INFORMATION AND COMPUTATION 95, 192–217 (1991)

Process Expressions and Hoare's Logic: Showing an Irreconcilability of Context-Free Recursion with Scott's Induction Rule

ALBAN PONSE*,†

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

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 and sequential composition. Furthermore it provides a mechanism for specifying processes recursively (including nested recursion). We consider processes as having a state: atomic actions are to be specified in terms of observable behaviour (relative to initial states) and state transformations. Any process having some initial state can be associated with a *transition system* representing all possible courses of execution. This leads to an operational semantics in the style of Plotkin. The *partial correctness assertion* $\{\alpha\}\ p\{\beta\}$ expresses that for any transition system associated with the process p and having some initial state satisfying α, its final states representing successful execution satisfy β. A logic in the style of Hoare, containing a proof system for deriving partial correctness assertions, is presented. This proof system is sound and relatively complete, so any partial correctness assertion can be evaluated by investigating its derivability. Included is a short discussion about the extension of the process language with "guarded recursion." It appears that such an extension violates the completeness of the Hoare logic. This reveals a remarkable property of Scott's induction rule in the context of non-determinism: only regular recursion allows a completeness result. © 1991 Academic Press, Inc.

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. In this paper we consider processes specifiable in a simple "process language" for

* The author received full support from the European Communities under ESPRIT Project 432, An Integrated Formal Approach to Industrial Software Development (METEOR).

† E-mail address: alban@cwi.nl

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 and sequential composition. It also provides a mechanism for specifying processes as the solution of a system of (recursive) equations satisfying some syntactical criteria. 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. By giving a calculus for deriving transitions (so called action rules), every closed process expression can be associated with a transition system that represents all possible courses of execution. In van Glabbeek (1987) this is worked out for a larger process language containing also a construct for the specification of concurrent (communicating) processes.

Here we take a less abstract point of view and use a *state space* for modelling executions. Let S be a set of states. The execution of an atomic action a in a state $s \in S$ is to be specified by functions *action* and *effect*. The expression $action(a, s)$ denotes what can be observed if a is executed in state s; with $effect(s, a)$ we denote the resulting state. If this execution terminates successfully (which is observable), 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 (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 (cf. Plotkin, 1983). Referring to de Bakker *et al.* (1986), our process language can be classified as *non-uniform*.

Most of this paper is about *partial correctness assertions*. Let α and β denote 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 starts in an initial state satisfying α and 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 implies 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 investigating for any s satisfying $\alpha(s)$ whether there is a transition $(a, s) \longrightarrow (\surd, s')$, and if so whether $\beta(s')$ holds.

Main Results. 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 Apt, 1981, 1984). 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 sound and 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 derivation or showing the absence of these may then replace the construction of a (possibly large) number of (complex) transition systems.

Furthermore it is shown by an example that the completeness of the Hoare logic is lost if our language would allow "guarded recursion." This type of recursion permits a more liberal format of the defining equations, only demanding the recursion variables in the defining terms to be preceded by an atomic action (a "guard"). The incompleteness is then caused by the nature of "Scott's induction rule," used for the introduction of recursively specified process expressions in partial correctness assertions (see e.g. de Bakker (1980) for an introduction to Scott's induction rule). It appears that only regular recursion allows a completeness result in the context of non-determinism.

Contents of the Paper. Section 2 is dedicated to the introduction of our process language. In Section 3 we discuss the way atomic actions having a state are to be specified and we present a calculus for deriving 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 specific "language of assertions," that is suitable to formulate any partial correctness assertion. Next, in Section 5, we present a proof system for deriving partial correctness assertions, which is shown to be sound and relatively complete. Techniques concerning the construction of derivations or proofs of their absence are not discussed. We end this section with an exposure of the mentioned example on incompleteness. The paper is concluded with a short discussion on some extensions.

Note. The process language introduced here refers to a basic 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 semantic approach. However, it can be easily proved that the semantical notions discussed in this paper satisfy the relevant ACP axioms. ACP is discussed in, e.g., Baeten and Weijland (1990), or Bergstra and Klop (1986).

2. Process Expressions

Let A be some countable, nonempty set of atomic actions with typical elements $a, b, c, \ldots$. We introduce a language, having A as a parameter, in which processes can be specified. Let $V = \{x, y, z, \ldots\}$ be a countable 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. *Recursion-Free Processes*

The signature Σ_0 can be used to specify finite, recursion-free processes. For each atomic action $a \in A$ there is a constant $a \in \Sigma_0$ and there is a special constant δ called *deadlock* ($\delta \notin A$). There are two binary functions in Σ_0, $+$ (sum) and $\cdot$ (product). A *process expression* over Σ_0 is just a term over this signature possibly containing variables from V. We mostly leave out the symbol $\cdot$ in process expressions and take $\cdot$ to be most binding of all operators and $+$ to be stronger binding than $+$, e.g., $xy + z$ means $(x \cdot y) + z$.

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

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

So $\mathscr{P}_0$ is the set of *closed* process expressions over Σ_0 and every element of $\mathscr{P}_0$ specifies a *recursion-free process.*

TABLE 1

The Signature Σ^+

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:	$\langle x \mid E \rangle$	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, $n \in \mathbb{N}$

Some intuitions: "deadlock" is the acknowledgment of a process that it has no possibility to do anything any more; $p+q$ represents the process which first makes a choice between its summands p and q, and then proceeds with the chosen course of action; pq represents the process p, followed after possible successful termination of p by q. The process p does not terminate successfully if it ends in deadlock.

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 àutomaton 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:

$\mathtt{in}$	(insert a coin)
$\mathtt{p}_\mathtt{a}$, $\mathtt{p}_\mathtt{b}$	(push one of the buttons a or b respectively)
$\mathtt{pr}$	(push the button for restitution and collect the returned coin)
$\mathtt{co}$	(collect the delivery of the automaton)
$\mathtt{st}$	(stop behaving as a user).

The recursive specification

$$x = \mathtt{in}((\mathtt{p}_\mathtt{a} + \mathtt{p}_\mathtt{b})\,\mathtt{co} + \mathtt{pr})\,x + \mathtt{st}$$

expresses that the left-hand side of this equation is specified recursively by the right-hand side. A user behaviour X satisfying this specification consists of

either inserting a coin or stopping the behaviour;

in case of the first atomic action, pushing a button for one of the drinks or handling restitution;

in case of the first alternative, collecting the delivery of the automaton, then restarting 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 (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 any of the semantics to be introduced, 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. Let $E = \{x = t_x \mid x \in V_E\}$ be a recursive specification over Σ_i.

1. We call E *guarded* iff each occurrence of a variable y in the expressions t_x occurs in a subterm $p \cdot M$ with $p \in \mathscr{P}_i$. We speak of *guarded systems* instead of guarded recursive specifications.[1]

2. We say that E is *pure* iff for any subterm $M \cdot N$ occurring in any 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 that are pure, we can prove our completeness result. Remark that the specification of X 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. Solutions of pure systems.

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 $\{\langle x \mid E\rangle \mid x \in V_E\}$, where $\langle x \mid E\rangle$ 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.

A process expression over Σ_i is a term over this signature possibly containing variables from V. For instance $\langle x \mid \{x = \langle y \mid \{y = ay + b\}\rangle x + c\}\rangle$ is a closed process expression over Σ (even over Σ_2), but $\langle x \mid \{x = axa + aa\}\rangle$ 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 $\langle t \mid E\rangle$ denotes the process

[1] In Section 5.3 we return to this definition of guardedness.

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

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

- $A_\delta \subseteq \mathscr{P}$,
- if $p, q \in \mathscr{P}$, then $p + q \in \mathscr{P}$ and $pq \in \mathscr{P}$,
- if $E = \{x = t_x \mid x \in V_E\}$ is a pure system over Σ, then $\{\langle x \mid E \rangle \mid x \in V_E\} \subseteq \mathscr{P}$,

and $\mathscr{P}_{i+1}$ defined likewise by only considering pure systems over Σ_i. Any element of $\mathscr{P}$ ($\mathscr{P}_{i+1}$ respectively) containing constants of the form $\langle x \mid E \rangle$ defines 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 transforms a process into deadlock after it has performed n atomic actions: e.g., $\pi_1(ab)$ behaves as the process $a\delta$ and $a \cdot \pi_2(b)$ behaves like ab.

