivalence, called *global $\mathcal{S}$-bisimilarity*, that is a congruence for the merge operators. The idea behind a global $\mathcal{S}$-bisimulation is that a context $p \parallel (.)$ around a process q can change the data-state of q at any time and global $\mathcal{S}$-bisimulation equivalence must be resistant against such changes. So, a configuration (p_1, s) is related to a configuration (p_2, s) if $(p_1, s) \stackrel{a}{\longrightarrow} (q_1, s')$ implies $(p_2, s) \stackrel{a}{\longrightarrow} (q_2, s')$ and, as the environment may change s', q_1 is related to q_2 in *any* data-state:

Definition 5.2.2 Let Σ be a signature, $\mathcal{S}$ a data environment with data-state space S and $\longrightarrow_{\mathcal{S}}$ a transition relation over $C(\Sigma, S)$.

- A binary relation $R \subseteq C(\Sigma,S) \times C(\Sigma,S)$ is a *global $\mathcal{S}$-bisimulation* iff R satisfies the following (global) version of the transfer property: for all $(p,s),(q,s) \in C(\Sigma,S)$ with $(p,s)R(q,s)$:
  1. whenever $(p,s) \stackrel{a}{\longrightarrow}_{\mathcal{S}} (p',s')$ for some a and (p',s'), then, for some q', also $(q,s) \stackrel{a}{\longrightarrow}_{\mathcal{S}} (q',s')$ and $\forall s'' \in S\ ((p',s'')R(q',s''))$,
  2. conversely, whenever $(q,s) \stackrel{a}{\longrightarrow}_{\mathcal{S}} (q',s')$ for some a and (q',s'), then, for some p', also $(p,s) \stackrel{a}{\longrightarrow}_{\mathcal{S}} (p',s')$ and $\forall s'' \in S\ ((p',s'')R(q',s''))$.

$a \in A \quad (a, s) \xrightarrow{a} (\epsilon, s') \quad \text{if } s' \in \mathit{effect}(a, s)$

$\phi \in G \quad (\phi, s) \xrightarrow{\surd} (\delta, s) \quad \text{if } \mathit{test}(\phi, s)$

$+ \quad \dfrac{(x, s) \xrightarrow{a} (x', s')}{(x + y, s) \xrightarrow{a} (x', s')} \qquad \dfrac{(y, s) \xrightarrow{a} (y', s')}{(x + y, s) \xrightarrow{a} (y', s')}$

$\cdot \quad \dfrac{(x, s) \xrightarrow{a} (x', s')}{(xy, s) \xrightarrow{a} (x'y, s')} \; a \neq \surd \qquad \dfrac{(x, s) \xrightarrow{\surd} (x', s') \quad (y, s) \xrightarrow{a} (y', s'')}{(xy, s) \xrightarrow{a} (y', s'')}$

$\| \quad \dfrac{(x, s) \xrightarrow{a} (x', s')}{(x \parallel y, s) \xrightarrow{a} (x' \parallel y, s')} \; a \neq \surd \qquad \dfrac{(y, s) \xrightarrow{a} (y', s')}{(x \parallel y, s) \xrightarrow{a} (x \parallel y', s')} \; a \neq \surd$

$\dfrac{(x, s) \xrightarrow{a} (x', s') \quad (y, s) \xrightarrow{b} (y', s'')}{(x \parallel y, s) \xrightarrow{\gamma(a,b)} (x' \parallel y', s''')} \; \gamma(a, b) \neq \delta, \; a, b \neq \surd \text{ and } s''' \in \mathit{effect}(\gamma(a, b), s)$

$\dfrac{(x, s) \xrightarrow{\surd} (x', s') \quad (y, s) \xrightarrow{\surd} (y', s')}{(x \parallel y, s) \xrightarrow{\surd} (x' \parallel y', s')}$

$\between \quad \dfrac{(x, s) \xrightarrow{a} (x', s')}{(x \between y, s) \xrightarrow{a} (x' \parallel y, s')} \; a \neq \surd$

$| \quad \dfrac{(x, s) \xrightarrow{a} (x', s') \quad (y, s) \xrightarrow{b} (y', s'')}{(x \mid y, s) \xrightarrow{\gamma(a,b)} (x' \parallel y', s''')} \; \gamma(a, b) \neq \delta, \; a, b \neq \surd \text{ and } s''' \in \mathit{effect}(\gamma(a, b), s)$

$\dfrac{(x, s) \xrightarrow{\surd} (x', s') \quad (y, s) \xrightarrow{\surd} (y', s')}{(x \mid y, s) \xrightarrow{\surd} (x' \parallel y', s')}$

$\partial_H \quad \dfrac{(x, s) \xrightarrow{a} (x', s')}{(\partial_H(x), s) \xrightarrow{a} (\partial_H(x'), s')} \; \text{if } a \notin H \subseteq A$

Table 12: Transition rules for ACP_G ($a, b \in A_{\surd}$, $H \subseteq A$)

- A configuration $(p, s) \in C(\Sigma, S)$ is *globally S-bisimilar* to a configuration $(q, s') \in C(\Sigma, S)$, notation $(p, s) \leftrightarrows_{\mathcal{S}} (q, s')$, iff $s = s'$ and there is a global $\mathcal{S}$-bisimulation containing the pair $((p, s), (q, s'))$.

- A transition system $\mathcal{A}(p, s) = \langle C(\Sigma, S), A_{\surd}, \longrightarrow_{\mathcal{S}}, (p, s)\rangle$ is *globally $\mathcal{S}$-bisimilar* with a transition system $\mathcal{A}(q, s') = \langle C(\Sigma, S), A_{\surd}, \longrightarrow_{\mathcal{S}}, (q, s')\rangle$, notation $\mathcal{A}(p, s) \leftrightarrows_{\mathcal{S}} \mathcal{A}(q, s')$, iff $(p, s) \leftrightarrows_{\mathcal{S}} (q, s')$.

- Two closed terms p, q over Σ are *globally $\mathcal{S}$-bisimilar*, notation $p \leftrightarrows_{\mathcal{S}} q$, iff

$$\mathcal{A}(p, s) \leftrightarrows_{\mathcal{S}} \mathcal{A}(q, s)$$

  for all $s \in S$.

□

By definition of global $\mathcal{S}$-bisimilarity we have for any two closed terms p, q over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$

$$p \leftrightarrows_{\mathcal{S}} q \implies p \leftrightarrows_{\mathcal{S}} q.$$

It is not difficult to see that for any data environment $\mathcal{S}$ the relation $\leftrightarrows_{\mathcal{S}}$ is an equivalence relation over the closed terms over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$.

