Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

A. Ponse

Process expressions and Hoare's logic

Computer Science/Department of Software Technology Report CS-R8905 March,

---

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.).

Copyright © Stichting Mathematisch Centrum, Amsterdam

Process expressions and Hoare's logic

Alban Ponse
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract

In this paper processes specifiable over a non-uniform language are considered. The language contains constants for a set of atomic actions and constructs for alternative composition and sequential composition. Furthermore it provides a mechanism for specifying processes by a form of guarded recursion (including nested applications). We consider processes as having side-effects: Atomic actions are to be specified in terms of observability and state transformations. The execution of a process having some initial state is formally described as a transition system containing information about observability and state transformations. This leads to an operational semantics in the style of Plotkin. Let S be a set of states, α and β denote unary predicates over S and p be the specification of some process. The *partial correctness assertion* $\{\alpha\}\, p\, \{\beta\}$ expresses that for any transition system associated with p and having an initial state satisfying α, its final states satisfy β. A logic in the style of Hoare and a proof system for deriving partial correctness assertions are presented. This proof system is complete (relative to all true assertions about the states in S), so any partial correctness assertion can be evaluated by investigating its derivability. Techniques concerning the construction of derivations or proofs of their absence are not discussed.

Key Words & Phrases: process expression, process algebra, side-effects, Hoare's logic, partial correctness assertions.
1985 Mathematics Subject Classification: 68Q55, 68Q60.
1980 Mathematics Subject Classification: 68B10, 68F20.
1982 CR Categories: D.3.1, F.3.1, F.3.2.
Note: The author received full support from the European Communities under ESPRIT project no. 432, An Integrated Formal Approach to Industrial Software Development (METEOR).

1 Introduction

A *process* is the behaviour of a system. A computer executing some program or a person using a drink dispenser are two examples of processes. In order to specify (or analyse) processes we assume that we have available a set of *atomic actions*, i.e. processes which are considered to be not divisible into smaller parts and not subject to further investigation. More complex processes can be seen as composed out of atomic actions. This paper is based on the notion of processes specifiable in a simple 'process language' for which a set A of atomic actions is a parameter. The language contains constants for all atomic actions in A and offers constructs for alternative composition, sequential composition and recursion. Any closed *process expression* represents a process. An atomic action $a \in A$ is considered to be observable pointwise in time. Its *execution* though takes a positive (finite) amount of time, at the beginning of which it can be observed as the process a, and at the end of which we say that the execution is *terminated successfully*. This can be formally described as a *(labelled) transition* $a \xrightarrow{a} \surd$, where the label a represents what can be observed and $\surd$ is a symbol representing successful termination (see [9]). By giving a calculus for deriving transitions (so called action rules), every closed process expression generates a transition system, representing all possible courses of execution. In [6] this is worked out for a larger process language, containing also a construct for the specification of concurrent processes.

In this paper we regard the use of a *state space* on which atomic actions, and thus all processes specifiable in our language, have side-effects involving state transformations. This means that we consider a *non-uniform* process language (this notion is eg. discussed in [4]). These side-effects are to be specified in terms of observability and state transformations by functions *action* and *effect* respectively: Let S be a set of states and a the representation of some atomic action. The expression $action(a, s)$ denotes what can be observed if a is executed in *initial* state s; with $effect(s, a)$ we denote the state resulting from this execution. If this execution terminates successfully it can be formally described by a transition

$$(a, s) \xrightarrow{action(a,s)} (\surd, effect(s, a))$$

where (a, s) represents the process a in initial state s and $effect(s, a)$ is called a *final* state. We present a calculus to derive transition systems concerning more complex process expressions out of these atomic transitions (so called effect rules), reflecting the observability and state transformations of any possible execution. We will use these transition systems to define an operational semantics.

Most of this paper is about *partial correctness assertions*. Let α and β refer to unary predicates over S and p represent some process. The partial correctness assertion $\{\alpha\}p\{\beta\}$ concerns *some* features of a number of transition systems associated with p: If the execution of p started in an initial state satisfying α terminates successfully, then the resulting final state satisfies β. So a partial correctness assertion abstracts from observability and intermediate states of execution. The adjective *partial* expresses that successful termination is not implied. The transition (system) displayed above expresses that the partial correctness assertion "$\{s\}\ a\ \{effect(s, a)\}$" is *true*. A partial correctness assertion $\{\alpha\}\, a\, \{\beta\}$ over an atomic action a can be evaluated by constructing the transitions $(a, s) \longrightarrow (\surd, .)$ for all s such that $\alpha(s)$ holds.

Main result. In order to reason formally about partial correctness assertions we present an application of *Hoare's logic*, an axiomatic method for proving programs correct (for a survey of Hoare's logic see [1]). This logic contains a proof system for deriving partial correctness assertions starting from atomic partial correctness assertions and the true assertions about the states in S. We show that the proof system is complete (relative to all true assertions about the states in S), so a partial correctness assertion is true iff it is derivable. Suppose one wants to investigate a partial correctness assertion concerning a complex process specification. Finding a right derivation or showing the absence of these may then replace the construction of a (possibly large) number of (complex) transition systems. Techniques concerning the construction of derivations or proofs of their absence are not discussed.

Contents of the paper. Section 2 is dedicated to the way processes can be specified. In section 3 we discuss the specification of side-effects and the construction of transition systems. In section 4 an operational semantics and the concept of partial correctness assertions are introduced. We define a 'partial correctness semantics' and a 'language of assertions', suitable to formulate any partial correctness assertion. Next, in section 5, we present a proof system for deriving partial correctness assertions, which will be proved relatively complete. The paper is concluded with a short discussion on some extensions.

Note. A related version of this report appeared as "Process algebra and Hoare's logic" (see [10]).
Note. The specification formalism introduced here refers to the sequential fragment of ACP, the Algebra of Communicating Processes. ACP is an algebraic framework suitable both for the specification and the verification of communicating processes. For the latter it provides axiom systems concerning the equality relation over process specifications. Here we do not consider such axiom systems, but from the start concentrate on a semantical approach. In [10] it is proved that the semantical notions discussed in this paper satisfy the relevant ACP axioms. ACP is discussed in e.g. [5].

Σ_0	*Constants:*	a	for any atomic action $a \in A$
		δ	deadlock ($\delta \notin A$)
	Binary functions:	$+$	alternative composition (sum)
		$\cdot$	sequential composition (product)
Σ_{i+1}	*Constants:*	$<x \mid E>$	if E is a pure system over Σ_i and x is a solution of E
Σ	$\bigcup_n \Sigma_n \quad (n \in \mathbb{N})$		
Σ^+	*Unary functions:*	π_n	projection, $n \in \mathbb{N}$

Table 1: The signature Σ^+

2 Process expressions

We introduce a language in which processes can be specified. A parameter with respect to this language is a set A of atomic actions with typical elements $a, b, c, \ldots$. Let $V = \{x, y, z, \ldots\}$ be a set of variables. The language is built inductively from V and the constants and functions of the signature Σ^+ in table 1, and will be defined in three stages.

2.1 Finite processes

The signature Σ_0 can be used to specify finite processes. For each atomic action $a \in A$ there is a constant $a \in \Sigma_0$. Further there is a special constant $\delta \notin A$ representing 'deadlock', i.e. the acknowledgement of a process that it has no possibility to do anything any more. There are two binary functions in Σ_0, $+$ (sum) and $\cdot$ (product). Some intuitions: $x + y$ represents the process which first makes a choice between its summands x and y, and then proceeds with the chosen course of action; $x \cdot y$ (or xy, for short) represents the process x, followed after possible successful termination of x by y. The process x does not terminate successfully if it ends in deadlock. We take $\cdot$ to be most binding of all operators and $+$ to be least binding, eg. $xy + z$ means $(x \cdot y) + z$.

Let $\mathcal{P}_0$ be the least set such that

- $A_\delta \subseteq \mathcal{P}_0$, where $A_\delta \stackrel{\text{def}}{=} A \cup \{\delta\}$.
- If $p, q \in \mathcal{P}_0$, then $p + q \in \mathcal{P}_0$ and $pq \in \mathcal{P}_0$.

So $\mathcal{P}_0$ is the set of closed process expressions over Σ_0. Every element of $\mathcal{P}_0$ now specifies a *finite process*.

2.2 Recursively specified processes

We extend our process language with a mechanism for specifying processes *recursively*. This gives us in particular the means to specify possibly infinite processes. We first give an example and then start with some formal definitions.