Let $\mathscr{P}^+$ denote the set of closed process expressions over this signature, so $\mathscr{P}^+$ is the least set satisfying

- $\mathscr{P} \subseteq \mathscr{P}^+$,
- if $p, q \in \mathscr{P}^+$ and $n \in \mathbb{N}$, then $p + q \in \mathscr{P}^+$, $pq \in \mathscr{P}^+$ and $\pi_n(p) \in \mathscr{P}^+$.

Remark that the π_n-operators do not occur in recursive specifications.

3. Processes Having States

In this section we discuss the specification of atomic actions having states in terms of observable behaviour and state transformations. Furthermore we introduce a calculus for deriving transition systems concerning the behaviour of processes having a state.

3.1. *Structures*

Let S be a nonempty set of states, with typical elements $s, s', \ldots$. The "state labelled process expression" (p, s) denotes the process p having 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 functions

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

which determine the relation between elements $a \in A$ and elements s of S, the set of states. The functions *action* and *effect* were introduced in Baeten and Bergstra (1988). By *action*(a, s) we denote the activity which can be observed when a is executed in state s; by *effect*(s, a) we denote the state resulting from this execution. The case *action*$(a, s) = \delta$ describes the (observable) situation in which a in s cannot be executed successfully. In this case it is not necessary that *effect*(s, a) have a special value (e.g., s or some special "error state"), for we will be only interested in states resulting from successful executions. 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 here introduce the following

Abbreviation. We write $a(s)$ instead of *action*(a, s), and $s(a)$ instead of *effect*(s, a).

An atomic action a is called *inert* iff $\forall s \in S(a(s) = a)$ and $\forall s \in S(s(a) = s)$. The function *action* is said to be *inert* iff for all $a \in A$ it satisfies $\forall s \in S(a(s) = a)$.

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* $\langle S, action, effect \rangle$ is a triple containing a nonempty set S of states and functions *action*: $A \times S \to A_\delta$ and *effect*: $S \times A \to S$.

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

3.2. *Transition Systems*

Let $\mathcal{S} = \langle S, action, effect \rangle$ be some structure. We introduce for all $a \in A$ a binary relation $\xrightarrow{a}$ over state labelled process expressions. In general we mean by a *transition* $(p, s) \xrightarrow{a} (p', s')$ that by performing an action a the process p in state s can evolve into p' in state s'. To represent successful termination we introduce a special element $\surd$ not in Σ^+ and for all $a \in A$ a

[2] As an example think of the representation of a program in a high level language as *Pascal* in our process language. Assume that a variable X is declared as an integer and S denotes the set of valuations for all declared integer variables. Let s represent some valuation and a the assignment $x := x + 1$; then $a' = a$ if $s(x) < \text{MAXINT}$ and δ otherwise. Of course $s'(x) = s(x) + 1$ and $s'(y) = s(y)$ for any integer variable $y \not\equiv x$.

relation $\xrightarrow{a} (\surd, \cdot)$ between state labelled process expressions and states: the expression $(p, s) \xrightarrow{a} (\surd, s')$ denotes that the process p 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. In Table 2 we present a proof system, the *effect rules*, suitable to derive transitions by means of *substitutions*, where a substitution θ is a mapping from the variables of V to process expressions over Σ^+. By defining $\theta(f(t_1, ..., t_k))$ as $f(\theta(t_1), ..., \theta(t_k))$ $(k \geqslant 0)$ we extend the domain of any substitution θ to the set of all process expressions over Σ^+. Now $(\theta(t), s) \xrightarrow{a} (\theta(t'), s')$ is a *derivable* transition iff it is the conclusion of an effect rule of which the θ-instance of its premiss is also derivable (note that the "$a \in A$"-rules have empty premises).

Remark that any structure fixes the effect rules. We define $\overset{\sigma}{\twoheadrightarrow}$ for $\sigma \in A^*$ as the reflexive and transitive closure of all transition relations:

- $(x, s) \overset{\lambda}{\twoheadrightarrow} (x, s)$ (λ denoting the empty string over A^*)
- $$\frac{(x, s) \overset{\sigma}{\twoheadrightarrow} (x', s')(x', s') \xrightarrow{a} (x'', s'')}{(x, s) \overset{\sigma a}{\twoheadrightarrow} (x'', s'')}$$

$$\frac{(x, s) \overset{\sigma}{\twoheadrightarrow} (x', s')(x', s') \xrightarrow{a} (\surd, s'')}{(x, s) \overset{\sigma a}{\twoheadrightarrow} (\surd, s'')} \qquad (a \in A).$$

TABLE 2

Effect Rules

$a \in A$:	$(a, s) \xrightarrow{a(s)} (\surd, s(a))$ 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:	$\dfrac{(\langle t_x \mid E \rangle, s) \xrightarrow{a} (y, s')}{(\langle x \mid E \rangle, s) \xrightarrow{a} (y, s')}$	$\dfrac{(\langle t_x \mid E \rangle, s) \xrightarrow{a} (\surd, s')}{(\langle x \mid E \rangle, s) \xrightarrow{a} (\surd, s')}$
π_n:	$\dfrac{(x, s) \xrightarrow{a} (x', s')}{(\pi_{n+1}(x), s) \xrightarrow{a} (\pi_n(x'), s')}$	$\dfrac{(x, s) \xrightarrow{a} (\surd, s')}{(\pi_{n+1}(x), s) \xrightarrow{a} (\surd, s')}$

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). *Let $\mathscr{S} = \langle S, action, effect \rangle$ be fixed. If for some string $\sigma \in A^*$, $s \in S$ we have $(tt', s) \xrightarrow{\sigma}\!\!\rightarrow (\surd, s')$, then there are σ_1, σ_2 and s'' such that $\sigma_1\sigma_2 \equiv \sigma$, $(t, s) \xrightarrow{\sigma_1}\!\!\rightarrow (\surd, s'')$ and $(t', s'') \xrightarrow{\sigma_2}\!\!\rightarrow (\surd, s')$.*

Proof. By induction on the length of σ (first proving an intermediate result). ∎

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. A graph $\mathscr{G}$ is a transition system for (p, s) iff

1. $\mathscr{G}$ contains a node labelled with (p, s), we call this node the *root* of $\mathscr{G}$.

2. (p', s') is a node of $\mathscr{G}$, and $(p', s') \xrightarrow{a} (p'', s'')$ is a derivable transition, then (p'', s''), is a node of $\mathscr{G}$ and there is an arc labelled with a from (p', s') to (p'', s'').

3. (p', s') is a node of $\mathscr{G}$ and $(p', s') \xrightarrow{a} (\surd, s'')$ is a derivable transition, then $(\surd, s'')$ is a node of $\mathscr{G}$ and there is an arc labelled with a from (p', s') to $(\surd, s'')$.