Our goal, i.e. global $\mathcal{S}$-bisimilarity being a congruence relation, has been achieved:

Lemma 5.2.3 *For any data environment $\mathcal{S}$ the relation $\leftrightarrows_{\mathcal{S}}$ is a congruence with respect to the operators of $\Sigma(\mathrm{ACP}_G)$.*

Proof. We only prove the lemma for the merge operator. Let $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ and assume that $p \leftrightarrows_{\mathcal{S}} p'$ and $q \leftrightarrows_{\mathcal{S}} q'$. So for all $s \in S$ we have global $\mathcal{S}$-bisimulations R_p^s and R_q^s such that $(p, s)R_p^s(p', s)$ and $(q, s)R_q^s(q', s)$. We have to show $(p \parallel q, s) \leftrightarrows_{\mathcal{S}} (p' \parallel q', s)$ for all $s \in S$. Fix $s_0 \in S$, and let $R_p \stackrel{\mathrm{def}}{=} \cup_{s\in S} R_p^s$ and $R_q \stackrel{\mathrm{def}}{=} \cup_{s\in S} R_q^s$. We define a relation R as follows:

$$R \stackrel{\mathrm{def}}{=} \{((r \parallel u, s), (r' \parallel u', s)) \mid (r, s)R_p(r', s)\ ,\ (u, s)R_q(u', s)\}$$

We have $(p \parallel q, s_0)R(p' \parallel q', s_0)$ and we show that R is a global $\mathcal{S}$-bisimulation. Suppose

$$(r \parallel u, s)R(r' \parallel u', s) \quad \text{and} \quad (r \parallel u, s) \xrightarrow{a} (v \parallel w, s').$$

We systematically check which application of the transition rules may have led to this transition:

$(r, s) \xrightarrow{a} (v, s')$, $u \equiv w$ and $a \neq \surd$. Because $(r, s)R_p(r', s)$ and R_p is (also) a global $\mathcal{S}$-bisimulation, there is a v' such that $(r', s) \xrightarrow{a} (v', s')$ and $\forall s''((v, s'')R_p(v', s''))$.
We derive $(r' \parallel u', s) \xrightarrow{a} (v' \parallel u', s')$. As $\forall s''((r', s'')R_p(v', s''))$ and $\forall s''((u, s'')R_q(u', s''))$, we have $\forall s''((v \parallel u, s'')R(v' \parallel u', s''))$ by definition of R.

$(u, s) \xrightarrow{a} (w, s')$, $r \equiv v$ and $a \neq \surd$. Likewise.

$(r,s) \xrightarrow{b} (v,s'')$, $(u,s) \xrightarrow{c} (w,s''')$, $a = \gamma(b,c)$ and $s' \in \mathit{effect}(a,s)$. In a similar way as above we can find v' and w' satisfying $(r',s) \xrightarrow{b} (v',s'')$ and $(u',s) \xrightarrow{c} (w',s''')$, and hence $(r' \parallel u',s) \xrightarrow{a} (v' \parallel w',s')$ for some s' in $\mathit{effect}(a,s)$. Furthermore we have $\forall s''((v,s'')R_p(v',s''))$ and $\forall s''((w,s'')R_q(w',s''))$. We conclude $\forall s''((v \parallel w,s'')R(v' \parallel w',s''))$.

$(r,s) \xrightarrow{\surd} (v,s')$, $(u,s) \xrightarrow{\surd} (w,s')$ and $a = \surd$. Likewise. □

Theorem 5.2.4 (Soundness) *Let p,q be closed terms over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$. If $\mathrm{ACP}_G + \mathrm{REC} + \mathrm{RSP} \vdash p = q$, then $p \leftrightarrows_{\mathcal{S}} q$ for any data environment $\mathcal{S}$.*

Proof. All the axioms of ACP_G, REC and RSP are sound and $\leftrightarrows_{\mathcal{S}}$ is a congruence. As an example we prove the soundness of the axiom EM1. Let $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ be a data environment over A and G_{at} and let p,q be closed over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$. Consider the relation

$$R \stackrel{\mathrm{def}}{=} Id \cup \{((p \parallel q,s),(p \between q + q \between p + p \mid q,s)) \mid s \in S\}$$

where Id is the identity relation on $C(\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}},S) \times C(\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}},S)$. It is not difficult to see that R is a global $\mathcal{S}$-bisimulation satisfying $(p \parallel q)R(p \between q + q \between p + p \mid q)$. □

With this result we immediately obtain the soundness of two-phase derivability.

Corollary 5.2.5 (Soundness) *Let p,q be closed terms over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$.*

1. *If $\mathrm{ACP}_G^4 + \mathrm{REC} + \mathrm{RSP} \vdash_c p = q$, then $p \leftrightarrows_{\mathcal{S}} q$ for any data environment $\mathcal{S}$.*
2. *Let $\mathcal{S}$ be a data environment such that weakest preconditions are expressible and that is sufficiently deterministic. If $\mathrm{ACP}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash_c p = q$, then $p \leftrightarrows_{\mathcal{S}} q$.* □

5.3 Completeness

We show that the axiom system ACP_G completely axiomatises global $\mathcal{S}$-bisimilarity in *all* data environments for the closed terms over $\Sigma(\mathrm{ACP}_G)$. From theorem 5.1.1 and lemma 3.3.6, it follows that we can restrict our attention to the G-basic and A-basic terms over $\Sigma(\mathrm{BPA}_G)$ defined in section 3. Due to the fact that global $\mathcal{S}$-bisimilarity is a finer equivalence than ordinary $\mathcal{S}$-bisimilarity, we are able to prove the related version of lemma 3.3.10 in a simple way. Note that the results from section 3 that are used here, are all proved using BPA_G^3.

Lemma 5.3.1 *If p_1,p_2 are G-basic terms over some reference set R and $\mathrm{ACP}_G \not\vdash p_1 = p_2$, then there is a data-state $\vec{\phi}$ in $\mathcal{S}(R)$ such that $(p_1,\vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (p_2,\vec{\phi})$.*

Proof. By induction on $|p_1|+|p_2|$. The case $|p_1|+|p_2| = 0$ is trivial, so assume $|p_1|+|p_2| > 0$. If $p_1 \neq p_2$, then $p_1 \not\subseteq p_2$ or $p_2 \not\subseteq p_1$. Assume $p_1 \not\subseteq p_2$, so there is an A-basic term q_1 over R such that $\vec{\phi}q_1 \sqsubseteq p_1$ and $\vec{\phi}q_1 \not\subseteq p_2$ (otherwise just sum up all syntactic summands of p_1 and conclude $p_1 \subseteq p_2$).

By definition p_2 has a syntactic summand $\vec{\phi}q_2$, but $q_1 \not\subseteq q_2$ (otherwise $\vec{\phi}q_1 \subseteq \vec{\phi}q_2 \subseteq p_2$). One of the following holds:

1. $\epsilon \sqsubseteq q_1$ and $\epsilon \not\sqsubseteq q_2$,

2. $ar \sqsubseteq q_1$ and $ar \not\sqsubseteq q_2$ for some $a \in A$ and G-basic term r.

(If all syntactic summands of q_1 would be *provable* summands of q_2, then $q_1 \subseteq q_2$.) In the first case we have $(p_1, \vec{\phi}) \xrightarrow{\surd} ...$, whereas by lemma 3.3.7 $(p_2, \vec{\phi})$ has no such transition, so $(p_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}} (p_2, \vec{\phi})$. We evaluate case 2:

either q_2 has no syntactic summands starting with a. Now $(p_1, \vec{\phi}) \not\leftrightarrows_{\mathcal{S}(R)} (p_2, \vec{\phi})$, for $(p_1, \vec{\phi})$ has an a-transition, whereas $(p_2, \vec{\phi})$ has no such transition by lemma 3.3.7;

or q_2 has $n+1$ syntactic summands starting with a, say $ar_0, ..., ar_n$. It holds that $r_i \neq r$ for all $i = 0, ..., n$ (otherwise $ar = ar_i \subseteq q_2$ for some i). By the induction hypothesis $(r, \vec{\psi_i}) \not\leftrightarrows_{\mathcal{S}(R)} (r_i, \vec{\psi_i})$ for a data-state $\vec{\psi_i} \in \mathcal{S}(R)$. By lemma 3.3.7 we have for all $i, j = 0, ..., n$ $(p_1, \vec{\phi}) \xrightarrow{a} (\epsilon r, \vec{\psi_i})$ and $(p_2, \vec{\phi}) \xrightarrow{a} (\epsilon r_j, \vec{\psi_i})$. Suppose $(p_1, \vec{\phi}) \leftrightarrows_{\mathcal{S}(R)} (p_2, \vec{\phi})$, then by definition of global $\mathcal{S}$-bisimilarity $(\epsilon r, \vec{\psi_i}) \leftrightarrows_{\mathcal{S}(R)} (\epsilon r_j, \vec{\psi_i})$ for all i, j, and hence $(r, \vec{\psi_i}) \leftrightarrows_{\mathcal{S}(R)} (r_j, \vec{\psi_i})$. But this was contradictory in case $i = j$.