Example. Consider an automaton which behaves as follows: After the insertion of a coin and a push on a button a or b it serves a drink encoded by that button; if the button for restitution is pushed, the inserted coin will be returned. Assume that the automaton is tacitly maintained: All drinks are always in stock and there is always room for insertions. Our running example concerns the (proper) behaviour of a user of this automaton, and can be described as a composition of the following atomic actions:

in	(insert a coin)
p_a, p_b	(push one of the buttons a or b respectively)
pr	(push the button for restitution and collect the returned coin)
co	(collect the delivery of the automaton)
st	(stop behaving as a user)

The recursive specification

$$U = in((p_a + p_b)co + pr)U + st$$

expresses that the left-hand side of this equation is specified recursively by the right-hand side: A user behaviour U consists of either inserting a coin or stopping the behaviour; in case of the first atomic action then to push a button for one of the drinks or to handle restitution; in case of the first alternative then to collect the delivery of the automaton; then to restart the preceding process. *(to be continued)*

Definition 2.2.1 *A* recursive specification $E = \{x = t_x \mid x \in V_E\}$ over Σ_i *is a set of equations where V_E is a set of variables and t_x some process expression over Σ_i only containing variables of V_E. For each $x \in V_E$ there is exactly one equation (the set V_E need not be finite).*

So up till now Σ_0 is the only signature over which recursive specifications can be defined. A *solution* of E is an interpretation of the variables in V_E as processes in a certain semantics, such that the equations of E are satisfied. For instance the recursive specification $\{x = x\}$ has any process as its solution and $\{x = ax\}$ has the infinite process "a^ω" as its solution. We introduce the following two syntactical restrictions on recursive specifications:

Definition 2.2.2 *1. A recursive specification $E = \{x = t_x \mid x \in V_E\}$ over Σ_i is called* guarded *if each occurrence of a variable y in the expressions t_x occurs in a subterm pM with $p \in \mathcal{P}_i$. We speak of* guarded systems *instead of guarded recursive specifications* [1].

2. A guarded system $E = \{x = t_x \mid x \in V_E\}$ over Σ_i is called pure *if for any subterm $M \cdot N$ of one of the t_x we have that M contains no variables of V_E.*

The notion 'pure' is typical for this paper. By considering only systems which are pure, we can prove our completeness result. Remark that the specification of U in the example above is pure. Now the signature Σ, in which we are interested, can be properly defined in an inductive manner. We will study partial correctness assertions over this signature.

Definition 2.2.3 *1. The signature Σ_{i+1} is obtained by extending Σ_i in the following way: For each pure system $E = \{x = t_x \mid x \in V_E\}$ over Σ_i a set of constants $\{<x \mid E> \mid x \in V_E\}$, where $<x \mid E>$ denotes the x-component of a solution of E, is added to Σ_i.*

2. The signature Σ is defined as $\bigcup_n \Sigma_n$ $(n \in \mathbb{N})$. We call a recursive specification E pure (or guarded) over Σ *if E is a pure (guarded respectively) system over Σ_i for some i.*

[1] In section 5.4 we return to this definition of guardedness.

For instance $<x \mid \{x = <y \mid \{y = ay + b\}> x + c\}>$ is a closed process expression over Σ (even over Σ_2), but $<x \mid \{x = axb + c\}>$ is not since it refers to a specification which is not pure. Unless stated otherwise we consider process expressions and pure systems over Σ. We introduce some more notations:

Let $E = \{x = t_x \mid x \in V_E\}$ be a pure system, and t a process expression. Then $<t \mid E>$ denotes the process expression in which each occurrence of $x \in V_E$ in t is replaced by $<x \mid E>$, e.g. $<aax \mid \{x = ax\}>$ denotes the process expression $aa<x \mid \{x = ax\}>$. If we assume that the variables in a recursive specification are chosen freshly, there is no need to repeat E in each occurrence of $<x \mid E>$. Variables reserved in this way are called *formal variables* and denoted by capital letters. We adopt the convention that $<X \mid E>$ can be abbreviated by X once E is declared. As an example consider $E = \{X = aX\}$: The closed process expression aaX abbreviates $aa<X \mid \{X = aX\}>$.

Let $\mathcal{P}$ be the least set satisfying

- $A_\delta \subseteq \mathcal{P}$.
- If $p, q \in \mathcal{P}$, then $p + q \in \mathcal{P}$ and $pq \in \mathcal{P}$.
- If $E = \{x = t_x \mid x \in V_E\}$ is a pure system over Σ, then $\{<x \mid E> \mid x \in V_E\} \subseteq \mathcal{P}$.

and $\mathcal{P}_\mathrm{i}$ defined likewise by only considering pure systems over Σ_i. Every element of $\mathcal{P}$ ($\mathcal{P}_\mathrm{i}$ respectively) now specifies a *recursively specifiable process*.

2.3 Finite projections

The signature Σ^+, defined as an extension of Σ by adding unary operators π_n for all $n \in \mathbb{N}$ to Σ, is only needed for technical matters. The projection operator π_n stops a process after it has performed n atomic actions. Let $\mathcal{P}^+$ denote the set of closed process expressions over this signature.

3 Processes having side-effects

In this section we discuss the specification of atomic actions having side-effects. Furthermore we introduce a calculus for deriving transition systems concerning processes having side-effects.

3.1 The functions *action* and *effect*

We regard processes as having a state: Let S be a nonempty set of states, with typical elements $s, s', \ldots$. The 'state labelled process expression' (x, s) denotes the process x in state s. The idea is that the execution of an atomic action a in state s results in an action a' representing the observable activity of this execution (an atomic action or δ), and in a resulting state s' [2]. This idea is formalized by *given* (total) functions

$$action : A_\delta \times S \to A_\delta \quad \text{and} \quad effect : S \times A_\delta \to S$$

which determine the relation between elements a of A_δ and elements s of S, the set of states. The functions *action* and *effect* were introduced in [2]. With $action(a, s)$ we denote the observable activity which represents the execution of a in state s; with $effect(s, a)$ we denote the resulting state. This 'operational view' on the execution of elements of A in some state will be generalized to an operational semantics based on transition rules in the style of Plotkin. We first introduce some restrictions and notations: It is assumed that

[2] As an example think of the representation of a program in a high level language as *Pascal* in process algebra. Let a variable x be declared as an integer and S denote the set of valuations from declared variables to their full domains. If we regard an assignment $x := x + 1$ as an atomic action a, then $a' = a$ if $s(x) <$ MAXINT and δ otherwise. Of course $s'(x) = s(x) + 1$ and $s'(y) = s(y)$ if $y \not\equiv x$.

1. $\forall s \in S(action(\delta, s) = \delta)$

for, in each state δ should indeed denote deadlock;

2. $\forall s \in S(effect(s, \delta) = s)$

because deadlock should not alter a state. We demand that for all functions *action* and *effect* considered the properties 1 and 2 hold. Another way to state this is to demand that δ is *inert* (with respect to the functions *action* and *effect*).

- We use the following abbreviations: $a(s)$ for $action(a, s)$ and $s(a)$ for $effect(s, a)$.

We conclude this section with a definition concerning the general framework in which we will study state labelled process expressions.

Definition 3.1.1 *A structure $\langle A, S, action, effect\rangle$ is a quadruple containing the set A of atomic actions, a nonempty set S of states and functions $action : A_\delta \times S \to A_\delta$ and $effect : S \times A_\delta \to S$ such that δ is inert.*

We use symbols $\mathcal{S}, \mathcal{S}'$ as syntactic variables for structures.

3.2 Transition systems

We introduce for all $a \in A$ a binary transition relation $\xrightarrow{a}$ over state labelled process expressions. In general we mean by a transition $(x, s) \xrightarrow{a} (y, s')$ that by performing an action a the process x in state s can evolve into y in state s'. To represent successful termination we introduce a special element $\surd$ not in Σ^+ and for all $a \in A$ a relation $\xrightarrow{a} (\surd, .)$ between state labelled process expressions and states: The expression $(x, s) \xrightarrow{a} (\surd, s')$ denotes that the process x in state s can terminate successfully in state s' by performing a. Typically an atomic action a will be related to transitions $(a, s) \xrightarrow{a(s)} (\surd, s(a))$, provided $a(s) \neq \delta$. In case $a(s) = \delta$ there simply is no related transition[3]. In table 2 we present a proof system, the *effect rules*, by which we can derive transitions. Notice that any structure fixes the effect rules. We define $\xrightarrow{\sigma}\!\!\!\!\rightarrow$ for $\sigma \in A^*$ as the reflexive and transitive closure of all transition relations:

- $(x, s) \xrightarrow{\lambda}\!\!\!\!\rightarrow (x, s)$ (λ denoting the empty string over A^*)
- $$\frac{(x, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (x', s') \quad (x', s') \xrightarrow{a} (x'', s'')}{(x, s) \xrightarrow{\sigma a}\!\!\!\!\rightarrow (x'', s'')} \qquad \frac{(x, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (x', s') \quad (x', s') \xrightarrow{a} (\surd, s'')}{(x, s) \xrightarrow{\sigma a} (\surd, s'')} \qquad (a \in A)$$

Instances of this relation will be called *effect reductions*. We here give an example of a property of effect reductions that will turn out to be useful:

Lemma 3.2.1 *'decomposition' If for some string $\sigma \in A^*$, $s \in S$ we have $(xy, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')$, then there are σ_1, σ_2 and s'' such that $\sigma_1\sigma_2 \equiv \sigma$, $(x, s) \xrightarrow{\sigma_1}\!\!\!\!\rightarrow (\surd, s'')$ and $(y, s'') \xrightarrow{\sigma_2}\!\!\!\!\rightarrow (\surd, s')$.* □

This can be proved by induction on the length of derivations. We may call *decomposition* a derived effect rule.