Any such graph $\mathscr{G}$ will be referred to as $ts((p, s))$. The state s will be called the *initial* state of $ts((p, s))$ and any state s' such that $(\surd, s')$ is a node in $\mathscr{G}((p, s))$ will be called a *final* state of $ts((p, s))$. We return to our running example:

EXAMPLE (*continued*). Concerning the recursive specification $x = \mathtt{in}((\mathtt{p_a} + \mathtt{p_b})\,\mathtt{co} + \mathtt{p\dot{r}})\,x + \mathtt{st}$ we consider a user having initially $N+1$ coins and no drinks. Take $\mathscr{S} = \langle S, action, effect \rangle$ with each state of S being a pair of counters that 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* as

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

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

$$\mathtt{co}((i, j)) \overset{\text{def}}{=} \begin{cases} \mathtt{co} & \text{if } (i, j) \in Busy \\ \delta & \text{otherwise} \end{cases}$$

$$(i, j)(\mathtt{co}) \overset{\text{def}}{=} \begin{cases} (i, j+1) & \text{if } (i, j) \in \mathit{Busy} \\ (i, j) & \text{otherwise} \end{cases}$$

$$\mathtt{pr}((i, j)) \overset{\text{def}}{=} \begin{cases} \mathtt{pr} & \text{if } (i, j) \in \mathit{Busy} \\ \delta & \text{otherwise} \end{cases}$$

$$(i, j)(\mathtt{pr}) \overset{\text{def}}{=} \begin{cases} (i+1, j) & \text{if } (i, j) \in \mathit{Busy} \\ (i, j) & \text{otherwise} \end{cases}$$

and the atomic actions $\mathtt{p}_\mathtt{a}$, $\mathtt{p}_\mathtt{b}$ and $\mathtt{st}$ inert. Assume $N = 0$. Figure 1 contains a transition system for the process X in initial state $(1, 0)$, representing the behaviour of a user having 1 coin. (*to be continued*)

4. Semantics

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

4.1. *An Operational Semantics*

Let $\mathscr{S} = \langle S, \mathit{action}, \mathit{effect} \rangle$ be some structure. The idea is that two closed process expressions p and q are *operationally equivalent in* $\mathscr{S}$ iff 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 Park, 1981):

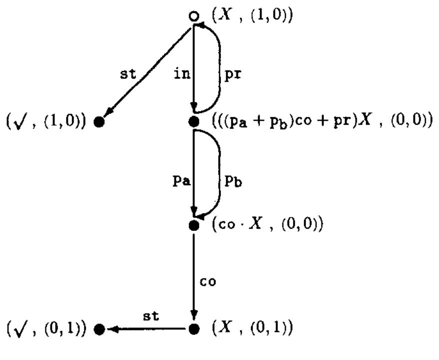

FIG. 1. A transition system.

DEFINITION 4.1.1. Let $\mathscr{S} = \langle S, \mathit{action}, \mathit{effect} \rangle$ be fixed. A binary relation $R \subseteq (\mathscr{P}^+ \times S) \times (\mathscr{P}^+ \times S)$ is a *bisimulation* iff the following conditions are satisfied ($a \in A$):

1. If $(p, s)\, R(q, s)$ and $(p, s) \xrightarrow{a} (p', s')$, then $\exists (q', s')$ 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 $\exists (p', s')$ 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' iff $(q, s) \xrightarrow{a} (\surd, s')$.

Two transition systems $ts((p, s))$ and $ts((q, s))$ are *bisimilar*, we write $ts((p, s)) \leftrightarrow ts((q, s))$, iff 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 $\leftrightarrow$ 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. Let $\mathscr{S} = \langle S, \mathit{action}, \mathit{effect} \rangle$ be fixed.

1. A closed process expression p is interpreted in $\mathscr{S}$ as $\{[(p, s)] \mid s \in S\}$.

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

Remark that if we want to consider a structure $\mathscr{S} = \langle S, \mathit{action}, \mathit{effect} \rangle$ 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 $\mathscr{S} \models a =_{\text{op}} b$. This reflects the circumstance that *in* $\mathscr{S}$ the constants a and b apparently denote the same atomic action. We finally prove that for any structure $\mathscr{S}$ the relation $=_{\text{op}}$ 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 $\mathscr{S}$ the relation $=_{\text{op}}$ is a congruence with respect to the operators involved.*

Proof. Fix $\mathscr{S}$ and assume $\mathscr{S} \models p =_{\text{op}} p'$, $\mathscr{S} \models q =_{\text{op}} q'$. We have to show $\mathscr{S} \models p \square q =_{\text{op}} p' \square q'$ for $\square \in \{+, \cdot\}$ and $\mathscr{S} \models \pi_n(p) =_{\text{op}} \pi_n(p')$ for all $n \in \mathbb{N}$. As an example we consider sequential composition: suppose that for any $s \in S$ we have $ts((p, s)) \leftrightarrow ts((p', s))$ by a bisimulation $R_{(p,s)}$ and $ts((q, s)) \leftrightarrow ts((q', s))$ by a bisimulation $R_{(q,s)}$. Fix $s_0 \in S$. We define a relation R as follows:

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

We have $(pq, s_0)\, R(p'q', s_0)$ and we show that R is a bisimulation. Suppose $(t, s)\, R(t', s)$ and $(t, s) \xrightarrow{a} (u, s')$. In case $(t, s)\, R_{(q, s'')}(t', s)$ for some s'' we are done, so assume $t \equiv rq$ and $t' \equiv r'q'$. Now $(r, s)\, R_{(p, s_0)}(r', s)$ and by inspection of the effect rules we find two possibilities for u:

$u \equiv vq$. In this case it must be that $(r, s) \xrightarrow{a} (v, s')$. Because $R_{(p, s_0)}$ is a bisimulation satisfying $(r, s)\, R_{(p, s_0)}(r', s)$ there is (v', s') such that $(r', s) \xrightarrow{a} (v', s')$ and $(v, s')\, R_{(p, s_0)}(v', s')$. We derive $(r'q', s) \xrightarrow{a} (v'q', s')$ and by definition of R also $(vq, s')\, R(v'q', s')$.

$u \equiv q$. In this case it must be that $(r, s) \xrightarrow{a} (\surd, s')$. Because $R_{(p, s_0)}$ is a bisimulation satisfying $(r, s)\, R_{(p, s_0)}(r', s)$ there is s' such that we have $(r', s) \xrightarrow{a} (\surd, s')$. We derive $(r'q', s) \xrightarrow{a} (q', s')$ and by definition of R also $(q, s')\, R(q', s')$. ∎

4.2. *A Partial Correctness Semantics*

We introduce a logical language $\mathscr{L}$, a *language of assertions*, in order to reason formally about any structure. Let *Pred* be some set of unary predicate symbols. We define $\mathscr{L}$ as follows:

one variable:	v	
unary function symbols:	$effect_a$	(for all $a \in A$)
unary predicate symbols:	$stop_a$	(for all $a \in A$)
unary predicate symbols:	P	(for all $P \in Pred$)
connectives:	$\neg, \bigvee, \bigwedge, \rightarrow, \leftrightarrow$	
auxiliary symbols:	), ,, (	

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

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

DEFINITION 4.2.1. Syntax of partial correctness assertions and correctness formulae.

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

2. A *correctness formulae over* $\mathscr{L}$ 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 use the more general concept of "correctness formulae" to define a proof system in which we can derive partial correctness assertions concerning recursively specified processes. Though this proof system is only suitable to derive partial correctness assertions concerning process expressions over Σ, we use the π_n-operators of Σ^+ in proving its soundness.

We now turn to the semantics of assertions and correctness formulae. Let $\mathcal{S} = \langle S, action, effect\rangle$ be some fixed structure. We define an interpretation $\mathcal{I}$ of $\mathcal{L}$ in $\mathcal{S}$ by fixing a set $\{\bar{P} \mid P \in Pred\}$ of unary predicates over S. Assertions are interpreted as

$$\begin{array}{ll} \mathcal{S} \models_{\mathcal{I}} \alpha & \text{iff } \forall s \in S(\mathcal{S} \models_{\mathcal{I}} \alpha[s]) \\ \mathcal{S} \models_{\mathcal{I}} \alpha(effect_a(v))[s] & \text{iff } \forall s \in S(\mathcal{S} \models_{\mathcal{I}} \alpha[effect(s, a)]) \\ \mathcal{S} \models_{\mathcal{I}} stop_a[s] & \text{iff } action(a, s) = \delta \\ \mathcal{S} \models_{\mathcal{I}} P[s] & \text{iff } \bar{P}(s) \text{ holds} \end{array}$$

and compound formulae as usual; e.g., $\mathcal{S} \models_{\mathcal{I}} \alpha \to \beta[s]$ iff $\mathcal{S} \models_{\mathcal{I}} \alpha[s] \Rightarrow \mathcal{S} \models_{\mathcal{I}} \beta[s]$.

We write $Tr_{\mathcal{S}}$ for the set of all true assertions in $\mathcal{L}$, so $\alpha \in Tr_{\mathcal{S}}$ iff $\mathcal{S} \models_{\mathcal{I}} \alpha$, i.e., iff $\forall s \in S$ $(\mathcal{S} \models_{\mathcal{I}} \alpha[s])$.