The case $p_2 \not\sqsubseteq p_1$ can be treated likewise. □

By this lemma, the previous completeness results and theorem 5.1.1 we obtain the following results.

Corollary 5.3.2 (Completeness) *Let r_1, r_2 be closed terms over $\Sigma(\mathrm{ACP}_G)$.*

1. *If $r_1 \leftrightarrows_{\mathcal{S}} r_2$ for all data environments $\mathcal{S}$, then $\mathrm{ACP}_G \vdash r_1 = r_2$.*

2. *If $r_1 \leftrightarrows_{\mathcal{S}} r_2$ for all data environments $\mathcal{S}$, then $\mathrm{ACP}_G^4 \vdash_c r_1 = r_2$.*

3. *Let $\mathcal{S}$ be a data environment such that weakest preconditions are expressible and that is sufficiently deterministic. If $r_1 \leftrightarrows_{\mathcal{S}} r_2$, then $\mathrm{ACP}_G(\mathcal{S}) \vdash_c r_1 = r_2$.* □

5.4 An example: a parallel predicate checker

In this section we illustrate the techniques that we introduced up till now by an example. Let $f \subseteq \mathbb{N}$ be some predicate, eg. the set of all primes. Now, given some number n, we want to calculate the smallest $m \geq n$ such that $f(m)$. Assume we have two devices P_1 and P_2 that can calculate for some given number k whether $f(k)$ holds. In figure 1 we depict a system that enables us to calculate m using both P_1 and P_2. A Generator/Collector G generates numbers $n, n+1, n+2, ...$, sends them to P_1 and P_2, and collects their answers. Furthermore G selects the smallest number satisfying f from the answers and presents it to the environment.

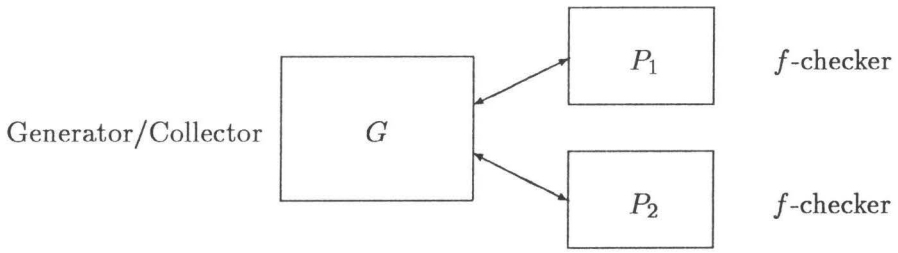


Figure 1: The parallel predicate checker Q

To describe this situation, we extend example 4.3 with the atomic actions ($i = 1, 2$):

$s(!x)$	send value of x,
$s_{ok}(!x_i)$	send the value x_i for which the evaluation of $f(x_i)$ was a succes,
s_{notok}	indicate that an evaluation of f was not succesful,
$r(?x_i)$	read a value for x_i,
$r_{ok}(?y)$	read a value for y for which $f(y)$ succeeded,
r_{notok}	read that an evaluation of f has failed,
c_{notok}	a communication between r_{notok} and s_{notok},
$w(!x),\ w(!y)$	write value of x, y to environment.

These atomic actions communicate according to the following scheme:

$$\gamma(s(!x), r(?x_i)) = \gamma(r(?x_i), s(!x)) = [x_i := x],$$
$$\gamma(s(!x_i), r(?y)) = \gamma(r(?y), s(!x_i)) = [y := x_i],$$
$$\gamma(s_{notok}, r_{notok}) = \gamma(r_{notok}, s_{notok}) = c_{notok}.$$

All new atomic actions do not change the data-state, i.e. for each new atomic action a:

$$\mathit{effect}(a, s) = \{s\}.$$

Probably, one would expect that for instance $\mathit{effect}(r(?y), s) = \{s[\mathit{new\ value}/y]\}$ as $r(?y)$ reads a new value for y. But this need not be so: the value of y is only changed if a communication takes place.

We add new atomic guards $f(x_i)$ to the setting of example 4.3. These guards have their obvious interpretation: $\mathit{test}(f(x_i), \rho)$ holds iff $f(\rho(x_i))$ holds.

The parallel predicate checker Q can now be specified by:

$$\begin{aligned}
G &= [x := n]\, s(!x)\, [x := x + 1]\, s(!x)\, G_1, \\
G_1 &= r_{notok}\, [x := x + 1]\, s(!x)\, G_1 + r_{ok}(?y)\, G_2, \\
G_2 &= \neg\langle x = y\rangle\, w(y) + \langle x = y\rangle (r_{ok}(?y)\, w(y) + r_{notok}\, w(x)),
\end{aligned}$$

$$\begin{aligned}
P_i &= r(?x)\, P_i' + \epsilon, \\
P_i' &= \langle f(x_i)\rangle s_{ok}(!x_i) + \neg\langle f(x_i)\rangle\, s_{notok}\, P_i + \epsilon,
\end{aligned}$$

$$Q = \partial_H(G \parallel P_1 \parallel P_2)$$

with $H = \{r(?x_i), r_{ok}(!y), r_{notok}, s(!x), s_{ok}(!x_i), s_{notok} \mid i = 1, 2\}$.

Q is correct if for an atomic action $w(x)$ (or $w(y)$), x (or y) contains the smallest number $m \geq n$ such that $f(m)$. We define the test $\alpha(x, y)$ to express this formally:

$$test(\alpha(x,y),\sigma) \iff \sigma(x) \leq \sigma(y) \wedge \Big(\bigwedge_{\substack{n \leq j < \sigma(y) \\ j \neq \sigma(x)}} \neg f(j) \Big).$$

Q is correct if we can prove that

$$\mathrm{ACP}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash_c Q = Q'$$

where Q' is defined by:

$$Q' = \delta_H(G' \parallel P_1 \parallel P_2)$$

with H, P_1 and P_2 as above and G':

$$\begin{aligned}
G' &= [x := n]\, s(!x)\, [x := x+1]\, s(!x)\, G'_1, \\
G'_1 &= r_{notok}\, [x := x+1]\, s(!x)\, G'_1 + r_{ok}(?y)\, G'_2, \\
G'_2 &= \neg\langle x = y\rangle \cdot \alpha(y,y)\langle f(y)\rangle \cdot w(y) \\
&\quad + \langle x = y\rangle (r_{ok}(?y) \cdot \alpha(y,y)\langle f(y)\rangle \cdot w(y) + r_{notok} \cdot \alpha(x,x)\langle f(x)\rangle \cdot w(x)).
\end{aligned}$$

Note that α is unnecessarily complex to state the correctness of Q. But this formulation is useful in the proof of $\mathrm{ACP}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash_c Q = Q'$.