The next step towards our operational semantics is to associate a *transition system* $ts((p, s))$ to any state labelled closed process expression (p, s), representing all possible transitions. Consider the graph $\mathcal{G}((p, s))$ defined as follows:

$$\begin{aligned} \text{NODES} &\stackrel{\text{def}}{=} \{(p', s') \mid \text{there is } \sigma \in A^* \text{ such that } (p, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (p', s')\} \cup \\ &\quad\ \{(\surd, s') \mid \text{there is } \sigma \in A^* \text{ such that } (p, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')\} \\ \text{ARCS} &\stackrel{\text{def}}{=} \{k \xrightarrow{a} k' \mid k, k' \in \text{NODES and } k \xrightarrow{a} k' \text{ a transition}\} \end{aligned}$$

[3] Another possible approach would be to define transitions $(a, s) \xrightarrow{a(s)} (\delta, s)$ in case $a(s) = \delta$.

$a \in A$:	$(a,s) \xrightarrow{a(s)} (\surd, s(a))$	(if $a(s) \neq \delta$)
$+$:	$\dfrac{(x,s) \xrightarrow{a} (x',s')}{(x+y,s) \xrightarrow{a} (x',s')}$	$\dfrac{(x,s) \xrightarrow{a} (\surd,s')}{(x+y,s) \xrightarrow{a} (\surd,s')}$
	$\dfrac{(y,s) \xrightarrow{a} (y',s')}{(x+y,s) \xrightarrow{a} (y',s')}$	$\dfrac{(y,s) \xrightarrow{a} (\surd,s')}{(x+y,s) \xrightarrow{a} (\surd,s')}$
$\cdot$:	$\dfrac{(x,s) \xrightarrow{a} (x',s')}{(xy,s) \xrightarrow{a} (x'y,s')}$	$\dfrac{(x,s) \xrightarrow{a} (\surd,s')}{(xy,s) \xrightarrow{a} (y,s')}$
recursion :	$\dfrac{(<t_x \mid E>,s) \xrightarrow{a} (y,s')}{(<x \mid E>,s) \xrightarrow{a} (y,s')}$	$\dfrac{(<t_x \mid E>,s) \xrightarrow{a} (\surd,s')}{(<x \mid E>,s) \xrightarrow{a} (\surd,s')}$
π_n :	$\dfrac{(x,s) \xrightarrow{a} (x',s')}{(\pi_{n+1}(x),s) \xrightarrow{a} (\pi_n(x'),s')}$	$\dfrac{(x,s) \xrightarrow{a} (\surd,s')}{(\pi_{n+1}(x),s) \xrightarrow{a} (\surd,s')}$

Table 2: Effect rules

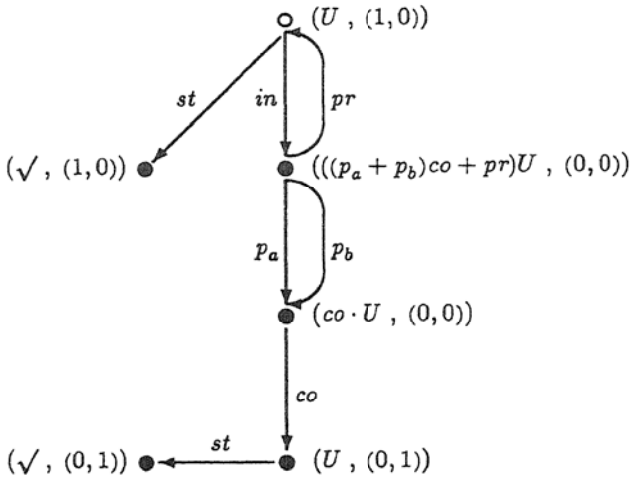

Figure 1: A transition system

By defining the node (p, s) as the *root* of $\mathcal{G}((p, s))$, this construction yields $ts((p, s))$, the transition system associated to (p, s). Here the state s will be called the *initial* state of $ts((p, s))$. Any state s' such that $(\surd, s')$ is a node in $\mathcal{G}((p, s))$ will be called a *final* state of $ts((p, s))$. We return to our running example:

Example. *(continued)* Concerning the recursive specification $U = in((p_a + p_b)co + pr)U + st$ we consider a user having initially $N + 1$ coins and no drinks. Take $\mathcal{S} = \langle A, S, action, effect \rangle$ with $A = \{in, p_a, p_b, co, pr, st\}$ and with each state of S being a pair of counters which are used to keep track of the number of coins a user has and the number of drinks already collected: Let $S = Busy \cup Final$ where $Busy = \{(i, j) \mid i + j = N\}$ and $Final = \{(i, j) \mid i + j = N + 1\}$. We define the functions *action* and *effect* in the following way:

$$in(s) \stackrel{\text{def}}{=} \begin{cases} in & \text{if } (i,j) \in Final - (0, N+1) \\ \delta & \text{otherwise} \end{cases} \qquad (i,j)(in) \stackrel{\text{def}}{=} \begin{cases} (i-1, j) & \text{if } (i,j) \in Final - (0, N+1) \\ (i,j) & \text{otherwise} \end{cases}$$

$$co(s) \stackrel{\text{def}}{=} \begin{cases} co & \text{if } (i,j) \in Busy \\ \delta & \text{otherwise} \end{cases} \qquad (i,j)(co) \stackrel{\text{def}}{=} \begin{cases} (i, j+1) & \text{if } (i,j) \in Busy \\ (i,j) & \text{otherwise} \end{cases}$$

$$pr(s) \stackrel{\text{def}}{=} \begin{cases} pr & \text{if } (i,j) \in Busy \\ \delta & \text{otherwise} \end{cases} \qquad (i,j)(pr) \stackrel{\text{def}}{=} \begin{cases} (i+1, j) & \text{if } (i,j) \in Busy \\ (i,j) & \text{otherwise} \end{cases}$$

and the atomic actions p_a, p_b and st inert. Assume $N = 0$. In figure 1 the transition system for the process U in initial state $(1, 0)$ is displayed. *(to be continued)*

4 Semantics

In this section we define an equality relation over transition systems, which will be used to construct an operational semantics. Furthermore the concept of partial correctness assertions is introduced and we derive a 'partial correctness semantics'. Finally we define a language of assertions, based on a structure $\mathcal{S}$, which can be used to refer to any part of the state space.

4.1 An operational semantics

Let $\mathcal{S}$ be some structure. The idea is that two closed process expressions p and q are *operationally equivalent in* $\mathcal{S}$ if they satisfy the following property: The representation of any execution of p in some initial state s (in terms of its performance of atomic actions) also represents an execution of q in initial state s, and vice versa. We now formalize this idea. Consider the set of all transition systems. In order to define an equality relation over this set, we use the notion of a bisimulation (see [8]):

Definition 4.1.1 *A binary relation $R \subseteq (\mathcal{P} \times S) \times (\mathcal{P} \times S)$ is a* bisimulation *if the following conditions are satisfied ($a \in A$):*

1. *If $(p,s)R(q,s)$ and $(p,s) \xrightarrow{a} (p',s')$, then there is a (q',s') such that $(q,s) \xrightarrow{a} (q',s')$ and $(p',s')R(q',s')$.*
2. *If $(p,s)R(q,s)$ and $(q,s) \xrightarrow{a} (q',s')$, then there is a (p',s') such that $(p,s) \xrightarrow{a} (p',s')$ and $(p',s')R(q',s')$.*
3. *If $(p,s)R(q,s)$, then $(p,s) \xrightarrow{a} (\surd,s')$ for some s' if and only if $(q,s) \xrightarrow{a} (\surd,s')$.*

Two transition systems $ts((p,s))$ and $ts((q,s))$ are bisimilar, *we write $ts((p,s)) \leftrightarrows ts((q,s))$, if there exists a bisimulation R with $(p,s)R(q,s)$. Remark that equality of initial states is demanded here.*

It is not difficult to see that $\leftrightarrows$ is an equivalence relation. Let $[(p,s)]$ be some unique representation of the equivalence class of $ts((p,s))$. We define an operational semantics as follows:

Definition 4.1.2 1. *A closed process expression p is interpreted in $\mathcal{S} = \langle A, S, action, effect \rangle$ as $\{[(p,s)] \mid s \in S\}$.*

2. *Two closed process expressions p and q are* operationally equivalent in $\mathcal{S}$, *we write $\mathcal{S} \models p =_{op} q$, if for all $s \in S$ we have $ts((p,s)) \leftrightarrows ts((q,s))$, that is if $\{[(p,s)] \mid s \in S\} = \{[(q,s)] \mid s \in S\}$.*

Remark that if we want to consider a structure $\mathcal{S} = \langle A, S, action, effect \rangle$ in which for two atomic actions a and b we have for all $s \in S$ that $a(s) = b(s)$ and $s(a) = s(b)$, then $\mathcal{S} \models a =_{op} b$. This reflects the circumstance that *in* $\mathcal{S}$ the constants a and b apparently denote the same atomic action. We finally prove that for any structure $\mathcal{S}$ the relation $=_{op}$ is a congruence, which implies that closed process expressions occurring in a specification may be replaced by operationally equivalent expressions.