Before defining the way correctness formulae are interpreted we introduce some more notation: let t be some process expression over Σ^+ and $\bar{x}$ a sequence of variables. We write $t = t(\bar{x})$ to indicate that all variables occurring in t are *among* the elements of the sequence $\bar{x}$. If $\bar{p}$ is a sequence of elements of $\mathcal{P}^+$ (the set of closed process expressions over Σ^+), then $t(\bar{p})$ denotes the closed process expression obtained by replacing all variables in t by the corresponding $\bar{p}$-elements. We write "$\forall \bar{p}$" if we want to consider all sequences of length $\bar{x}$ over $\mathcal{P}^+$.

DEFINITION 4.2.2. Let $\mathcal{S}$, $\mathcal{L}$ be fixed and $\mathcal{I}$ an interpretation of $\mathcal{L}$ in $\mathcal{S}$.

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

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

2. A correctness formula $\{\alpha\}\ t(\bar{x})\ \{\beta\}$ over $\mathcal{L}$ is *true in $\mathcal{S}$ under $\mathcal{I}$*, we write $\mathcal{S} \models_{\mathcal{I}} \{\alpha\}\ t\ \{\beta\}$, iff

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

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. Let $\mathcal{S}$, $\mathcal{L}$ be fixed and $\mathcal{I}$ an interpretation of $\mathcal{L}$ in $\mathcal{S}$. We call two closed process expressions p and q over Σ^+ *equivalent under partial correctness in* $\mathcal{S}$, $\mathcal{L}$, $\mathcal{I}$, we write $\mathcal{S} \models_{\mathcal{I}} p =_{\text{pc}} q$, iff for all $\mathcal{L}$-assertions α, β we have

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

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

THEOREM 4.2.4. *For all* $\mathcal{S}$, $\mathcal{L}$, $\mathcal{I}$ *the relation* $=_{\text{pc}}$ *is a congruence with respect to the operators involved.*

Proof. Let $\mathcal{S}$, $\mathcal{L}$ be fixed, $\mathcal{I}$ an interpretation of $\mathcal{L}$ in $\mathcal{S}$ and assume $\mathcal{S} \models_{\mathcal{I}} p =_{\text{pc}} p'$, $\mathcal{S} \models_{\mathcal{I}} q =_{\text{pc}} q'$. We have to show $\mathcal{S} \models_{\mathcal{I}} p \,\Box\, q =_{\text{pc}} p' \,\Box\, q'$ for $\Box \in \{+, \cdot\}$ and $\mathcal{S} \models_{\mathcal{I}} \pi_n(p) =_{\text{pc}} \pi_n(p')$ for all $n \in \mathbb{N}$. As an example we consider alternative composition: it is sufficient to show that if we have a reduction $(p+q, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')$, then $(p'+q', s) \xrightarrow{\rho}\!\!\!\!\rightarrow (\surd, s')$. This follows easily: suppose the first transition in our reduction, say $(p+q, s) \xrightarrow{a} (r, s'')$, is a consequence of $(p, s) \xrightarrow{a} (r, s'')$, then $(p, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')$ is also derivable. By assumption we have $(p', s) \xrightarrow{\rho}\!\!\!\!\rightarrow (\surd, s')$ for some string $\rho \in A^*$, so using the first transition of this reduction we derive $(p'+q', s) \xrightarrow{\rho}\!\!\!\!\rightarrow (\surd, s')$. ▌

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

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

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

LEMMA 4.2.5 (distributivity). *Let* $\mathcal{S}$, $\mathcal{L}$, $\mathcal{I}$ *be fixed. For all* t, t', t'' *over* Σ^+ *we have*

$$\mathcal{S} \models_{\mathcal{I}} t(t'+t'') =_{\text{pc}} tt' + tt'' \qquad \textit{and} \qquad \mathcal{S} \models_{\mathcal{I}} (t+t')\,t'' =_{\text{pc}} tt'' + t't''.$$

Proof. By induction on the length of derivations, using *decomposition* (see Lemma 3.2.1). ▌

Because the relation $=_{\text{pc}}$ identifies in particular closed process expressions having bisimilar transition systems, we have the following correspondence with the relation $=_{\text{op}}$:

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

The converse does not hold: as an example consider a structure $\mathcal{S} = \langle \{s\}, \mathit{action}, \mathit{effect} \rangle$ with the atomic action a inert. Marking roots with the symbol $\downarrow$, the transition system

$$\begin{tikzcd} \overset{\downarrow}{(a(a+\delta), s)} \arrow[r, "a"] & (a+\delta, s) \arrow[r, "a"] & (\surd, s) \end{tikzcd}$$

represents the process expression $a(a+\delta)$ in $\mathcal{S}$, and the transition system

$$\begin{tikzcd} (\delta, s) & \overset{\downarrow}{(aa+a\delta, s)} \arrow[l, "a"'] \arrow[r, "a"] & (a, s) \arrow[r, "a"] & (\surd, s) \end{tikzcd}$$

represents the process expression $aa + a\delta$. Now these transition systems are not bisimilar, for the latter contains the "deadlock state" (δ, s), which cannot be related by a bisimulation to any state present in the upper transition system. We conclude $\mathcal{S} \not\models a(a+\delta) =_{\mathrm{op}} aa + a\delta$, whereas by *distributivity* we have $\mathcal{S} \models_{\mathcal{I}} a(a+\delta) =_{\mathrm{pc}} aa + a\delta$ for all $\mathcal{L}$, $\mathcal{I}$.

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 $\mathcal{S} = \langle S, \mathit{action}, \mathit{effect} \rangle$ be some fixed structure and $\mathcal{L}$ a language of assertions such that *Pred* contains exactly one predicate symbol for each subset of S. We write in this case $\mathcal{L}_{\mathcal{S}}$, the language of assertions about S, and we interpret the symbols of *Pred* as the corresponding predicates over S. We omit the subscript $\mathcal{I}$ when interpreting assertions of $\mathcal{L}_{\mathcal{S}}$.

5. DERIVING PARTIAL CORRECTNESS ASSERTIONS

In the following we consider various proof systems in a natural deduction format (see e.g., van Dalen, 1983), suitable to derive correctness formulae, and in particular correctness assertions. We use symbols Φ, Φ', ... as syntactical variables for (possibly empty) sets of correctness formulae.

Let $\mathcal{S} = \langle S, \mathit{action}, \mathit{effect} \rangle$ be some structure, $\mathcal{L}$ a language of assertions, $\mathcal{I}$ an interpretation of $\mathcal{L}$ in $\mathcal{S}$, and G some proof system. We write $Tr_{\mathcal{S}}, \Phi \vdash^{G} \phi$ if there is a derivation of a correctness formula ϕ in G which uses hypotheses from $Tr_{\mathcal{S}}$ and Φ. If G is known from the context we omit the superscript G in $\vdash^{G}$. A partial correctness assertion $\{\alpha\}\ p\ \{\beta\}$ over $\mathcal{L}$ is called *derivable* iff $Tr_{\mathcal{S}} \vdash^{G} \{\alpha\}\ p\ \{\beta\}$ (so $\Phi = \varnothing$). 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 need not write here $\Sigma_G \subseteq \Sigma^+$, for the π_n-operators will never occur in derivable partial correctness assertions.

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

$$Tr_{\mathscr{S}}, \Phi \vdash\!\!-\; \{\alpha\}\, t\, \{\beta\} \quad\Rightarrow\quad \forall \bar{p}[\mathscr{S} \models_{\mathscr{I}} \Phi(\bar{p}) \quad\Rightarrow\quad \mathscr{S} \models_{\mathscr{I}} \{\alpha\}\, t(\bar{p})\, \{\beta\}],$$

where $\Phi(\bar{p}) \overset{\text{def}}{=} \{\{\alpha\}\, t(\bar{p})\, \{\beta\} \mid \{\alpha\}\, t\, \{\beta\} \in \Phi\}$. So in particular any derivable partial correctness assertion is true in $\mathscr{S}$ under $\mathscr{I}$.