This proof is given by first expanding Q and Q' to the *merge*-free forms R and R'. R is defined by:

$$R = [x := n]([x_1 := x]\, [x := x+1]\, [x_2 := x] + [x_2 := x]\, [x := x+1]\, [x_1 := x]) \cdot R_1$$

$$\begin{aligned}
R_1 = {} & \langle f(x_1)\rangle\, [y := x_1]\, R_2 + \\
& \langle f(x_2)\rangle\, [y := x_2]\, R_3 + \\
& (\neg\langle f(x_1)\rangle c_{notok}\, [x := x+1]\, [x_1 := x] + \\
& \neg\langle f(x_2)\rangle c_{notok}\, [x := x+1]\, [x_2 := x]) \cdot R_1
\end{aligned}$$

$$\begin{aligned}
R_2 = {} & \neg\langle x = y\rangle\, w(y) + \\
& \langle x = y\rangle (\neg\langle f(x_2)\rangle\, c_{notok}\, w(x) + \langle f(x_2)\rangle\, [y := x_2]\, w(y))
\end{aligned}$$

$$\begin{aligned}
R_3 = {} & \neg\langle x = y\rangle\, w(y) + \\
& \langle x = y\rangle (\neg\langle f(x_1)\rangle\, c_{notok}\, w(x) + \langle f(x_1)\rangle\, [y := x_1]\, w(y))
\end{aligned}$$

R' is defined likewise, except that $w(x)$ and $w(y)$ are replaced by $\langle \alpha(x,x)\rangle\, \langle f(x)\rangle\, w(x)$ and $\langle \alpha(y,y)\rangle\, \langle f(y)\rangle\, w(y)$, respectively. We state the following fact without proof.

Fact 1.

$$\mathrm{ACP}_G + \mathrm{REC} + \mathrm{RSP} \vdash Q = R \text{ and } \mathrm{ACP}_G + \mathrm{REC} + \mathrm{RSP} \vdash Q' = R'.$$

In order to show that $\mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash R = R'$ we need the following instances of SI, WPC1 and WPC2 in addition to those given in example 4.3. Let $t, u, ...$ be integer expressions containing the variables x_1, x_2, x, y and F some function on integer expressions.

$$
\begin{array}{l}
\phi\, c_{notok}\, \phi = \phi\, c_{notok} \text{ for all } \phi \in G \\
\neg\langle t = t\rangle = \delta \\
\langle t = u\rangle\, \neg\langle u = t\rangle = \delta \\
\langle t = u\rangle\, \langle u = v\rangle\, \neg\langle t = v\rangle = \delta \\
\langle t_1 = u_1\rangle \cdot ... \cdot \langle t_k = u_k\rangle\, \neg\langle F(t_1, ..., t_k) = F(u_1, ..., u_k)\rangle = \delta \\
\langle t + 1 = u\rangle\, \langle t = u\rangle = \delta \\
\neg\langle f(t)\rangle\, \langle \alpha(t, u)\rangle\, \neg\langle \alpha(u, u + 1)\rangle = \delta \\
\neg\langle f(t)\rangle\, \langle \alpha(u, t)\rangle\, \neg\langle \alpha(u, t + 1)\rangle = \delta \\
\langle \alpha(t, u - 1)\rangle\, \langle t = u\rangle = \delta
\end{array}
$$

Note that these identities are sound. We define

$$\beta(x, x_1, x_2) = \neg\langle x_1 = x_2\rangle(\langle \alpha(x_1, x_2)\rangle\, \langle x = x_2\rangle + \langle \alpha(x_2, x_1)\rangle\langle x = x_1\rangle).$$

It is easy to show that

$$R\ ,\ \beta(x, x_1, x_2)\, R_1\ ,\ \langle y = x_1\rangle\langle f(x_1)\rangle\beta(x, x_1, x_2)\, R_2\ ,\ \langle y = x_2\rangle\langle f(x_2)\rangle\beta(x, x_1, x_2)\, R_3$$

and

$$R'\ ,\ \beta(x, x_1, x_2)\, R_1'\ ,\ \langle y = x_1\rangle\langle f(x_1)\rangle\beta(x, x_1, x_2)\, R_2'\ ,\ \langle y = x_2\rangle\langle f(x_2)\rangle\beta(x, x_1, x_2)\, R_3'$$

are solutions for T, T_1, T_2 and T_3, respectively, in the following specification:

$$T = [x := n]([x_1 := x]\,[x := x + 1]\,[x_2 := x] + [x_2 := x]\,[x := x + 1]\,[x_1 := x]) \cdot T_1$$

$$
\begin{array}{rcl}
T_1 & = & \beta(x, x_1, x_2)\,(\langle f(x_1)\rangle\,[y := x_1]\,T_2 + \\
& & \quad \langle f(x_2)\rangle\,[y := x_2]\,T_3 + \\
& & \quad (\neg\langle f(x_1)\rangle c_{notok}\,[x := x + 1]\,[x_1 := x] + \\
& & \quad\ \neg\langle f(x_2)\rangle c_{notok}\,[x := x + 1]\,[x_2 := x]) \cdot T_1)
\end{array}
$$

$$
\begin{array}{rcl}
T_2 & = & \langle y = x_1\rangle\langle f(x_1)\rangle\beta(x, x_1, x_2)(\neg\langle x = y\rangle\, w(y) + \\
& & \quad \langle x = y\rangle(\neg\langle f(x_2)\rangle\, c_{notok}\, w(x) + \langle f(x_2)\rangle\,[y := x_2]\, w(y)))
\end{array}
$$

$$
\begin{array}{rcl}
T_3 & = & \langle y = x_2\rangle\langle f(x_2)\rangle\beta(x, x_1, x_2)(\neg\langle x = y\rangle\, w(y) + \\
& & \quad \langle x = y\rangle(\neg\langle f(x_1)\rangle\, c_{notok}\, w(x) + \langle f(x_1)\rangle\,[y := x_1]\, w(y)))
\end{array}
$$

Hence it follows by RSP that R and R' are equal. Concluding we have the following fact:

Fact 2.

$$\mathrm{ACP}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash_c Q = Q'.$$

6 Partial correctness and Hoare logic

In this section we show that we can capture Hoare logic for process terms [Pon89, PV89] in the algebraic framework developed thus far. We consider partial correctness formulas of the form $\{\alpha\}\, p\, \{\beta\}$, where p is a closed term over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$ and α, β are guards over $\Sigma(\mathrm{ACP}_G)$. It turns out that the validity of partial correctness formulas can be elegantly expressed with $\mathcal{S}$-bisimulation equivalence: $\{\alpha\}\, p\, \{\beta\}$ is valid in $\mathcal{S}$ iff $\alpha p \leftrightarrows_{\mathcal{S}} \alpha p \beta$. We further show a soundness result for a Hoare logic for linear processes over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ by translating proofs in Hoare logic into process algebra proofs.

6.1 Hoare logic for process terms

Hoare logic is meant for proving the correctness of programs. Proof systems are mostly given in a natural deduction format (see eg. [Dal83] for 'natural deduction') and are parameterised with

1. a class of *programs*, and
2. a language of *assertions* to express correctness properties of programs (usually some first-order language with equality).

In general a *partial correctness formula* has the syntax

$$\{pre\}\ P\ \{post\}$$

where $pre, post$ are assertions and P is a program. The intuitive meaning of $\{pre\}\ P\ \{post\}$ is that whenever the assertion pre holds *before* the execution of P and P terminates, then the assertion $post$ holds after the execution of P.

Given a set A of atomic actions and a set G_{at} of atomic guards, we here consider the guards over $\Sigma(\mathrm{ACP}_G)$ as a language of assertions, and we take the closed terms over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$ as the class of programs.

With respect to data-state transformations there are hardly any constraints on the way we provide process terms with an (operational) semantics. Therefore this instantiation is on a rather abstract level, and is suitable to express many programming primitives and constructs (cf. the examples in sections 4.3, 5.4 or Hoare's conditionals). We only consider data environments that are sufficiently deterministic and such that weakest preconditions are expressible. These are no serious restrictions that occur often in some related form in the study of Hoare logic (cf. [Bak80, Apt81]).

6.2 Partial correctness formulas and bisimulation

We now present formal definitions for the interpretation of partial correctness formulas and assertions in any data environment $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$. The main work is already done in section 5, where the operational semantics for the closed terms over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$ was defined. Let $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ be some data environment. In this section we use the transition relation $\longrightarrow_{\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}},\mathcal{S}}$ as defined in section 5.2 which is here simply written as $\longrightarrow$.