Theorem 4.1.3 *For all structures $\mathcal{S}$ the relation $=_{op}$ is a congruence with respect to the operators involved.*

Proof. The theorem can be proved by inspection: Fix $\mathcal{S}$ and assume $\mathcal{S} \models p =_{op} p', \mathcal{S} \models q =_{op} q'$. We have to show $\mathcal{S} \models p \Box q =_{op} p' \Box q'$ for $\Box \in \{+, \cdot\}$. As an example we consider sequential composition: Suppose that $ts((p,s)) \leftrightarrows ts((p',s))$ by the relation $R_{(p,s)}$ and $ts((q,s)) \leftrightarrows ts((q',s))$ by $R_{(q,s)}$ for all $s \in S$. Fix $s_0 \in S$ and let $S_0 = \{s \in S \mid \exists \sigma \in A^*((p,s_0) \xrightarrow{\sigma} (\surd,s))\}$. We define a relation R as follows:

$$R \stackrel{\text{def}}{=} \{((rq,s),(r'q',s)) \mid (r,s)R_{(p,s_0)}(r',s)\} \cup \bigcup_{s \in S_0} R_{(q,s)}$$

We have $(pq,s_0)R(p'q',s_0)$ and by induction on the length of derivations it follows that R is a bisimulation. $\Box$

4.2 A partial correctness semantics

Let A be a set of atomic actions. We introduce a logical language $\mathcal{L}$, the *language of assertions*, in order to reason formally about any structure $\mathcal{S} = \langle A, S, action, effect \rangle$. Let *Pred* be some set of unary predicate symbols. We define $\mathcal{L}$ as follows:

one variable:	v	
unary function symbols:	$effect_a$	(for all $a \in A_\delta$)
unary predicate symbols:	$stop_a$	(for all $a \in A_\delta$)
unary predicate symbols:	P	(for all $P \in Pred$)
connectives:	$\neg, \vee, \wedge, \rightarrow, \leftrightarrow$	
auxiliary symbols:	), ,, (	

Parameters for $\mathcal{L}$ are a set A of atomic actions and some set *Pred* of unary predicate symbols. $\mathcal{L}-$formulae are called *assertions* and we use $\alpha, \beta, \ldots$ as syntactic variables for assertions. Remark that a term always contains exactly one occurrence of the variable v.

Having defined $\mathcal{L}$ we can give the definition of a partial correctness assertion in syntactical terms:

Definition 4.2.1 *1. A* partial correctness assertion over $\mathcal{L}$ *is an expression of the form* $\{\alpha\}\, p\, \{\beta\}$ *where* p *is a closed process expression over* Σ.

2. A correctness formula over $\mathcal{L}$ *is an expression of the form* $\{\alpha\} t \{\beta\}$ *where* t *is a process expression over* Σ.

So a partial correctness assertion can be regarded as a 'closed' correctness formula. We need the more general concept of 'correctness formulae' to define a proof system in which we can derive partial correctness assertions concerning recursively specified processes. Correctness formulae are not subject to Boolean operations. Let $\Phi, \Phi', \ldots$ be syntactic variables for (possibly empty) sets of correctness formulae.

We now define the way we interprete assertions and correctness formulae. Let $\mathcal{S} = \langle A, S, action, effect \rangle$ be some fixed structure and $\{\overline{P} \mid P \in Pred\}$ a set of unary predicates over S. We define an interpretation $\mathcal{I}$ of $\mathcal{L}$ with domain S as follows:

$$\textit{terms}: \quad [\![effect_a]\!]^{\mathcal{S},\mathcal{I}} = \lambda s.s(a) \quad (s \in S)$$

$$\begin{array}{lll} \textit{formulae}: & \mathcal{S} \models_{\mathcal{I}} \alpha & \text{if} \quad \forall s \in S(\mathcal{S} \models_{\mathcal{I}} \alpha[s]) \\ & \mathcal{S} \models_{\mathcal{I}} stop_a[s] & \text{if} \quad a(s) = \delta \\ & \mathcal{S} \models_{\mathcal{I}} P[s] & \text{if} \quad \overline{P}(s) \text{ holds} \\ & \text{and compound formulae as usual} & \end{array}$$

Before defining the way correctness formulae are interpreted we introduce some more notation:

Let t be some process expression over Σ and $\overline{x}$ a sequence of variables. We write $t = t(\overline{x})$ to indicate that all variables occurring in t are *among* the elements of the sequence $\overline{x}$. If $\overline{p}$ is a sequence of elements of $\mathcal{P}^+$ (the set of closed process expressions over Σ^+), then $t(\overline{p})$ denotes the closed process expression obtained by replacing all variables in t by the corresponding $\overline{p}-$elements. We write '$\forall \overline{p}$ ' if we want to consider all sequences of length $\overline{x}$ over $\mathcal{P}^+$.

This applies to correctness formulae as follows: If Φ is a set of correctness formulae and $\overline{x}$ a sequence such that $\{\alpha\} t \{\beta\} \in \Phi \Longrightarrow t = t(\overline{x})$, then we write $\Phi = \Phi(\overline{x})$ and $\Phi(\overline{p}) = \{\{\alpha\} t(\overline{p}) \{\beta\} \mid \{\alpha\} t \{\beta\} \in \Phi\}$.

Definition 4.2.2 *1. A partial correctness assertion* $\{\alpha\}\, p\, \{\beta\}$ *is* true in $\mathcal{S}$ under $\mathcal{I}$, *we write* $\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, p\, \{\beta\}$, *if for all* $s, s' \in S$, $\sigma \in A^*$ *we have:*

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

2. *A correctness formula* $\{\alpha\}\, t\, \{\beta\}$ *with* $t = t(\overline{x})$ *is* true in $\mathcal{S}$ under $\mathcal{I}$, *we write* $\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, t\, \{\beta\}$, *if*

$$\forall \overline{p}\, [\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, t(\overline{p})\, \{\beta\}].$$

So the truth of a partial correctness assertion $\{\alpha\}p\{\beta\}$ expresses the fact that any successful execution of p in an initial state satisfying α, results in a final state satisfying β. A semantical relation based on partial correctness assertions can be defined as follows:

Definition 4.2.3 *We call two closed process expressions* p *and* q equivalent under partial correctness in $\mathcal{S}$ under I, *we write* $\mathcal{S} \models_{\mathcal{I}} p =_{pc} q$, *if for all* $\mathcal{L}-$*formulae* α, β *we have:*

$$\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, p\, \{\beta\} \iff \mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, q\, \{\beta\}.$$

Obviously $=_{pc}$ is always an equivalence relation, we show that it is also a congruence:

Theorem 4.2.4 *The relation* $=_{pc}$ *is a congruence with respect to the operators involved.*

Proof. We prove the theorem by inspection: Let $\mathcal{I}$ be an interpretation of $\mathcal{L}$ in $\mathcal{S}$ and suppose $\mathcal{S} \models_{\mathcal{I}} p =_{pc} p', \mathcal{S} \models_{\mathcal{I}} q =_{pc} q'$. We have to show $\mathcal{S} \models_{\mathcal{I}} p \Box q =_{pc} p' \Box q'$ for $\Box \in \{+, \cdot\}$. As an example we consider alternative composition: It is sufficient to show that if we have a reduction $(p+q, s) \xrightarrow{\sigma} (\surd, s')$, then there is a reduction $(p'+q', s) \xrightarrow{\rho} (\surd, s')$. This follows easily: Suppose the first transition in our reduction, say $(p+q, s) \xrightarrow{a} (r, s'')$, is a consequence of $(p, s) \xrightarrow{a} (r, s'')$. By the induction hypothesis we have $(p', s) \xrightarrow{\rho} (\surd, s')$ for some string $\rho \in A^*$, so using the first transition of this reduction we derive $(p'+q', s) \xrightarrow{\rho} (\surd, s')$. □

We extend the relation $=_{pc}$ to open process expressions in the obvious way: $\mathcal{S} \models_{\mathcal{I}} t =_{pc} t'$ if

$$\forall \overline{p}\, [\mathcal{S} \models_{\mathcal{I}} t(\overline{p}) =_{pc} t'(\overline{p})]$$

for $t = t(\overline{x})$ and $t' = t'(\overline{x})$. In the following lemma we present a useful property of this extended relation.