The proof system G is (*relatively*) *complete* iff for any structure $\mathscr{S}$ *and* its language of assertions $\mathscr{L}_{\mathscr{S}}$ the converse holds for all partial correctness assertions $\{\alpha\}\, p\, \{\beta\}$ over $\mathscr{L}_{\mathscr{S}}$, where p is a *closed* process expression over Σ:

$$\mathscr{S} \models \{\alpha\}\, p\, \{\beta\} \qquad\Rightarrow\qquad Tr_{\mathscr{S}} \vdash\!\!-^{G} \{\alpha\}\, p\, \{\beta\}.$$

The adjective "relatively" refers to the fact that $Tr_{\mathscr{S}}$ may be used in derivations: *relative* to all true assertions about $\mathscr{S}$, we have that truth in $\mathscr{S}$ implies derivability.

5.1. *The Proof System H*

In Table 3 we present the proof system H associated with Σ. Rule VI, known as Scott's induction rule (see, e.g., de Bakker, 1980), should be used

TABLE 3

The Proof System H

I	axiom	$\{\alpha\}\, \delta\, \{\beta\}$
II	axioms ($a \in A$)	$\dfrac{\neg\, \text{stop}_a(v) \wedge \alpha(v) \rightarrow \beta(\text{effect}_a(v))}{\{\alpha\}\, a\, \{\beta\}}$
III	alternative composition	$\dfrac{\{\alpha\}\, t\, \{\beta\} \quad \{\alpha\}\, t'\, \{\beta\}}{\{\alpha\}\, t + t'\, \{\beta\}}$
IV	sequential composition	$\dfrac{\{\alpha\}\, t\, \{\beta\} \quad \{\beta\}\, t'\, \{\gamma\}}{\{\alpha\}\, tt'\, \{\gamma\}}$
V	consequence	$\dfrac{\alpha \rightarrow \alpha' \quad \{\alpha'\}\, t\, \{\beta'\} \quad \beta' \rightarrow \beta}{\{\alpha\}\, t\, \{\beta\}}$
VI	Scott's induction rule	If $E = \{x = t_x \mid x \in V_E\}$ is a pure system, then $[\{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}]$ $\vdots$ $\dfrac{\{\alpha_y\}\, t_y\, \{\beta_y\}}{\{\alpha_z\}\, \langle z \mid E\rangle\, \{\beta_z\}}\quad \begin{array}{l}\text{for all } \; y \in V_E \\ z \in V_E\end{array}$

as follows: if $E = \{x = t_x \mid x \in V_E\}$ is a pure system, α_x, β_x $(x \in V_E)$ are assertions, and

> for any $y \in V_E$ the partial correctness formula $\{\alpha_y\}\ t_y\ \{\beta_y\}$ is derived from a set of hypotheses Γ_y containing no other correctness formulae 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 assertion $\{\alpha_z\}\ \langle z \mid E\rangle\ \{\beta_z\}$ can be derived from $\bigcup_{x \in V_E} \Gamma_x - \{\{\alpha_x\}\ x\ \{\beta_x\} \mid x \in V_E\}$.

In other words, any hypothesis in $\{\{\alpha_x\}\ x\ \{\beta_x\} \mid x \in V_E\}$ is *cancelled* after the application of *Rule* VI. This is indicated by the square brackets. We show by our running example how to use H (see also Fig. 2):

EXAMPLE (*continued*). Concerning the recursive specification $x = \mathtt{in}((\mathtt{p_a} + \mathtt{p_b})\,\mathtt{co} + \mathtt{pr})\,x + \mathtt{st}$ we introduced a user having initially $N+1$ coins and no drinks, and the structure $\mathcal{S} = \langle S, \mathit{action}, \mathit{effect}\rangle$ with $S = \mathit{Busy} \cup \mathit{Final}$, 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:

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

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

$$\{\mathtt{p_a}{}^{\mathit{busy}}_{\mathit{busy}}, \mathtt{p_b}{}^{\mathit{busy}}_{\mathit{busy}}, \mathtt{co}^{\mathit{busy}}_{\mathit{final}}, \mathtt{pr}^{\mathit{busy}}_{\mathit{final}}, \mathtt{in}^{\mathit{final}}_{\mathit{busy}}, \mathtt{st}^{\mathit{final}}_{\mathit{final}}, \mathit{init} \to \mathit{final}\}$$
$$\vdash \{\mathit{init}\}\ X\ \{\mathit{final}\}$$

by which we conclude $\mathit{Tr}_{\mathcal{S}} \vdash \{\mathit{init}\}\ X\ \{\mathit{final}\}$. (*end example*)

5.2. *The Proof System H Is Sound and Complete*

LEMMA 5.2.1. *The proof system H is sound.*

Proof. Let $\mathcal{S}$, $\mathcal{L}$ be fixed and $\mathcal{I}$ an interpretation of $\mathcal{L}$ in $\mathcal{S}$. Assume

$$\mathit{Tr}_{\mathcal{S}}, \Phi \vdash \{\alpha\}\ t\ \{\beta\}$$

$$\dfrac{\mathit{init} \rightarrow \mathit{final} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\texttt{in}^{\mathit{final}}_{\mathit{busy}}}{\{\mathit{final}\}\,\texttt{in}\,\{\mathit{busy}\}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\texttt{pa}^{\mathit{busy}}_{\mathit{busy}}}{\{\mathit{busy}\}\,\texttt{pa}\,\{\mathit{busy}\}} \qquad \dfrac{\texttt{p}_\texttt{b}{}^{\mathit{busy}}_{\mathit{busy}}}{\{\mathit{busy}\}\,\texttt{p}_\texttt{b}\,\{\mathit{busy}\}}}{\{\mathit{busy}\}\,\texttt{pa}+\texttt{p}_\texttt{b}\,\{\mathit{busy}\}} \qquad \dfrac{\texttt{co}^{\mathit{busy}}_{\mathit{final}}}{\{\mathit{busy}\}\,\texttt{co}\,\{\mathit{final}\}}}{\{\mathit{busy}\}\,(\texttt{pa}+\texttt{p}_\texttt{b})\texttt{co}\,\{\mathit{final}\}} \qquad \dfrac{\texttt{pr}^{\mathit{busy}}_{\mathit{final}}}{\{\mathit{busy}\}\,\texttt{pr}\,\{\mathit{final}\}}}{\{\mathit{busy}\}\,(\texttt{pa}+\texttt{p}_\texttt{b})\texttt{co}+\texttt{pr}\,\{\mathit{final}\}}}{\{\mathit{final}\}\,\texttt{in}((\texttt{pa}+\texttt{p}_\texttt{b})\texttt{co}+\texttt{pr})\,\{\mathit{final}\}} \qquad [\{\mathit{final}\}\,x\,\{\mathit{final}\}]}{\{\mathit{final}\}\,\texttt{in}((\texttt{pa}+\texttt{p}_\texttt{b})\texttt{co}+\texttt{pr})x\,\{\mathit{final}\}} \qquad \dfrac{\texttt{st}^{\mathit{final}}_{\mathit{final}}}{\{\mathit{final}\}\,\texttt{st}\,\{\mathit{final}\}}}{\{\mathit{final}\}\,\texttt{in}((\texttt{pa}+\texttt{p}_\texttt{b})\texttt{co}+\texttt{pr})x+\texttt{st}\,\{\mathit{final}\}}}{\{\mathit{final}\}\,X\,\{\mathit{final}\}}}{\{\mathit{init}\}\,X\,\{\mathit{final}\}}$$

FIG. 2. A derivation in H.

for some term t over Σ^+, assertions α, β, and a set Φ of correctness formulae. It suffices to show that for each derivation of $\{\alpha\}\, t\, \{\beta\}$ using hypotheses in $Tr_{\mathscr{S}} \cup \Phi$ it holds that

$$\forall \bar{p}[\mathscr{S} \models_{\mathscr{I}} \Phi(\bar{p}) \quad \Rightarrow \quad \mathscr{S} \models_{\mathscr{I}} \{\alpha\}\, t(\bar{p})\, \{\beta\}].$$

We use induction on the length of derivations, and only consider the cases of a derivation of $\{\alpha\}\, t\, \{\beta\}$ in which Rule IV or Rule VI is applied last (the other cases are straightforward).