The interpretation of basic guards is such that a basic guard ϕ holds in $s \in S$ iff

$$(\phi, s) \xrightarrow{\surd} (\delta, s).$$

We define the interpretation of our assertions in $\mathcal{S}$ using $\surd$-transitions.

Definition 6.2.1 Let α be an assertion and $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test} \rangle$ some data environment.

1. The assertion α *holds in* $s \in S$, notation $\mathcal{S} \models \alpha[s]$, iff $(\alpha, s) \xrightarrow{\surd} (\delta, s)$.
2. The assertion α *is valid in* $\mathcal{S}$, notation $\mathcal{S} \models \alpha$, iff $\forall s \in S\ (\mathcal{S} \models \alpha[s])$.

□

In order to define the interpretation of partial correctness formulas , we introduce sequences of transitions. Let A^* be the set of finite strings over A, with typical elements $\sigma, \sigma', \ldots$ and λ denoting the empty string. We define for all $\sigma \in A^*$ relations $\xrightarrow{\sigma}\!\!\!\!\rightarrow$ and $\xrightarrow{\sigma\surd}\!\!\!\!\rightarrow$ that describe sequences of transitions:

- $(x, s) \xrightarrow{\lambda}\!\!\!\!\rightarrow (x, s)$
- $$\frac{(x, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (x', s') \quad (x', s') \xrightarrow{a} (x'', s'')}{(x, s) \xrightarrow{\sigma a}\!\!\!\!\rightarrow (x'', s'')} \qquad (a \in A_{\surd})$$

Now the interpretation of a partial correctness formula in $\mathcal{S}$ is defined as follows:

Definition 6.2.2 A partial correctness formula $\{\alpha\}p\{\beta\}$ *is valid in* $\mathcal{S}$, notation $\mathcal{S} \models \{\alpha\}p\{\beta\}$, iff for all $s \in S$ and all $\sigma \in A^*$:

$$\mathcal{S} \models \alpha[s] \text{ and } (p, s) \xrightarrow{\sigma\surd}\!\!\!\!\rightarrow (p', s') \implies \mathcal{S} \models \beta[s'].$$

□

We show that for any partial correctness formula $\{\alpha\}\ p\ \{\beta\}$ it holds that $\mathcal{S} \models \{\alpha\}\ p\ \{\beta\}$ iff $\alpha p \leftrightarrows_{\mathcal{S}} \alpha p \beta$. This alternative characterisation of validity of partial correctness formulas gives us the means to use process algebra for proving partial correctness formulas.

Lemma 6.2.3 (Decomposition) *Let $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test} \rangle$ be some data environment. For any closed term p over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$, guard α over $\Sigma(\mathrm{ACP}_G)$ and $\sigma \in (A \cup \{\surd\})^*$ the following properties hold:*

1. *If $(\alpha p, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (p', s')$ and $\sigma \not\equiv \lambda$, then $(\alpha, s) \xrightarrow{\surd} (\delta, s)$ and $(p, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (p', s')$.*
2. *If $(p\alpha, s) \xrightarrow{\sigma\surd}\!\!\!\!\rightarrow (p', s')$, then $(p, s) \xrightarrow{\sigma\surd}\!\!\!\!\rightarrow (p', s')$ and $(\alpha, s') \xrightarrow{\surd} (\delta, s')$.*

Proof. By induction on the length of σ (first proving some intermediate properties of sequences of non-terminating transitions). □

Lemma 6.2.4 *Let p be a closed term over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$, α some guard over $\Sigma(\mathrm{ACP}_G)$ and $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ a data environment. Then the following statements are equivalent:*

(i). For all $s \in S$ and $\sigma \in A^*$ it holds that $(p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s') \implies (\alpha,s') \xrightarrow{\surd} (\delta,s')$.

(ii). $p \leftrightarrows_{\mathcal{S}} p\alpha$.

Proof. First observe that if $(p,s) \xrightarrow{\surd} (p',s')$, then $p' \equiv \delta$ and $s' \equiv s$.

$(i) \implies (ii)$. Fix some $s \in S$ and take

$$R \stackrel{\text{def}}{=} \{((\delta,s'),(\delta,s')) \mid s' \in S\} \cup \{((r,s'),(r\alpha,s')) \mid (p,s) \xrightarrow{\sigma} (r,s') \text{ for some } \sigma \in A^*\}$$

Note that $(p,s)R(p\alpha,s)$. We show that R is an $\mathcal{S}$-bisimulation. For pairs $((\delta,s'),(\delta,s'))$ it is trivial to check the transfer property. Assume $(q,s')R(q\alpha,s')$.

- Suppose $(q,s') \xrightarrow{a} (q',s'')$ with $a \in A$. We derive $(q\alpha,s') \xrightarrow{a} (q'\alpha,s'')$ and by definition of R also $(q',s'')R(q'\alpha,s'')$.
- Suppose $(q,s') \xrightarrow{\surd} (\delta,s')$, so $(p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (\delta,s')$ for some σ. By assumption we have $(\alpha,s') \xrightarrow{\surd} (\delta,s')$ and we derive $(q\alpha,s') \xrightarrow{\surd} (\delta,s')$. By definition $(\delta,s')R(\delta,s')$.
- Suppose $(q\alpha,s') \xrightarrow{a} (q',s'')$ with $a \in A$. By 'decomposition' it follows that $q' \equiv q''\alpha$ and $(q,s') \xrightarrow{a} (q'',s'')$. By definition $(q'',s'')R(q',s'')$.
- Suppose $(q\alpha,s') \xrightarrow{\surd} (\delta,s')$. It follows that $(q,s') \xrightarrow{\surd} (\delta,s')$ and $(\alpha,s') \xrightarrow{\surd} (\delta,s')$. By definition $(\delta,s')R(\delta,s')$.

$(ii) \implies (i)$. Suppose $(p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s')$ for some $\sigma \in A^*$. By assumption then also $(p\alpha,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s')$, and by decomposition we have $(\alpha,s') \xrightarrow{\surd} (\delta,s')$.

□

Now we can easily prove the following characterisation of the $\mathcal{S}$-validity of partial correctness formulas in terms of $\mathcal{S}$-bisimilarity.

Theorem 6.2.5 *Let p be a closed term over $\Sigma(\mathrm{ACP}_G)_{\mathrm{REC}}$, α,β guards over $\Sigma(\mathrm{ACP}_G)$ and $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ a data environment. Then*

$$\mathcal{S} \models \{\alpha\}\, p\, \{\beta\} \iff \alpha p \leftrightarrows_{\mathcal{S}} \alpha p\beta.$$

Proof.

$\implies$ Suppose $\mathcal{S} \models \{\alpha\}p\{\beta\}$. By the previous lemma it is sufficient to show that if $(\alpha p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s')$, then $(\beta,s') \xrightarrow{\surd} (\delta,s')$. So let $(\alpha p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s')$. By 'decomposition' we have $(\alpha,s) \xrightarrow{\surd} (p',s)$ and $(p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s')$. By $\mathcal{S} \models \{\alpha\}\, p\, \{\beta\}$ this implies $(\beta,s') \xrightarrow{\surd} (p',s')$.

$\impliedby$ Suppose $\mathcal{S} \not\models \{\alpha\}\, p\, \{\beta\}$, so for some $s \in S$ and $\sigma \in A^*$:

$$(\alpha,s) \xrightarrow{\surd} (\delta,s) \text{ and } (p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s') \text{ and } \mathcal{S} \not\models \beta[s'].$$

We derive $(\alpha p,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s')$ and by $\mathcal{S}$-bisimilarity we have $(\alpha p\beta,s) \xrightarrow{\sigma\surd\!\!\!\!\rightarrow} (p',s')$. By 'decomposition' this implies that $(\beta,s') \xrightarrow{\surd} (\delta,s')$, which contradicts the supposition.