Lemma 4.2.5 *'distributivity' For all* t, t', t'' *over* Σ^+ *we have* $t(t'+t'') =_{pc} tt' + tt''$ *and* $(t+t')t'' =_{pc} tt'' + t't''$. □

This can be proved by induction on the length of derivations, using *decomposition* (see lemma 3.2.1). The semantical relations $=_{op}$ and $=_{pc}$ are by definition related in the following way:

Theorem 4.2.6 *If for some* $\mathcal{S}$ *and closed process expressions* p *and* q *we have* $\mathcal{S} \models p =_{op} q$, *then also* $\mathcal{S} \models_{\mathcal{I}} p =_{pc} q$ *for any interpretation* $\mathcal{I}$ *of* $\mathcal{L}$ *in* $\mathcal{S}$. □

Of course the converse does not hold. It is not difficult to define $\mathcal{L}, \mathcal{S}, \mathcal{I}$ such that $\mathcal{S} \not\models a(b+c) =_{op} ab + ac$, whereas $\mathcal{S} \models_{\mathcal{I}} a(b+c) =_{pc} ab + ac$ by *distributivity*.

We finally introduce for any structure $\mathcal{S}$ a special language of assertions, suitable to refer to any unary predicate over the state space of $\mathcal{S}$.

Definition 4.2.7 *Let* $\mathcal{S} = \langle A, S, action, effect \rangle$ *be some fixed structure and Pred contain exactly one predicate symbol for each subset of* S. *We write in this case* $\mathcal{L}_S$, *the language of assertions about* S, *and we interpret the symbols of Pred as the corresponding predicates over* S. *We will omit the subscript* $\mathcal{I}$ *when interpreting assertions of* $\mathcal{L}_S$

We write Tr_S for the set of all true assertions in $\mathcal{L}_S$, so $\alpha \in Tr_S$ if $\forall s \in S\ (\mathcal{S} \models \alpha[s])$.

5 Deriving partial correctness assertions

In this section we show how to derive partial correctness assertions. We present a proof system H, and we show that H is *relatively complete*. We further discuss a somewhat more restricted definition of 'guarded systems' and we finally show that in H we cannot handle partial correctness assertions concerning all *guardedly* specifiable processes.

5.1 Proof systems, soundness and completeness

A *proof system* G is a finite set of (schemes of) axioms and rules in a natural deduction format. It can be used to derive correctness formulae, and in particular partial correctness assertions, over any language $\mathcal{L}$ and in any structure $\mathcal{S} = \langle A, S, \mathit{action}, \mathit{effect}\rangle$.

We write $Tr_{\mathcal{S}}, \Phi \vdash^G \psi$ if there is a derivation of ψ in G which uses hypotheses from $Tr_{\mathcal{S}}$ and Φ. If G is fixed, we omit the superscript G in $\vdash^G$. A partial correctness assertion $\{\alpha\}\, p\, \{\beta\}$ over $\mathcal{L}$ is called *derivable in G and $\mathcal{S}$* if $Tr_{\mathcal{S}} \vdash^G \{\alpha\}\, p\, \{\beta\}$ (so $\Phi = \emptyset$). A proof system is always associated with a signature $\Sigma_G \subseteq \Sigma$ of the process expressions occurring in all derivable partial correctness assertions.

We call a proof system G *sound* if for all structures $\mathcal{S}$, interpretations $\mathcal{I}$ of a fixed language $\mathcal{L}$ and all correctness formulae $\{\alpha\}\, t\, \{\beta\}$ over $\mathcal{L}$ we have

$$Tr_{\mathcal{S}}, \Phi \vdash^G \{\alpha\}\, t\, \{\beta\} \implies \forall \overline{p}\, [\mathcal{S} \models_{\mathcal{I}} \Phi(\overline{p}) \Longrightarrow \mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, t(\overline{p})\, \{\beta\}]$$

with $\Phi = \Phi(\overline{x})$ and $t = t(\overline{x})$. So in particular any partial correctness assertion over $\mathcal{L}$ derivable in $G, \mathcal{S}$ is true in $\mathcal{S}$ under $\mathcal{I}$.

The proof system G is *(relatively) complete* if for any structure $\mathcal{S}$ *and* its language of assertions $\mathcal{L}_{\mathcal{S}}$ the converse holds as well for all derivable partial correctness assertions over $\mathcal{L}_{\mathcal{S}}$:

$$\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, p\, \{\beta\} \iff Tr_{\mathcal{S}} \vdash^G \{\alpha\}\, p\, \{\beta\}$$

that is, a partial correctness assertion $\{\alpha\}\, p\, \{\beta\}$ is true in $\mathcal{S}$ if and only if $\{\alpha\}\, p\, \{\beta\}$ is derivable in G (using $Tr_{\mathcal{S}}$). The adjective 'relatively' refers to the fact that $Tr_{\mathcal{S}}$ may be used in derivations: *Relative* all true assertions about $\mathcal{S}$, we have that truth and derivability in $\mathcal{S}$ coincide.

5.2 The proof system H

In table 3 we present the proof system H associated with Σ. Some comments on the rules of H: Let $\mathcal{S}, \mathcal{L}$ be fixed. The axioms I and rule IV can only be applied if the assertions occurring in their premisses are in $Tr_{\mathcal{S}}$. Concerning rule V, if $E = \{x = t_x \mid x \in V_E\}$ is a pure system, $z \in V_E$ and we have all derivations $\{\alpha_y\}\, t_y\, \{\beta_y\}$ $(y \in V_E)$ from a set of hypotheses $Tr_{\mathcal{S}} \cup \Phi \supseteq \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}$, then $\{\alpha_z\}\, <z \mid E>\, \{\beta_z\}$ is derivable from $Tr_{\mathcal{S}} \cup \Phi - \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}$. In other words, the set of hypotheses $\{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}$ is *cancelled* after the application of rule V. We show by an example how to use H (see also figure 2):

Example. *(continued)* We consider the recursive specification $U = \mathit{in}((p_a + p_b)\mathit{co} + \mathit{pr})U + \mathit{st}$ and a user having initially $N + 1$ coins and no drinks. We already introduced the structure $\mathcal{S} = \langle A, S, \mathit{action}, \mathit{effect}\rangle$ with $A = \{\mathit{in}, p_a, p_b, \mathit{co}, \mathit{pr}, \mathit{st}\}$ and $S = \mathit{Busy} \cup \mathit{Final}$ where $\mathit{Busy} = \{(i,j) \mid i + j = N\}$ and $\mathit{Final} = \{(i,j) \mid i + j = N + 1\}$. We define the following predicates over S:

$$\begin{array}{lcl} \overline{\mathit{init}}\,(i,j) & \iff & i = N + 1 \text{ and } j = 0 \\ \overline{\mathit{busy}}\,(i,j) & \iff & (i,j) \in \mathit{Busy} \\ \overline{\mathit{final}}\,(i,j) & \iff & (i,j) \in \mathit{Final} \end{array}$$

I	*axioms* $(a \in A_\delta)$	$\dfrac{\neg stop_a(v) \wedge \alpha(v) \rightarrow \beta(effect_a(v))}{\{\alpha\}\, a\, \{\beta\}}$
II	*alternative composition*	$\dfrac{\{\alpha\}\, t\, \{\beta\} \quad \{\alpha\}\, t'\, \{\beta\}}{\{\alpha\}\, t + t'\, \{\beta\}}$
III	*sequential composition*	$\dfrac{\{\alpha\}\, t\, \{\beta\} \quad \{\beta\}\, t'\, \{\gamma\}}{\{\alpha\}\, tt'\, \{\gamma\}}$
IV	*consequence*	$\dfrac{\alpha \rightarrow \alpha' \quad \{\alpha'\}\, t\, \{\beta'\} \quad \beta' \rightarrow \beta}{\{\alpha\}\, t\, \{\beta\}}$
V	*recursion*	*If* $E = \{x = t_x \mid x \in V_E\}$ *is a pure system and* $z \in V_E$, *then* $\dfrac{\forall y \in V_E \left[\begin{array}{c} \cancel{\{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}} \\ \vdots \\ \{\alpha_y\}\, t_y\, \{\beta_y\} \end{array}\right]}{\{\alpha_z\} <z \mid E> \{\beta_z\}}$

Table 3: The proof system H

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p_a \,{}^{busy}_{busy}}{\{busy\}\ p_a\ \{busy\}} \quad \dfrac{p_b \,{}^{busy}_{busy}}{\{busy\}\ p_b\ \{busy\}}}{\{busy\}\ p_a + p_b\ \{busy\}} \quad \dfrac{co\ {}^{busy}_{final}}{\{busy\}\ co\ \{final\}}}{\{busy\}\ (p_a + p_b)co\ \{final\}} \quad \dfrac{pr\ {}^{busy}_{final}}{\{busy\}\ pr\ \{final\}}}{\{final\}\ in\ \{busy\}} \quad \{busy\}\ (p_a + p_b)co + pr\ \{final\}}{\{final\}\ in((p_a + p_b)co + pr)\ \{final\}} \quad \dfrac{}{\{final\}\ x\ \{final\}}}{\{final\}\ in((p_a + p_b)co + pr)x\ \{final\}} \quad \dfrac{st\ {}^{final}_{final}}{\{final\}\ st\ \{final\}}}{\{final\}\ in((p_a + p_b)co + pr)x + st\ \{final\}} \quad \{final\}\ U\ \{final\}}{\{init\}\ U\ \{final\}}\ init \rightarrow final$$

Figure 2: A derivation in H

So we have for instance $\mathcal{S} \models init \rightarrow final$ with *init* and *final* denoting the associated predicate *symbols*. Let a^{α}_{β} be short for the assertion $\neg stop_a(v) \wedge \alpha(v) \rightarrow \beta(effect_a(v))$. In figure 2 we display a derivation of

$$\left\{p_a \,{}^{busy}_{busy},\ p_b \,{}^{busy}_{busy},\ co\ {}^{busy}_{final},\ pr\ {}^{busy}_{final},\ in\ {}^{final}_{busy},\ st\ {}^{final}_{final},\ init \rightarrow final\right\} \vdash \{init\}\ U\ \{final\}$$

by which we conclude $Tr_{\mathcal{S}} \vdash \{init\}\ U\ \{final\}$. *(end example)*

5.3 The proof system H is complete

We first prove that H is sound, and then split up H in a countable number of subsystems which will all be proven complete by an inductive argument.