1. Assume there is a derivation of $\{\alpha\}\, t\, \{\beta\}$ in which IV was applied last, so $t \equiv t_1 t_2$ and there are $\Phi_1, \Phi_2 \subseteq \Phi$ and γ such that $Tr_{\mathscr{S}}, \Phi_1 \vdash \{\alpha\}\, t_1\, \{\gamma\}$ and $Tr_{\mathscr{S}}, \Phi_2 \vdash \{\gamma\}\, t_2\, \{\beta\}$. Suppose $\bar{p}_0$ is such that $\mathscr{S} \models_{\mathscr{I}} \Phi(\bar{p}_0)$; then $\mathscr{S} \models_{\mathscr{I}} \Phi_i(\bar{p}_0)$ $(i = 1, 2)$ and by induction $\mathscr{S} \models_{\mathscr{I}} \{\alpha\}\, t_1(\bar{p}_0)\, \{\gamma\}$ and $\mathscr{S} \models_{\mathscr{I}} \{\gamma\}\, t_2(\bar{p}_0)\, \{\beta\}$. We have to show $\mathscr{S} \models_{\mathscr{I}} \{\alpha\}\, t_1 t_2(\bar{p}_0)\, \{\beta\}$: let s be such that $\mathscr{S} \models_{\mathscr{I}} \alpha[s]$ and $(t_1 t_2(\bar{p}_0), s) \twoheadrightarrow (\surd, s')$. By *decomposition* (see Lemma 3.2.1) there is s'' such that $(t_1(\bar{p}_0), s) \twoheadrightarrow (\surd, s'')$ and $(t_2(\bar{p}_0), s'') \twoheadrightarrow (\surd, s')$. Using the induction hypothesis we find $\mathscr{S} \models_{\mathscr{I}} \gamma[s'']$ and consequently $\mathscr{S} \models_{\mathscr{I}} \beta[s']$, which was to be proved.

2. Assume $\{\alpha\}\, t\, \{\beta\}$ follows from an application of VI, so $\{\alpha\}\, t\, \{\beta\} \equiv \{\alpha_z\}\, \langle z \mid E\rangle\, \{\beta_z\}$ for some pure system $E = \{x = t_x \mid x \in V_E\}$ and there are assertions α_x, β_x $(x \in V_E)$ and $\Phi' \subseteq \Phi$ such that $Tr_{\mathscr{S}}, \Phi'$, $\{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\} \vdash \{\alpha_y\}\, t_y\, \{\beta_y\}$ for all $y \in V_E$ and Φ' contains no correctness formulae with free variables in V_E. Suppose $\mathscr{S} \models_{\mathscr{I}} \Phi(\bar{p}_0)$ for some $\bar{p}_0$. We have to show $\mathscr{S} \models_{\mathscr{I}} \{\alpha_z\}\, \langle z \mid E\rangle\, \{\beta_z\}$. By the induction hypothesis we have

$$\begin{aligned} &\forall \bar{p}[\mathscr{S} \models_{\mathscr{I}} \Phi' \cup \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_E\}(\bar{p}) \\ &\quad \Rightarrow \quad \mathscr{S} \models_{\mathscr{I}} \{\{a_y\}\, t_y(\bar{p})\, \{\beta_y\} \mid x \in V_E\}]. \end{aligned}$$

Define $E' = \{x = t'_x \mid x \in V_{E'}\}$ as the system obtained from E by removing all brackets in the expressions t_x as allowed by *distributivity* (see Lemma 4.2.5), so $V_E = V_{E'}$. It follows that

$$\forall \bar{p}[\mathcal{S} \models_{\mathcal{I}} \Phi' \cup \{\{\alpha_x\}\, x\, \{\beta_x\} \mid x \in V_{E'}\}(\bar{p}) \\ \Rightarrow \quad \mathcal{S} \models_{\mathcal{I}} \{\{\alpha_x\}\, t'_x(\bar{p})\, \{\beta_x\} \mid x \in V_{E'}\}]. \qquad (1)$$

For a start we prove as an intermediate result

$$\mathcal{S} \models_{\mathcal{I}} \{\{\alpha_x\}\, \pi_n(\langle x \mid E'\rangle)\, \{\beta_x\} \mid x \in V_{E'}\}$$

for all $n \in \mathbb{N}$ by induction on n:

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

$n+1$. Let $x_0 \in V_{E'}$ and suppose $\mathcal{S} \models_{\mathcal{I}} \alpha_{x_0}[s]$ and $(\pi_{n+1}(\langle x_0 \mid E'\rangle), s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')$. Assume $t'_{x_0} \equiv \sum p_i y_i + \sum q_j$, 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:

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

In both cases it follows that $(t'_{x_0}(\overline{\pi_n(\langle x \mid E'\rangle)}), s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')$. Let $\bar{q}_k$ $(k \in \mathbb{N})$ be obtained from $\bar{p}_0$ by replacing any "$V_{E'}$-coordinate" by $\pi_k(\langle x \mid E'\rangle)$. Because Φ' does not contain correctness formulae with free variables in $V_{E'}$, it follows that $\mathcal{S} \models_{\mathcal{I}} \Phi'(\bar{q}_k)$ for all $k \in \mathbb{N}$. By the "local induction hypothesis" we have $\mathcal{S} \models_{\mathcal{I}} \{\{\alpha_x\}\, \pi_n(\langle x \mid E'\rangle)\, \{\beta_x\} \mid x \in V_{E'}\}$ and using $\bar{q}_n$ in (1) we find that $\mathcal{S} \models_{\mathcal{I}} \{\alpha_{x_0}\}\, t'_{x_0}(\overline{\pi_n(\langle x \mid E'\rangle)})\, \{\beta_{x_0}\}$. We conclude $\mathcal{S} \models_{\mathcal{I}} \beta_{x_0}[s']$, which proves the intermediate result.

So in particular we have $\mathcal{S} \models_{\mathcal{I}} \{\alpha_z\}\, \pi_n(\langle z \mid E'\rangle)\, \{\beta_z\}$ for all n. Now suppose $\mathcal{S} \models_{\mathcal{I}} \alpha_z[s]$ and $(\langle z \mid E'\rangle, s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')$. By inspection of the effect rules for the π_n-operators it follows easily that for n sufficiently large $(\pi_n(\langle z \mid E'\rangle), s) \xrightarrow{\sigma}\!\!\!\!\rightarrow (\surd, s')$, so $\mathcal{S} \models_{\mathcal{I}} \beta_z[s']$, which shows that $\mathcal{S} \models_{\mathcal{I}} \{\alpha_z\}\, \langle z \mid E'\rangle\, \{\beta_z\}$. By regarding effect reductions as parts composed by the effect rule *recursion*, it follows that $\mathcal{S} \models_{\mathcal{I}} \langle x \mid E'\rangle =_{\mathrm{pc}} \langle x \mid E\rangle$ for all $x \in V_E$. We conclude $\mathcal{S} \models_{\mathcal{I}} \{\alpha_z\}\, \langle z \mid E\rangle\, \{\beta_z\}$, as was to be 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–V; obviously H_0 is associated with Σ_0.

• H_{i+1} denotes the proof system H with the applicability of Rule VI 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 5.2.2. *The proof system H_0 is complete.*

Proof. Let $\mathscr{S}$ be a fixed structure. We have to prove

$$\mathscr{S} \models \{\alpha\}\, p\, \{\beta\} \quad \Rightarrow \quad Tr_{\mathscr{S}} \vdash^{H_0} \{\alpha\}\, p\, \{\beta\}$$

for all p occurring in partial correctness assertions over $\mathscr{L}_{\mathscr{S}}$. Suppose $\mathscr{S} \models \{\alpha\}\, p\, \{\beta\}$. Recall that $\mathscr{P}_0$, 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.

$p \equiv \delta$. By Axiom I we derive $Tr_{\mathscr{S}} \vdash^{H_0} \{\alpha\}\, \delta\, \{\beta\}$.