□

6.3 A proof system for deriving partial correctness formulas

In this section we present a proof system H in a natural deduction format for deriving partial correctness formulas over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ (cf. [Pon89, PV89]). The proof system H is displayed in table 13. Notice that the rules of H refer to terms over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ that need not be closed. Let Γ be a set of assertions and partial correctness formulas. We write $\Gamma \vdash_H \{\alpha\}\, t\, \{\beta\}$ iff we can derive $\{\alpha\}\, t\, \{\beta\}$ in H using elements of Γ as axioms.

- The axiom scheme I only makes sense if weakest preconditions are expressible, and it is only valid in data environments that are sufficiently deterministic. Weakest preconditions are defined in definition 4.1.1 and remark 4.1.4.
- The axiom scheme II introduces partial correctness formulas for basic guards.
- Rules III and IV express how the operators $+$ and $\cdot$ may be introduced in partial correctness formulas.
- Rule V, *consequence*, is a standard proof rule in Hoare logic. The intended interpretation of an expression $\alpha \to \beta$ is as expected: $\mathcal{S} \models (\alpha \to \beta)[s]$ iff $\mathcal{S} \models \alpha[s] \Longrightarrow \mathcal{S} \models \beta[s]$.
- Rule VI, an instance of *Scott's induction rule* (see eg. [Bak80, Apt81]), is suitable to derive partial correctness formulas with recursive terms over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$. This rule allows *cancellation* of hypotheses, indicated by the square brackets in its premises: let $E = \{x = t_x \mid x \in V_E\}$ be a guarded recursive specification and α_x, β_x $(x \in V_E)$ be guards. If for all $y \in V_E$ we can derive (indicated by the dots in the rule) $\{\alpha_y\}\, t_y\, \{\beta_y\}$ from a set of hypotheses Γ_y containing *no* other partial correctness formulas with free variables in V_E than those in $\{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}$, then for any $z \in V_E$ the partial correctness formula $\{\alpha_z\} <z \mid E> \{\beta_z\}$ can be derived from

$$\bigcup_{x \in V_E} \Gamma_x - \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}.$$

6.4 Soundness of the proof system

In this section we prove a soundness result for H with respect to a data environment $\mathcal{S} = \langle S, \mathit{effect}, \mathit{test}\rangle$ over A and G such that weakest preconditions are expressible and $\mathcal{S}$ is sufficiently deterministic. Let $Tr_{\mathcal{S}}$ be the set of assertions that are *true* (valid) in $\mathcal{S}$. We prove that

$$Tr_{\mathcal{S}} \vdash_H \{\alpha\}\, p\, \{\beta\} \quad \Longrightarrow \quad \mathcal{S} \models \{\alpha\}\, p\, \{\beta\}$$

provided that recursive specifications have a finite number of equations and are *linear* (cf. *linear* context free grammars [HU79]):

Definition 6.4.1 A process term t over $\Sigma(\mathrm{BPA}_G)$ is called *linear* over $V' \subseteq V$ iff

$$t ::= p \mid x \mid pt \mid tp \mid t + t$$

where p is a closed term over $\Sigma(\mathrm{BPA}_G)$ and $x \in V'$. A recursive specification $E = \{x = t_x \mid x \in V_E\}$ is *linear* whenever the terms t_x are linear over V_E. □

I	*axioms for* $a \in A$	$\{wp(a,\alpha)\}\ a\ \{\alpha\}$
II	*axioms for* $\phi \in G$	$\{\alpha\}\ \phi\ \{\alpha \cdot \phi\}$
III	*alternative composition*	$\dfrac{\{\alpha\}\ t\ \{\beta\} \quad \{\alpha\}\ t'\ \{\beta\}}{\{\alpha\}\ t + t'\ \{\beta\}}$
IV	*sequential composition*	$\dfrac{\{\alpha\}\ t\ \{\alpha'\} \quad \{\alpha'\}\ t'\ \{\beta\}}{\{\alpha\}\ t \cdot t'\ \{\beta\}}$
V	*consequence*	$\dfrac{\alpha \to \alpha' \quad \{\alpha'\}\ t\ \{\beta'\} \quad \beta' \to \beta}{\{\alpha\}\ t\ \{\beta\}}$
VI	*Scott's induction rule*	For $E = \{x = t_x \mid x \in V_E\}$ a *guarded* recursive specification

$$\frac{\begin{array}{c}[\{\{\alpha_x\}\ x\ \{\beta_x\} \mid x \in V_E\}] \\ \vdots \\ \{\alpha_y\}\ t_y\ \{\beta_y\} \qquad \text{for all } y \in V_E\end{array}}{\{\alpha_z\} <z \mid E> \{\beta_z\}} \quad z \in V_E$$

Table 13: The proof system H $(a \in A, \phi \in G)$

In [Pon89, PV89] only processes definable by regular recursion were considered in the context of Scott's induction rule. This class is contained in the class of processes definable by guarded, linear recursion.

By lemma 6.2.4 and the soundness of $\mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP}$, the soundness of H follows from the statement

$$Tr_{\mathcal{S}} \vdash_H \{\alpha\}\, p\, \{\beta\} \implies \mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash \alpha p = \alpha p \beta.$$

In the rest of this section we prove this statement by translating H-derivations in a canonical way to proofs in process algebra.

We first show that H is sound for the (recursion-free) terms over $\Sigma(\mathrm{BPA}_G)$.

Lemma 6.4.2 (Soundness of H for recursion-free terms) *Let p be a closed term over $\Sigma(\mathrm{BPA}_G)$ and α, β guards over $\Sigma(\mathrm{BPA}_G)$. Then*

$$Tr_{\mathcal{S}} \vdash_H \{\alpha\}\, p\, \{\beta\} \implies \mathrm{BPA}_G(\mathcal{S}) \vdash \alpha p = \alpha p \beta.$$

Proof. By induction on the length of derivations. The soundness of I – IV is straightforward. We only show that rule V (*consequence*) is sound (we need not consider rule VI, as this rule introduces recursively defined processes). Rule V contains expressions of the form $\alpha \to \beta$ with the interpretation $\mathcal{S} \models (\alpha \to \beta)[s]$ iff $\mathcal{S} \models \alpha[s] \implies \mathcal{S} \models \beta[s]$. It is easy to show that such expressions can be algebraically characterised as follows:

$$\alpha \to \beta \in Tr_{\mathcal{S}} \iff \mathrm{BPA}_G(\mathcal{S}) \vdash \alpha \cdot \beta = \alpha.$$

Assume

$$Tr_{\mathcal{S}} \vdash_H \{\alpha'\}\, p\, \{\beta'\} \text{ and } \alpha \to \alpha', \beta' \to \beta \in Tr_{\mathcal{S}}.$$

By induction we can prove $\alpha' p = \alpha' p \beta'$, $\alpha\alpha' = \alpha$ and $\beta'\beta = \beta'$ in $\mathrm{BPA}_G(\mathcal{S})$. We derive

$$\begin{aligned} \alpha p &= \alpha\alpha' p \\ &= \alpha\alpha' p \beta' \\ &= \alpha\alpha' p \beta' \beta \\ &= \alpha\alpha' p \beta \\ &= \alpha p \beta \end{aligned}$$

as was to be shown. □

Using this fact we can prove a general result concerning *linear* terms that connects H-derivability under $Tr_{\mathcal{S}}$ to provable equality in $\mathrm{BPA}_G(\mathcal{S})$.