Lemma 5.3.1 *The proof system H is sound.*

Proof. Induction on the length of derivations. We check the soundness of rules III and V (the other cases are straightforward). Let $\mathcal{S}, \mathcal{L}$ be fixed and $\mathcal{I}$ an interpretation of $\mathcal{L}$ in $\mathcal{S}$.

1. Suppose $Tr_{\mathcal{S}}, \Phi \vdash \{\alpha\}\ t\ \{\beta\}$ and $Tr_{\mathcal{S}}, \Phi' \vdash \{\beta\}\ t'\ \{\gamma\}$. We have to show that

$$\forall \overline{p}\ [\mathcal{S} \models_{\mathcal{I}} \Phi''(\overline{p}) \implies \mathcal{S} \models_{\mathcal{I}} \{\alpha\}\ tt'(\overline{p})\ \{\gamma\}]$$

   for any $\Phi'' \supseteq \Phi \cup \Phi'$. Let $\overline{p}$ be such that $\mathcal{S} \models_{\mathcal{I}} \Phi''(\overline{p})$, then $\mathcal{S} \models_{\mathcal{I}} \Phi(\overline{p})$ and $\mathcal{S} \models_{\mathcal{I}} \Phi'(\overline{p})$. By induction we have $\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\ t(\overline{p})\ \{\beta\}$ and $\mathcal{S} \models_{\mathcal{I}} \{\beta\}\ t'(\overline{p})\ \{\gamma\}$. Now assume $\mathcal{S} \models_{\mathcal{I}} \alpha[s]$ and $(tt(\overline{p}), s) \longrightarrow\!\!\!\!\!\rightarrow (\surd, s')$. By *decomposition* (see lemma 3.2.1) we have $(t(\overline{p}), s) \longrightarrow\!\!\!\!\!\rightarrow (\surd, s'')$ and $(t'(\overline{p}), s'') \longrightarrow\!\!\!\!\!\rightarrow (\surd, s')$, so $\mathcal{S} \models_{\mathcal{I}} \beta[s'']$ and therefore $\mathcal{S} \models_{\mathcal{I}} \gamma[s']$, by which the soundness of rule III is proved.

2. Suppose $E = \{x = t_x \mid x \in V_E\}$ is a pure system over Σ and $Tr_{\mathcal{S}} \cup \Phi$ contains the hypotheses of

$$\forall y \in V_E \left[\begin{array}{c} \{\{\alpha_x\}\ x\ \{\beta_x\} \mid x \in V_E\} \\ \vdots \\ \{\alpha_y\}\ t_y\ \{\beta_y\} \end{array}\right]$$

so $Tr_{\mathcal{S}}, \Phi, \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\} \vdash \{\alpha_y\}\, t_y\, \{\beta_y\}$ for all $y \in V_E$. By the induction hypothesis we have

$$\forall \overline{p}\, [\mathcal{S} \models_{\mathcal{I}} \Phi \cup \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}(\overline{p}) \implies \mathcal{S} \models_{\mathcal{I}} \{\alpha_y\}\, t_y(\overline{p})\, \{\beta_y\}]$$

for all $y \in V_E$. Let $\Phi' \supseteq \Phi$ and $z \in V_E$. We have to show

$$\forall \overline{p}\, [\mathcal{S} \models_{\mathcal{I}} \Phi'(\overline{p}) \implies \mathcal{S} \models_{\mathcal{I}} \{\alpha_z\}\, <z \mid E>(\overline{p})\, \{\beta_z\}].$$

Because $<z \mid E>$ is a closed process expression it is sufficient to show $\mathcal{S} \models_{\mathcal{I}} \{\alpha_z\}\, <z \mid E>\, \{\beta_z\}$. Let $E' = \{x = t'_x \mid x \in V_{E'}\}$ be the system obtained by removing all brackets in the expressions t_x as suggested by *distributivity* in lemma 4.2.5 (so $V_E = V_{E'}$). It follows that

$$\forall \overline{p}\, [\mathcal{S} \models_{\mathcal{I}} \Phi \cup \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_{E'}\}(\overline{p}) \implies \mathcal{S} \models_{\mathcal{I}} \{\alpha_y\}\, t'_y(\overline{p})\, \{\beta_y\}]$$

for all $y \in V_{E'}$. For a start we prove that $\mathcal{S} \models_{\mathcal{I}} \{\alpha_x\}\, \pi_n(<x \mid E'>)\, \{\beta_x\}$ for all $n \in \mathbb{N}, x \in V_E$:

$n = 0$: By lack of an effect rule introducing the π_0−operator it follows that no expression $(\pi_0(p), s)$ can reduce to $(\surd, .)$, so $\mathcal{S} \models_{\mathcal{I}} \big\{\{\alpha_x\}\, \pi_0(<x \mid E'>)\, \{\beta_x\} \mid x \in V_{E'}\big\}$.

$n + 1$: Suppose that for some $x_0 \in V_{E'}$ we have $\mathcal{S} \models_{\mathcal{I}} \alpha[s]$ and $(\pi_{n+1}(<x_0 \mid E'>), s) \xrightarrow{\sigma} (\surd, s')$. Let $t'_{x_0} \equiv \sum p_i y_i \; + \sum q_j$, where p_i, q_j are closed process expressions and the y_i are in $V_{E'}$. At least one of the following cases must hold:

- $(\pi_{n+1}(q_{j_0}), s) \xrightarrow{\sigma} (\surd, s')$ for a summand q_{j_0} of t'_{x_0} because $(q_{j_0}, s) \xrightarrow{\sigma} (\surd, s')$.
- $(\pi_{n+1}(<p_{i_0} y_{i_0} \mid E'>), s) \xrightarrow{\sigma} (\surd, s')$ for a summand $p_{i_0} y_{i_0}$ of t'_{x_0} because $(\pi_{n+1}(<p_{i_0} y_{i0} \mid E'>), s) \xrightarrow{\rho} (\pi_m(<y_{i_0} \mid E'>), s'') \xrightarrow{\nu} (\surd, s')$ for some $m \leq n$, $s'' \in S$ and $\rho\nu \equiv \sigma$.

In both cases we may conclude that $(t'_{x_0}(\,\overline{\pi_n(<x \mid E'>)}\,), s) \xrightarrow{\sigma} (\surd, s')$. By the induction hypothesis we have $\mathcal{S} \models_{\mathcal{I}} \big\{\{\alpha_x\}\, \pi_n(<x \mid E'>)\, \{\beta_x\} \mid x \in V_{E'}\big\}$, so by supposition we have in particular that $\mathcal{S} \models_{\mathcal{I}} \{\alpha_{x_0}\}\, t'_{x_0}(\,\overline{\pi_n(<x \mid E'>)}\,)\, \{\beta_{x_0}\}$ and thus $\beta_{x_0}(s')$ holds. We conclude $\mathcal{S} \models_{\mathcal{I}} \big\{\{\alpha_x\}\, \pi_{n+1}(<x \mid E'>)\, \{\beta_x\} \mid x \in V_{E'}\big\}$.

Next we show that if $\forall n \in \mathbb{N}\big(\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\pi_n(p)\{\beta\}\big)$, then $\mathcal{S} \models_{\mathcal{I}} \{\alpha\}p\{\beta\}$ for all $p \in \mathcal{P}^+$. Suppose $\mathcal{S} \models_{\mathcal{I}} \alpha[s]$ and $(p, s) \xrightarrow{\sigma} (\surd, s')$. By inspection of the effect rules for the π_n−operators it follows easily that for n sufficiently large $(\pi_n(p), s) \xrightarrow{\sigma} (\surd, s')$, so $\mathcal{S} \models_{\mathcal{I}} \beta[s']$, which shows that $\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\, p\, \{\beta\}$. Now we may conclude $\mathcal{S} \models_{\mathcal{I}} \big\{\{\alpha_x\}\, <x \mid E'>\, \{\beta_x\} \mid x \in V_{E'}\big\}$. By inspection of the effect rules for *recursion* it follows that $\mathcal{S} \models_{\mathcal{I}} \big\{\{\alpha_x\}\, <x \mid E>\, \{\beta_x\} \mid x \in V_E\big\}$, so in particular

$$\mathcal{S} \models_{\mathcal{I}} \{\alpha_z\}\, <z \mid E>\, \{\beta_z\}$$

by which the soundness of rule V is proved. □

We now turn to the issue of the (relative) completeness of H. We introduce the following abbreviations:

- H_0 denotes the proof system containing rules I – IV; obviously H_0 is associated with Σ_0.
- H_{i+1} denotes the proof system H with the applicability of rule V restricted to pure systems over Σ_i. So H_{i+1} is associated with the signature Σ_{i+1}.