$p \equiv a \in A$. We show that $\neg\, stop_a(v) \wedge \alpha(v) \rightarrow \beta(effect_a(v)) \in Tr_{\mathscr{S}}$. Assume $\mathscr{S} \models \neg\, stop_a \wedge \alpha[s]$, then $(a, s) \xrightarrow{a(s)} (\surd, s(a))$ and by supposition we find $\mathscr{S} \models \beta[s(a)]$. Therefore $\mathscr{S} \models \beta(effect_a(v))[s]$. By the Axiom II we derive $Tr_{\mathscr{S}} \vdash^{H_0} \{\alpha\}\, a\, \{\beta\}$.

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

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

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 $\mathscr{S} \models \{\alpha\}\, \langle x \mid E\rangle\, \{\beta\}$ with E a pure system over Σ_i. In the following crucial lemma we show that such a statement implies $Tr_{\mathscr{S}} \vdash^{H_{i+1}} \{\alpha\}\, \langle x \mid E\rangle\, \{\beta\}$.

LEMMA 5.2.3. *Let $\mathscr{S} = \langle S, action, effect\rangle$ be some structure, $E = \{x = t_x \mid x \in V_E\}$ a pure system over Σ_i and $x_0 \in V_E$. If H_i is complete, then*

$$\mathscr{S} \models \{\alpha\}\, \langle x_0 \mid E\rangle\, \{\beta\} \quad \Rightarrow \quad Tr_{\mathscr{S}} \vdash^{H_{i+1}} \{\alpha\}\, \langle x_0 \mid E\rangle\, \{\beta\}.$$

Proof. Let $E' = \{x = t'_x \mid x \in V_{E'}\}$ be the system obtained by removing all brackets in the expressions t_x as allowed by *distributivity* (see

Lemma 4.2.5), so $V_E = V_{E'}$. It follows that $\mathscr{S} \models \{\alpha\}\ \langle x_0 \mid E' \rangle\ \{\beta\}$. By definition of $\mathscr{L}_{\mathscr{S}}$ we can define *weakest preconditions* for all constants $\langle x \mid E' \rangle$ and β. For any $x \in V_{E'}$ let the assertion α_x be defined as follows:

$$\mathscr{S} \models \alpha_x[s] \quad \Leftrightarrow \quad \text{There are } s' \in S, \sigma \in A^* \text{ such that}$$

$$(\langle x_0 \mid E' \rangle, s') \xrightarrow{\sigma}\!\!\!\!\to (\langle x \mid E' \rangle, s) \quad \text{and} \quad \mathscr{S} \models \alpha[s'].$$

Observe that $\mathscr{S} \models \alpha \to \alpha_{x_0}$ and $\mathscr{S} \models \{\alpha_{x_0}\}\ \langle x_0 \mid E' \rangle\ \{\beta\}$. We first prove that for all $y \in V_E$

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

Let $x_1 \in V_{E'}$ be fixed. We distinguish two cases:

1. For any summand pz of t'_{x_1} $(p \in \mathscr{P}_i, z \in V_{E'})$ we prove $Tr_{\mathscr{S}}$, $\{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\alpha_{x_1}\}\ pz\ \{\beta\}$. It is sufficient to show that $\mathscr{S} \models \{\alpha_{x_1}\}\ p\ \{\alpha_z\}$: by the completeness of H_i it follows that $Tr_{\mathscr{S}} \vdash^{H_i} \{\alpha_{x_1}\}\ p\ \{\alpha_z\}$ and thus $Tr_{\mathscr{S}}, \{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\alpha_{x_1}\}\ pz\ \{\beta\}$. Suppose $\mathscr{S} \models \alpha_{x_1}[s]$ for some $s \in S$ and $(p, s) \xrightarrow{v}\!\!\!\!\to (\surd, s'')$. We derive $(\langle x_1 \mid E' \rangle, s) \xrightarrow{v}\!\!\!\!\to (\langle z \mid E' \rangle, s'')$. By construction of α_{x_1} there is an $s' \in S$ such that $\mathscr{S} \models \alpha[s']$ and $(\langle x_0 \mid E' \rangle, s') \xrightarrow{\sigma}\!\!\!\!\to (\langle x_1 \mid E' \rangle, s)$. So $(\langle x_0 \mid E' \rangle, s') \xrightarrow{\sigma v}\!\!\!\!\to (\langle z \mid E' \rangle, s'')$, by which we conclude $\mathscr{S} \models \alpha_z[s'']$.

2. For any summand q of t'_{x_1} $(q \in \mathscr{P}_i)$ we prove $Tr_{\mathscr{S}}$, $\{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\alpha_{x_1}\}\ q\ \{\beta\}$. Again it is sufficient to show that $\mathscr{S} \models \{\alpha_{x_1}\}\ q\ \{\beta\}$, for $Tr_{\mathscr{S}}, \{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\alpha_{x_1}\}\ q\ \{\beta\}$ follows then by the completeness of H_i. Suppose $\mathscr{S} \models \alpha_{x_1}[s]$ for some $s \in S$ and $(q, s) \xrightarrow{v}\!\!\!\!\to (\surd, s'')$. We derive $(\langle x_1 \mid E' \rangle, s) \xrightarrow{v}\!\!\!\!\to (\surd, s'')$. By construction of α_{x_1} there is an $s' \in S$ such that $\mathscr{S} \models \alpha[s']$ and $(\langle x_0 \mid E' \rangle, s') \xrightarrow{\sigma}\!\!\!\!\to (\langle x_1 \mid E' \rangle, s)$. So $(\langle x_0 \mid E' \rangle, s') \xrightarrow{\sigma v}\!\!\!\!\to (\surd, s'')$. Because $\mathscr{S} \models \{\alpha\}\ \langle x_0 \mid E' \rangle\ \{\beta\}$ we have that $\mathscr{S} \models \beta[s'']$. We conclude $\mathscr{S} \models \{\alpha_{x_1}\}\ q\ \{\beta\}$.

So $Tr_{\mathscr{S}}, \{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\alpha_y\}\ t'_y\ \{\beta\}$ for all $y \in V_{E'}$. We now have to show that we may replace the expressions t'_y by t_y in this result. For simplicity consider the following typical case:

$t'_y \equiv pq + pz$ and $t_y \equiv p(q+z)$ and $Tr_{\mathscr{S}}, \{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\alpha_y\}\ pq + pz\ \{\beta\}$ with $p, q \in \mathscr{P}_i$, $y, z \in V_{E'}$. Now there must be assertions γ_1, γ_2 such that

$$Tr_{\mathscr{S}} \vdash^{H_i} \{\alpha_y\}\ p\ \{\gamma_1\}, \qquad Tr_{\mathscr{S}} \vdash^{H_i} \{\gamma_1\}\ q\ \{\beta\}$$

and

$$Tr_{\mathscr{S}} \vdash^{H_i} \{\alpha_y\}\ p\ \{\gamma_2\},$$

$$Tr_{\mathscr{S}}, \{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\gamma_2\}\ z\ \{\beta\}.$$

Using the soundness of H we find $\mathscr{S} \models \{\alpha_y\}\ p\ \{\gamma_1 \wedge \gamma_2\}$, and by the completeness of H_i it follows that $Tr_{\mathscr{S}} \vdash^{H_i} \{\alpha_y\}\ p\ \{\gamma_1 \wedge \gamma_2\}$. Because $\gamma_1 \wedge \gamma_2 \rightarrow \gamma_1$, $\gamma_1 \wedge \gamma_2 \rightarrow \gamma_2 \in Tr_{\mathscr{S}}$, we can use Rule V to derive $Tr_{\mathscr{S}}, \{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{\alpha_y\}\ p(q+z)\ \{\beta\}$.