Lemma 6.4.3 *Let $t(x_1, ..., x_n)$ be a term over $\Sigma(\mathrm{BPA}_G)$ and $\alpha, \beta, \alpha_i, \beta_i$ be guards over $\Sigma(\mathrm{BPA}_G)$ for $i = 1, ..., n$. If $t(x_1, ..., x_n)$ is linear over $\{x_1, ..., x_n\}$, and*

$$Tr_{\mathcal{S}}, \{\{\alpha_i\}\, x_i\, \{\beta_i\} \mid i = 1, ..., n\} \vdash_H \{\alpha\}\, t(x_1, ..., x_n)\, \{\beta\},$$

then

1. $\mathrm{BPA}_G(\mathcal{S}) \vdash \alpha \cdot t(\alpha_1 x_1, ..., \alpha_n x_n) = \alpha \cdot t(x_1, ..., x_n)$,

2. $\mathrm{BPA}_G(\mathcal{S}) \vdash \alpha \cdot t(x_1\beta_1, ..., x_n\beta_n) = \alpha \cdot t(x_1\beta_1, ..., x_n\beta_n) \cdot \beta$.

Proof. By induction on the length of the derivation of

$$Tr_{\mathcal{S}}, \{\{\alpha_i\}\ x_i\ \{\beta_i\} \mid i = 1, ..., n\} \vdash_H \{\alpha\}\ t(x_1, ..., x_n)\ \{\beta\}.$$

The cases in which one of I – III is applied last are straightforward. We give a proof for the cases in which IV or V is applied last (note that by definition of linearity rule VI of H again need not be considered):

ad IV. Because all terms in the proof are linear, we may assume that $t(x_1, ..., x_n) \equiv p \cdot u(x_1, ..., x_n)$ or $t(x_1, ..., x_n) \equiv u(x_1, ..., x_n) \cdot p$, with p a closed term over $\Sigma(\mathrm{BPA}_G)$. Let $t(x_1, ..., x_n) \equiv p \cdot u(x_1, ..., x_n)$ and

$$\frac{\begin{array}{c} Tr_{\mathcal{S}}, \{\{\alpha_i\}\ x_i\ \{\beta_i\} \mid i = 1, ..., n\} \\ \vdots \\ \{\alpha\}\ p\ \{\alpha'\} \quad \{\alpha'\}\ u(x_1, ..., x_n)\ \{\beta\} \end{array}}{\{\alpha\}\ p \cdot u(x_1, ..., x_n)\ \{\beta\}}$$

Apparently $Tr_{\mathcal{S}} \vdash_H \{\alpha\}\ p\ \{\alpha'\}$, so we have by lemma 6.4.2 that $\mathrm{BPA}_G(\mathcal{S}) \vdash \alpha p = \alpha p \alpha'$. We derive

$$\begin{array}{lrcl} 1. & \alpha p \cdot u(\alpha_1 x_1, ..., \alpha_n x_n) & = & \alpha p \alpha' \cdot u(\alpha_1 x_1, ..., \alpha_n x_n) \\ & & \overset{IH}{=} & \alpha p \alpha' \cdot u(x_1, ..., x_n) \\ & & = & \alpha p \cdot u(x_1, ..., x_n). \\ & & & \\ 2. & \alpha p \cdot u(x_1\beta_1, ..., x_n\beta_n) & = & \alpha p \alpha' \cdot u(x_1\beta_1, ..., x_n\beta_n) \\ & & \overset{IH}{=} & \alpha p \alpha' \cdot u(x_1\beta_1, ..., x_n\beta_n) \cdot \beta \\ & & = & \alpha p \cdot u(x_1\beta_1, ..., x_n\beta_n) \cdot \beta. \end{array}$$

The case in which $t(x_1, ..., x_n) \equiv u(x_1, ..., x_n) \cdot p$ with p a closed term over $\Sigma(\mathrm{BPA}_G)$ can be proved likewise.

ad V. Assume

$$\frac{\begin{array}{c} Tr_{\mathcal{S}}, \{\{\alpha_i\}\ x_i\ \{\beta_i\} \mid i = 1, ..., n\} \\ \vdots \\ \alpha \to \alpha' \quad \{\alpha'\}\ t(x_1, ..., x_n)\ \{\beta'\} \quad \beta' \to \beta \end{array}}{\{\alpha\}\ t(x_1, ..., x_n)\ \{\beta\}}$$

By induction we have $\mathrm{BPA}_G(\mathcal{S})$-derivations of $\alpha\alpha' = \alpha$ and $\beta'\beta = \beta'$. We derive

$$\begin{array}{lrcl}
1. & \alpha \cdot t(\alpha_1 x_1, ..., \alpha_n x_n) & = & \alpha\alpha' \cdot t(\alpha_1 x_1, ..., \alpha_n x_n) \\
& & \stackrel{IH}{=} & \alpha\alpha' \cdot t(x_1, ..., x_n) \\
& & = & \alpha \cdot t(x_1, ..., x_n).
\end{array}$$

$$\begin{array}{lrcl}
2. & \alpha \cdot t(x_1\beta_1, ..., x_n\beta_n) & = & \alpha\alpha' \cdot t(x_1\beta_1, ..., x_n\beta_n) \\
& & \stackrel{IH}{=} & \alpha\alpha' \cdot t(x_1\beta_1, ..., x_n\beta_n) \cdot \beta' \\
& & = & \alpha\alpha' \cdot t(x_1\beta_1, ..., x_n\beta_n) \cdot \beta'\beta \\
& & \stackrel{IH}{=} & \alpha\alpha' \cdot t(x_1\beta_1, ..., x_n\beta_n) \cdot \beta \\
& & = & \alpha \cdot t(x_1\beta_1, ..., x_n\beta_n) \cdot \beta.
\end{array}$$

□

This result can be used to show the soundness of the proof system H for the following subset of terms over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$.

Theorem 6.4.4 (Soundness of H) *Let p be a closed term over $\Sigma(\mathrm{BPA}_G)_{\mathrm{REC}}$ in which all occurrences of the form $<x \mid E>$ refer to a (guarded) recursive specification E over $\Sigma(\mathrm{BPA}_G)$ that is linear and contains only finitely many equations. Let α, β be guards over $\Sigma(\mathrm{BPA}_G)$. Then*

$$\begin{array}{rcl}
Tr_{\mathcal{S}} \vdash_H \{\alpha\}\, p\, \{\beta\} & \Longrightarrow & \mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash \alpha p = \alpha p\beta \\
& \Longrightarrow & \alpha p \leftrightarrows_{\mathcal{S}} \alpha p\beta \\
& \Longleftrightarrow & \mathcal{S} \models \{\alpha\}\, p\, \{\beta\}.
\end{array}$$

Proof. By theorems 4.2.1 and 6.2.5 we only have to prove the first implication. We apply induction on the length of H-derivations. The proof of the soundness of I – V is straightforward (cf. the proof of lemma 6.4.2). We only give a proof of the soundness of VI. Let $E = \{x_i = t_i(x_1, ..., x_n) \mid i = 1, ..., n\}$ be a guarded linear recursive specification and assume

$$Tr_{\mathcal{S}}, \{\{\alpha_i\}\, x_i\, \{\beta_i\} \mid i = 1, ..., n\} \vdash_H \{\alpha_j\}\, t_j(x_1, ..., x_n)\, \{\beta_j\}$$

for $j = 1, ..., n$. So we have an H-derivation of the premises of Scott's rule. We have to show

$$\mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash \alpha_j X_j = \alpha_j X_j \beta_j$$

for $j = 1, ..., n$. In order to do so we use the recursive specifications

$$E' = \{y_i = \alpha_i \cdot t_i(y_1, ..., y_n) \mid i = 1, ..., n\}$$

$$E'' = \{z_i = \alpha_i \cdot t_i(z_1\beta_1, ..., z_n\beta_n) \mid i = 1, ..., n\}$$

and show for any $j \in \{1, ..., n\}$ that

1. $\alpha_j X_j = Y_j$,
2. $Z_j\beta_j = Z_j$.