We will prove that H is complete by showing that H_0 is complete, and that the completeness of H_i leads to the completeness of H_{i+1}.

Lemma 5.3.2 *The proof system H_0 is complete.*

Proof. Let $\mathcal{S}$ be a fixed structure. Because H is sound (and therefore H_0 as well), we only have to prove

$$\mathcal{S} \models \{\alpha\}\, p\, \{\beta\} \implies Tr_{\mathcal{S}} \vdash^{H_0} \{\alpha\}\, p\, \{\beta\}$$

for all p occurring in partial correctness assertions derivable in $H0$. Recall that $\mathcal{P}_0$, the set of closed process expressions over Σ_0, is specified inductively (see section 2.1). Therefore we apply induction on the structure of p.

$p \equiv a \in A_\delta$: Now $\neg stop_a(v) \wedge \alpha(v) \rightarrow \beta(effect_a(v)) \in Tr_{\mathcal{S}}$ for, if $\mathcal{S} \models \neg stop_a \wedge \alpha[s]$, then $\mathcal{S} \models \beta[s(a)]$ by supposition, and therefore $\mathcal{S} \models \beta(effect_a(v))[s]$. By the axiom I we derive $Tr_{\mathcal{S}} \vdash^{H_0} \{\alpha\}\, a\, \{\beta\}$.

$p \equiv q + r$: Note that $\mathcal{S} \models \{\alpha\}\, q\, \{\beta\}$, $\mathcal{S} \models \{\alpha\}\, r\, \{\beta\}$. By the induction hypothesis and rule II we derive $Tr_{\mathcal{S}} \vdash^{H_0} \{\alpha\}\, q + r\, \{\beta\}$.

$p \equiv qr$: By *decomposition* (see lemma 3.2.1) and the definition of $\mathcal{L}_{\mathcal{S}}$ there must be an assertion γ such that $\mathcal{S} \models \{\alpha\}\, q\, \{\gamma\}$ and $\mathcal{S} \models \{\gamma\}\, r\, \{\beta\}$. By the induction hypothesis and rule III we derive $Tr_{\mathcal{S}} \vdash^{H_0} \{\alpha\}\, qr\, \{\beta\}$. □

This is the basis for an inductive proof of the completeness of H. Before proving the completeness of H_{i+1}, we take a closer look at a statement $\mathcal{S} \models \{\alpha\} <x \,|\, E> \{\beta\}$ with E a pure system over Σ_i. In the following lemma we show that such a statement implies $Tr_{\mathcal{S}} \vdash^{H_{i+1}} \{\alpha\} <x \,|\, E> \{\beta\}$.

Lemma 5.3.3 *Let $\mathcal{S} = \langle A, S, action, effect\rangle$ be some structure, $E = \{x = t_x \,|\, x \in V_E\}$ a pure system over Σ_i and $x_0 \in V_E$. If H_i is complete, then*

$$\mathcal{S} \models \{\alpha\} <x_0 \,|\, E> \{\beta\} \implies Tr_{\mathcal{S}} \vdash^{H_{i+1}} \{\alpha\} <x_0 \,|\, E> \{\beta\}.$$

Proof. Let $E' = \{x = t'_x \,|\, x \in V_{E'}\}$ be the system obtained by removing all brackets in the expressions t_x as suggested by *distributivity* (see lemma 4.2.5), so $V_E = V_{E'}$. It follows that $\mathcal{S} \models \{\alpha\} <x_0 \,|\, E'> \{\beta\}$. We construct *weakest preconditions* for all constants $<x \,|\, E'>$ and β. For any $x \in V_{E'}$ let the assertion α_x be as follows:

$\mathcal{S} \models \alpha_x[s] \iff$ There are $s' \in S, \sigma \in A^*$ such that $(<x_0 \,|\, E'>, s') \xrightarrow{\sigma}\!\!\!\!\rightarrow\ (<x \,|\, E'>, s)$ and $\mathcal{S} \models \alpha[s']$.

Observe that $\mathcal{S} \models \alpha \rightarrow \alpha_{x_0}$ and $\mathcal{S} \models \{\alpha_{x_0}\} <x_0 \,|\, E'> \{\beta\}$. We first prove that for all $y \in V_E$

$$Tr_{\mathcal{S}}, \left\{\{\alpha_x\}\, x\, \{\beta\} \mid x \in V_{E'}\right\} \vdash^{H_i} \{\alpha_y\}\, t'_y\, \{\beta\}.$$

Define Θ as $Tr_{\mathcal{S}} \cup \left\{\{\alpha_x\}\, x\, \{\beta\} \mid x \in V_{E'}\right\}$ and fix $x_1 \in V_{E'}$. We distinguish two cases:

1. For any summand pz of t'_{x_1} $(z \in V_{E'})$ we have $\Theta \vdash^{H_i} \{\alpha_{x_1}\} pz \{\beta\}$: We show that $\mathcal{S} \models \{\alpha_{x_1}\} p \{\alpha_z\}$ and because p is a closed process expression over Σ_i we conclude by the completeness of H_i that $Tr_{\mathcal{S}} \vdash^{H_i} \{\alpha_{x_1}\}\, p\, \{\alpha_z\}$ and thus $\Theta \vdash^{H_i} \{\alpha_{x_1}\}\, pz\, \{\beta\}$. Assume $\mathcal{S} \models \alpha_{x_1}[s]$ for some $s \in S$ and $(p, s) \xrightarrow{\nu}\!\!\!\!\rightarrow\ (\surd, s'')$. We derive $(<x_1 \mid E'>, s) \xrightarrow{\nu}\!\!\!\!\rightarrow\ (<z \mid E'>, s'')$. By construction of α_{x_1} there is an $s' \in S$ such that $\mathcal{S} \models \alpha[s']$ and $(<x_0 \mid E'>, s') \xrightarrow{\sigma}\!\!\!\!\rightarrow\ (<x_1 \mid E'>, s)$. So $(<x_0 \,|\, E'>, s') \xrightarrow{\sigma * \nu}\!\!\!\!\rightarrow\ (<z \,|\, E'>, s'')$, by which we conclude $\mathcal{S} \models \alpha_z[s'']$.

2. For any summand q of t'_{x_1} we have $\Theta \vdash^{H_i} \{\alpha_{x_1}\}\, q\, \{\beta\}$: We show that $\mathcal{S} \models \{\alpha_{x_1}\}\, q\, \{\beta\}$ and conclude that $\Theta \vdash^{H_i} \{\alpha_{x_1}\}\, q\, \{\beta\}$ by completeness of H_i. Assume $\mathcal{S} \models \alpha_{x_1}[s]$ for some $s \in S$ and $(q, s) \xrightarrow{\nu}\!\!\!\!\rightarrow\ (\surd, s'')$. We derive $(<x_1 \,|\, E'>, s) \xrightarrow{\nu}\!\!\!\!\rightarrow\ (\surd, s'')$. By construction of α_{x_1} there is an $s' \in S$ such that $\mathcal{S} \models \alpha[s']$ and $(<x_0 \,|\, E'>, s') \xrightarrow{\sigma}\!\!\!\!\rightarrow\ (<x_1 \,|\, E'>, s)$. So $(<x_0 \,|\, E'>, s') \xrightarrow{\sigma * \rho}\!\!\!\!\rightarrow\ (\surd, s'')$. Because $\mathcal{S} \models \{\alpha\} <x_0 \,|\, E'> \{\beta\}$ we have that $\mathcal{S} \models \beta[s'']$. We conclude $\mathcal{S} \models \{\alpha_{x_1}\}\, q\, \{\beta\}$.