Generalizing this argument one may conclude $Tr_{\mathscr{S}}, \{\{\alpha_x\}\ x\ \{\beta\} \mid x \in V_{E'}\} \vdash^{H_i} \{a_y\}\ t_y\ \{\beta\}$ for all $y \in V_{E'} = V_E$. Using Rule VI we derive $Tr_{\mathscr{S}} \vdash^{H_{i+1}} \{\alpha_{x_0}\}\ \langle x_0 \mid E\rangle\ \{\beta\}$. Because $\alpha \rightarrow \alpha_{x_0} \in Tr_{\mathscr{S}}$ we derive by Rule V $Tr_{\mathscr{S}} \vdash^{H_{i+1}} \{\alpha\}\ \langle x_0 \mid E\rangle\ \{\beta\}$, which completes our proof. Note that if α represents the empty predicate the lemma still holds. ▮

THEOREM 5.2.4. *If the proof system H_i is complete, then the proof system H_{i+1} is complete.*

Proof. In the proof of Lemma 5.2.2 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 $\mathscr{P}_{i+1}$ is also specified inductively, we only have to check one more basic clause than in the proof of Lemma 5.2.2, namely $p \equiv \langle x \mid E\rangle$ with $E = \{x = t_x \mid x \in V_E\}$ a pure system over Σ_i. This has just been done in Lemma 5.2.3. ▮

COROLLARY 5.2.5. *The proof system H is complete.*

5.3. *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., Baeten and Bergstra, 1988a, b). 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* iff each variable in the expressions t_x occurs in a subterm $a \cdot M$ with $a \in A$. Let Σ_s denote the restriction of Σ obtained by considering only strictly pure systems, and $\mathscr{P}_s$ the corresponding set of closed process expressions. We define H_s by restricting the applicability of Rule VI 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

$$Tr_{\mathscr{S}} \vdash^{H} \{\alpha\}\ p\ \{\beta\} \quad \Rightarrow \quad Tr_{\mathscr{S}} \vdash^{H_s} \{\alpha\}\ p\ \{\beta\}$$

for all $p \in \mathscr{P}_s$, 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 VI applicable to all guarded systems is not complete any more (and neither is H_s). Assume that

H is still sound with respect to this extension (this is proved in Ponse and de Vries, 1989). We show by an example that H cannot be complete:

EXAMPLE. Let $\mathscr{S} = \langle \{s, s'\}, \mathit{action}, \mathit{effect} \rangle$ be a structure with *action* inert and *effect* satisfying

$$s(a) = s' \qquad \text{and} \qquad s'(a) = s.$$

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

$$Tr_{\mathscr{S}}, \{\alpha\}\ x\ \{\beta\} \vdash \{\alpha\}\ axa + aa\ \{\beta\} \tag{2}$$

$$\sigma \to \alpha,\ \beta \to \sigma \in Tr_{\mathscr{S}}. \tag{3}$$

Now (2) implies that $Tr_{\mathscr{S}} \vdash \{\alpha\}\ aa\ \{\beta\}$ and by (3) we derive $Tr_{\mathscr{S}} \vdash \{\sigma\}\ aa\ \{\beta\}$. Because H is sound and $(aa, s) \xrightarrow{a} (a, s') \xrightarrow{a} (\surd, s)$, we conclude $\mathscr{S} \models \beta[s]$. Using (3) we find

$$\beta \leftrightarrow \sigma \in Tr_{\mathscr{S}}. \tag{4}$$

Also $Tr_{\mathscr{S}}$, $\{\alpha\}\ x\ \{\beta\} \vdash \{\alpha\}\ axa\ \{\beta\}$, so there must be γ_1, γ_2 such that $Tr_{\mathscr{S}} \vdash \{\alpha\}\ a\ \{\gamma_1\}$,

$$Tr_{\mathscr{S}} \vdash \{\gamma_2\}\ a\ \{\beta\} \tag{5}$$

and $Tr_{\mathscr{S}}$, $\{\alpha\}\ x\ \{\beta\} \vdash \{\gamma_1\}\ x\ \{\gamma_2\}$. From the derivability of $\{\gamma_1\}\ x\ \{\gamma_2\}$ we conclude

$$\beta \to \gamma_2 \in Tr_{\mathscr{S}}. \tag{6}$$

From (5), (6), and (4) we derive $Tr_{\mathscr{S}} \vdash \{\sigma\}\ a\ \{\sigma\}$, so by the soundness of H we have $\mathscr{S} \models \{\sigma\}\ a\ \{\sigma\}$. This is a contradiction, for $(a, s) \xrightarrow{a} (\surd, s')$, but not $\mathscr{S} \models \sigma[s']$. 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.)

In terms of the semantics of partial correctness assertions there is a formalizable correspondence between processes defined with the use of a *finite* pure (guarded) system and regular (context-free, respectively) languages over A_δ (see, e.g., Hopcroft and Ullman, 1979, for an introduction to such languages). So the example above shows that Scott's induction rule is not compatible with "context-free recursion" in our set-up.

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 to use a number of algebraic laws that define the equality relation over 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* that 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*

It can be proved that the constant τ, representing "silent" or "unobservable" action, can be added to Σ without invalidating our completeness result. The semantical rules

$$\tau\text{-laws: } (a, s) \xrightarrow{a(s)} (\tau, s(a)) \qquad \text{if} \quad a(s) \neq \delta$$

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

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

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 Milner, 1980). 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.

Acknowledgments

I 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, Fer-Jan de Vries (who suggested the format of the recursion rule of H, presented here), and two anonymous referees.

Received September 29, 1989; final manuscript received May 17, 1990

REFERENCES

APT, K. R. (1981), Ten Years of Hoare's Logic: A Survey—Part I, *ACM Trans. Prog. Languages Systems* **3**(4), 431–483.

APT, K. R. (1984), Ten Years of Hoare's Logic: A Survey—Part II: Nondeterminism, *Theoret. Comput. Sci.* **28**, 83–109.

BAETEN, J. C. M., AND BERGSTRA, J. A. (1988a), Global Renaming Operators over Concrete Process Algebra, *Inform. Comput.* **78**(3), 205–245.

BAETEN, J. C. M., AND BERGSTRA, J. A. (1988b), Recursive Process Definitions with the State Operator, *in* "Proceedings Computing Science in the Netherlands (SION)," pp. 279–294.

BAETEN, J. C. M., AND WEIJLAND, W. P. (1990), "Process Algebra," Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press.

DE BAKKER, J. W. (1980), "Mathematical Theory of Program Correctness," Prentice–Hall, London.

DE BAKKER, J. W., KOK, J. N., MEYER, J.-J. CH., OLDEROG, E.-R., AND ZUCKER, J. I. (1986), Contrasting themes in the semantics of imperative concurrency, *in* "Current Trends in Concurrency" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), pp. 51–121, Lecture Notes in Computer Science, Vol. 224, Springer-Verlag, Berlin/New York.

BERGSTRA, J. A., AND KLOP, J. W. (1986), Process Algebra: Specification and Verification in Bisimulation Semantics, *in* "Mathematics and Computer Science II" (M. Hazewinkel, J. K. Lenstra, and L. G. L. T. Meertens, Eds.), pp. 61–94, CWI Monographs, Vol. 4, North-Holland, Amsterdam.

VAN DALEN, D. (1983), "Logic and Structure," Springer-Verlag, Berlin/New York.

VAN GLABBEEK, R. J. (1987), Bounded Nondeterminism and the Approximation Induction Principle in Process Algebra, *in* "Proceedings STACS 87" (F. J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, Eds.), pp. 336–347, Lecture Notes in Computer Science, Vol. 247, Springer-Verlag, Berlin/New York.

HOPCROFT, J. E., AND ULLMAN, J. D. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, MA.

MILNER, A. J. R. G. (1980), "A Calculus of Communicating Systems," Lecture Notes in Computer Science, Vol. 92, Springer-Verlag, Berlin/New York.

PARK, D. M. R. (1981), Concurrency and automata on infinite sequences, *in* "Proceedings 5th GI Conference" (P. Deussen, Ed.), pp. 167–183, Lecture Notes in Computer Science, Vol. 104, Springer-Verlag, Berlin/New York.

PLOTKIN, G. D. (1983), An Operational Semantics for CSP, *in* "Formal Description of Programming Concepts—II" (D. Bjørner, Ed.), pp. 199–223, North-Holland, Amsterdam.

PONSE, A., AND DE VRIES, F.-J. (1989), "Strong Completeness for Hoare Logics of Recursive Processes—An Infinitary Approach," Report CS-R8957, Centrum voor Wiskunde en Informatica, Amsterdam.