3. $Z_j = Y_j$.

As a consequence we can derive

$$\mathrm{BPA}_G(\mathcal{S}) + \mathrm{REC} + \mathrm{RSP} \vdash \alpha_j X_j = Y_j = Z_j = Z_j\beta_j = \alpha_j X_j \beta_j$$

as has to be shown. So we are left to prove 1,2 and 3. Observe that E', E'' are guarded linear recursive specifications, so we may use both RSP and the previous lemma 6.4.3.

ad 1. We first show that $\alpha_j X_j = Y_j$ for all $j \in \{1, ..., n\}$. We derive

$$\alpha_j X_j \overset{\mathrm{REC}}{=} \alpha_j \cdot t_j(X_1, ..., X_n) \overset{6.4.3.1}{=} \alpha_j \cdot t_j(\alpha_1 X_1, ..., \alpha_n X_n).$$

So $\alpha_1 X_1, ..., \alpha_n X_n$ are solutions of $y_1, ..., y_n$ in E'. With RSP we conclude $\alpha_j X_j = Y_j$ for $j = 1, ..., n$.

ad 2. We show that $Z_j\beta_j = Z_j$ for all $j \in \{1, ..., n\}$. We derive

$$\begin{array}{rcl} Z_j\beta_j & \overset{\mathrm{REC}}{=} & \alpha_j \cdot t_j(Z_1\beta_1, ..., Z_n\beta_n) \cdot \beta_j \\ & \overset{6.4.3.2}{=} & \alpha_j \cdot t_j(Z_1\beta_1, ..., Z_n\beta_n) \\ & = & \alpha_j \cdot t_j((Z_1\beta_1)\beta_1, ..., (Z_n\beta_n)\beta_n). \end{array}$$

So $Z_1\beta_1, ..., Z_n\beta_n$ are solutions of $z_1, ..., z_n$ in E''. With RSP we conclude $Z_j\beta_j = Z_j$ for all $j \in \{1, ..., n\}$.

ad 3. We show (using 2) $Z_j = Y_j$ for all $j \in \{1, ..., n\}$ as follows:

$$Z_j \overset{\mathrm{REC}}{=} \alpha_j \cdot t_j(Z_1\beta_1, ..., Z_n\beta_n) \overset{2}{=} \alpha_j \cdot t_j(Z_1, ..., Z_n).$$

So $Z_1, ..., Z_n$ are solutions of $y_1, ..., y_n$ in E'. With RSP we conclude $Z_j = Y_j$ for all $j \in \{1, ..., n\}$. □

References

[AB84] D. Austry and G. Boudol. Algèbre de processus et synchronisations. *Theoretical Computer Science*, 30(1):91–131, 1984.

[Apt81] K.R. Apt. Ten years of Hoare's logic: a survey – Part I. *ACM Transactions on Programming Languages and Systems*, 3(4):431–483, 1981.

[Apt84] K.R. Apt. Ten years of Hoare's logic: a survey – Part II; Nondeterminism. *Theoretical Computer Science*, 28:83–109, 1984.

[Bak80] J.W. de Bakker. *Mathematical theory of program correctness.* Prentice Hall International, 1980.

[BB88] J.C.M. Baeten and J.A. Bergstra. Global renaming operators in concrete process algebra. *Information and Computation*, 78(3):205–245, 1988.

[BG87] J.C.M. Baeten and R.J. van Glabbeek. Merge and termination in process algebra. In K.V. Nori, editor, *Proceedings 7^{th} Conference on Foundations of Software Technology and Theoretical Computer Science,* Pune, India, volume 287 of *Lecture Notes in Computer Science*, pages 153–172. Springer-Verlag, 1987.

[BK84a] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In J. Paredaens, editor, *Proceedings 11^{th} ICALP,* Antwerp, volume 172 of *Lecture Notes in Computer Science*, pages 82–95. Springer-Verlag, 1984.

[BK84b] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.

[BK86] J.A. Bergstra and J.W. Klop. Verification of an alternating bit protocol by means of process algebra. In W. Bibel and K.P. Jantke, editors, *Math. Methods of Spec. and Synthesis of Software Systems '85, Math. Research 31*, pages 9–23, Berlin, 1986. Akademie-Verlag. First appeared as: Report CS-R8404, CWI, Amsterdam, 1984.

[BKT85] J.A. Bergstra, J.W. Klop, and J.V. Tucker. Process algebra with asynchronous communication mechanisms. In S.D. Brookes, A.W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science*, pages 76–95. Springer-Verlag, 1985.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[Dal83] D. van Dalen. *Logic and Structure.* Springer-Verlag, 1983.

[Dij76] E.W. Dijkstra. *A Discipline of Programming.* Prentice Hall International, Englewood Cliff, 1976.

[Gla90] R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer-Verlag, 1990.

[GV89] R.J. van Glabbeek and F.W. Vaandrager. Modular specifications in process algebra – with curious queues (extended abstract). In M. Wirsing and J.A. Bergstra, editors, *Algebraic Methods: Theory, Tools and Applications, Workshop Passau 1987*, volume 394 of *Lecture Notes in Computer Science*, pages 465–506. Springer-Verlag, 1989.

[HHJ+87] C.A.R. Hoare, I.J. Hayes, He Jifeng, C.C. Morgan, A.W. Roscoe, J.W. Sanders, I.H. Sorensen, J.M. Spivey, and B.A. Sufrin. Laws of programming. *Communications of the ACM*, 30(8):672–686, August 1987.

[Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10), October 1969.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[ISO87] ISO. *Information processing systems – open systems interconnection – LOTOS – a formal description technique based on the temporal ordering of observational behaviour*, 1987. ISO/TC97/SC21/N DIS8807.

[KV85] C.P.J. Koymans and J.L.M. Vrancken. Extending process algebra with the empty process ϵ. Logic Group Preprint Series Nr. 1, CIF, State University of Utrecht, 1985.

[Lam80] L. Lamport. The 'Hoare logic' of concurrent programs. *Acta Informatica*, 14:21–37, 1980.

[MA86] E.G. Manes and M.A. Arbib. *Algebraic Approaches to Program Semantics.* Texts and Monographs in Computer Science. Springer-Verlag, 1986.

[Man74] Z. Manna. *Mathematical Theory of Computation.* McGraw-Hill Book Co., 1974.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and concurrency.* Prentice Hall International, 1989.

[OG76] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, pages 319–340, 1976.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pon89] A. Ponse. Process expressions and Hoare's logic. Report CS-R8905, CWI, Amsterdam, March 1989. To appear in *Information and Computation.*

[PV89] A. Ponse and F.J. de Vries. Strong completeness for Hoare logics of recursive processes: an infinitary approach. Report CS-R8957, CWI, Amsterdam, 1989.

[Sio64] F.M. Sioson. Equational bases of Boolean algebras. *Journal of Symbolic Logic*, 29(3):115–124, September 1964.

[Ss90] SPECS-semantics. *Definition of MR and CRL Version 2.1*, 1990.

[Sti88] C. Stirling. A generalization of Owicki-Gries's Hoare logic for a concurrent while-language. *Theoretical Computer Science*, 58:34–359, 1988.

[Vaa89] F.W. Vaandrager. Specificatie en verificatie van communicatieprotocollen met procesalgebra. Dept. of Computer Science, University of Amsterdam, 1989. Lecture notes, in Dutch.

[Vra86] J.L.M. Vrancken. The algebra of communicating processes with empty process. Report FVI 86-01, Dept. of Computer Science, University of Amsterdam, 1986.