By the completeness of H_i and *distributivity* we may conclude $\Theta \vdash^{H_i} \{\alpha_y\}\, t_y\, \{\beta\}$ for all $y \in V_E$, which is just the premiss for an application of the recursion rule in H_{i+1}, so $Tr_{\mathcal{S}} \vdash^{H_{i+1}} \{\alpha_{x_0}\} <x_0 \mid E> \{\beta\}$. Because $\alpha \rightarrow \alpha_{x_0} \in Tr_{\mathcal{S}}$ we derive $Tr_{\mathcal{S}} \vdash^{H_{i+1}} \{\alpha\} <x_0 \mid E> \{\beta\}$, which completes our proof. Note that if α is the empty predicate the lemma still holds. □

Theorem 5.3.4 *If the proof system H_i is complete, then the proof system H_{i+1} is complete.*

Proof. The soundness of H, and thus of H_{i+1} is proved in lemma 5.3.1. In the proof of lemma 5.3.2 we showed that the proof system H_0 was complete by induction on the structure of the process expression p involved in a partial correctness assertion $\{\alpha\}\, p\, \{\beta\}$. As the set $\mathcal{P}_{i+1}$ is also specified inductively, we only have to check one more 'basic clause' than in the proof of lemma 5.3.2, namely $p \equiv <x \mid E>$ with $E = \{x = t_x \mid x \in V_E\}$ a pure system over Σ_i. This has just been done in lemma 5.3.3. □

Corollary 5.3.5 *The proof system H is complete.*

5.4 Guarded systems and the proof system H

The notion of 'guardedness' is mostly defined more strictly than is done here in section 2.2 (see e.g. [2] and [3]). In order to discuss this restricted notion we will refer to it as follows: We call a recursive specification $E = \{x = t_x \mid x \in V_E\}$ *strictly guarded* if each variable in the expressions t_x is preceded by an atomic action $a \in A$. Let Σ_s denote the restriction of Σ obtained by considering only strictly pure systems, and $\mathcal{P}_s$ denote the corresponding set of closed process expressions. We define H_s by restricting the applicability of rule V of H to systems which are strictly pure. Of course H_s is still sound, as it is a subsystem of H. Since H contains no rules which decompose the process expression involved in a correctness formula, it follows that

$$Tr_{\mathcal{S}} \vdash^{H} \{\alpha\}\, p\, \{\beta\} \quad \Longrightarrow \quad Tr_{\mathcal{S}} \vdash^{H_s} \{\alpha\}\, p\, \{\beta\}$$

for all $p \in \mathcal{P}_s$, so H_s is also a complete proof system.

We further show that if we extend Σ with constants for the solutions of *all* guarded systems, then the proof system H with rule V applicable to all guarded systems, is not complete any more (and neither is H_s). If H is not sound with respect to this extension this is the case by definition. So assume that H preserves soundness. We show by an example that H cannot be complete:

Example. Consider the structure $\mathcal{S} = \langle \{a, b, c\}, \{s, s'\}, \mathit{action}, \mathit{effect} \rangle$ with the functions *action* and *effect* defined as follows: All atomic actions are inert with respect to the function *action*,

$$s(a) \stackrel{\text{def}}{=} s(b) \stackrel{\text{def}}{=} s'(c) \stackrel{\text{def}}{=} s' \quad \text{and} \quad s'(a) \stackrel{\text{def}}{=} s'(b) \stackrel{\text{def}}{=} s(c) \stackrel{\text{def}}{=} s.$$

Let σ and σ' be assertions such that σ is only satisfied by s and σ' only by s'. Consider the guarded system $E = \{X = aXb + c\}$. Now it is not difficult to see that $\mathcal{S} \models \{\sigma\}\, X\, \{\sigma\}$. Suppose that H is complete, and thus $Tr_{\mathcal{S}} \vdash \{\sigma\}\, X\, \{\sigma\}$. We may assume that the last two rules applied are V respectively IV (rule IV is the only rule not adding complexity to the process expression involved). So there must be α, β such that

$$Tr_{\mathcal{S}}, \{\{\alpha\}\, x\, \{\beta\}\} \vdash \{\alpha\}\, axb + c\, \{\beta\} \tag{1}$$

$$\sigma \rightarrow \alpha, \beta \rightarrow \sigma \in Tr_{\mathcal{S}}. \tag{2}$$

Now (1) implies that $Tr_{\mathcal{S}} \vdash \{\alpha\} c \{\beta\}$ and by (2) we derive $Tr_{\mathcal{S}} \vdash \{\sigma\} c \{\beta\}$. By the assumed soundness of H we conclude that $\sigma \rightarrow \beta \in Tr_{\mathcal{S}}$, so by (2) we have

$$\beta \leftrightarrow \sigma \in Tr_{\mathcal{S}}. \tag{3}$$

Also $Tr_S, \{\{\alpha\}\, x\, \{\beta\}\} \vdash \{\alpha\}\, axb\, \{\beta\}$, so there must be γ_1, γ_2 such that $\{\alpha\}\, a\, \{\gamma_1\}$, $\{\gamma_1\}\, x\, \{\gamma_2\}$ and $\{\gamma_2\}\, b\, \{\beta\}$ are derivable from $Tr_S \cup \{\{\alpha\}\, x\, \{\beta\}\}$. From the derivability of $\{\gamma_1\}\, x\, \{\gamma_2\}$ we conclude

$$\beta \to \gamma_2 \in Tr_S \tag{4}$$

and from the derivability of $\{\gamma_2\}\, b\, \{\beta\}$ and the soundness of H we conclude by (3) that $\gamma_2 \to \sigma' \in Tr_S$, so by (4) we have

$$\beta \to \sigma' \in Tr_S. \tag{5}$$

Now (3) and (5) are contradictory, so the proof system H is incomplete with respect to all guarded systems. (This holds as well for H_s, since E is a strictly guarded system.) □

6 Some extensions

6.1 Involving all guardedly specifiable processes

As shown in section 5.4 we cannot add constants for *all* guardedly specifiable processes to Σ without losing completeness of H. A solution to this problem is presented in [10]. The idea is to use a number of algebraic laws concerning the equality relation on process expressions. It can be proved that all structures considered respect these axioms, and any guardedly specified process is the solution of some pure system. By adding a proof rule ***substitution***, which permits interchangeability of (algebraically) equivalent process expressions in partial correctness assertions, one can prove a completeness result for all guardedly specifiable process expressions.

6.2 Involving silent actions

The constant τ, representing unobservable action, can be added to Σ without invalidating our completeness result. This is proved in [10]. The following semantical rules:

τ−laws : $(a,s) \xrightarrow{a(s)} (\tau, s(a))$ (if $a(s) \neq \delta$)

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

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

take care that τ satisfies the 'τ−laws of Milner': $x\tau = x$, $\tau x + x = \tau x$ and $a(\tau x + y) = a(\tau x + y) + ax$ (see [7]). We demand that τ is inert with respect to all structures considered. It should be mentioned that the definition of the effect rules for the π_n−operators in table 2 should be slightly changed, in this case.

Acknowledgements

I would like to thank Jan Bergstra for suggesting me the subject of this paper, and for his constructive help in writing it. For critical remarks, discussions and corrections I also thank Jos Baeten, Henk Goeman, Jan Friso Groote, Frits Vaandrager and in particular Fer-Jan de Vries, who suggested the format of the recursion rule of H, presented here.

References

[1] K.R. APT, *Ten Years of Hoare's Logic: A Survey — Part 1*, in: ACM Transactions on Programming Languages and Systems, Vol. 3, No. 4, 1981, pp. 431-483.

[2] J.C.M. BAETEN, J.A. BERGSTRA, *Global Renaming Operators over Concrete Process Algebra*, in: Inf. & Comp. 78(3), 1988, pp. 205-245.

[3] J.C.M. BAETEN, J.A. BERGSTRA, *Recursive Process Definitions with the State Operator*, in: Proceedings Computing Science in the Netherlands (SION), 1988, pp. 279-294.

[4] J.W DE BAKKER, J.N. KOK, J.-J.CH. MEYER, E.-R. OLDEROG, J.I. ZUCKER, *Contrasting themes in the semantics of imperative concurrency*, in: Current Trends in Concurrency (J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds.), LNCS 224, Springer-Verlag, 1986, pp. 51-121.

[5] J.A. BERGSTRA, J.W. KLOP, *Process Algebra: Specification and Verification in Bisimulation Semantics*, in: Mathematics and Computer Science II, CWI monograph 4 (M. Hazewinkel, J.K. Lenstra, L.G.L.T. Meertens, eds.), North-Holland, Amsterdam, 1986, pp. 61-94.

[6] R.J. VAN GLABBEEK, *Bounded Nondeterminism and the Approximation Induction Principle in Process Algebra*, in: Proceedings STACS 87 (F.J. Brandenburg, G. Vidal-Naquet, M. Wirsing, eds.), LNCS 247, Springer-Verlag, 1987, pp. 336-347.

[7] A.J.R.G. MILNER, *A Calculus of Communicating Systems*, LNCS 92, Springer-Verlag, 1980.

[8] D.M.R. PARK, *Concurrency and automata on infinite sequences*, in: Proceedings 5th GI Conference (P. Deussen, ed.), LNCS 104, Springer-Verlag, 1981, pp. 167-183.

[9] G.D. PLOTKIN, *An Operational Semantics for CSP*, in: Formal Description of Programming Concepts-II (D. Bjørner, ed.), North-Holland, Amsterdam, 1983, pp. 199-223.

[10] A. PONSE, *Process algebra and Hoare's logic*, Note CS-N8802, Centrum voor Wiskunde en Informatica, Amsterdam, 1988.