Centrum voor Wiskunde en Informatica

Centre for Mathematics and Computer Science

A. Ponse, F.-J. de Vries

Strong completeness for Hoare logics of recursive processes: an infinitary approach

Computer Science/Department of Software Technology Report CS-R8957 December

1989

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

A. Ponse, F.-J. de Vries

Strong completeness for Hoare logics of recursive processes:
an infinitary approach

Computer Science/Department of Software Technology Report CS-R8957 December

---

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

Strong completeness for Hoare logics of recursive processes: an infinitary approach

Alban Ponse (alban@cwi.nl)

Fer-Jan de Vries (ferjan@cwi.nl)

Centre for Mathematics and Computer Science
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

Abstract: We abstract from datatypes and set up Hoare logics for partial correctness for process languages based on process algebra. These process languages cover features of programming languages unrelated to program variables and data types: sequential composition, non-deterministic choice, tests, conditionals, iterations and recursion. As partial correctness formulas can be expressed in infinitary logics we define an $\mathcal{L}_{\omega_1\omega}$-assertion language for which completeness (= soundness + adequacy) is well-known. For recursion we have two different recursion rules (Scott's induction rule and the infinitary induction rule) and hence two different Hoare logics. Both logics are complete for different classes of recursive processes (respectively processes which define a regular or a context-free language). The strong completeness theorems for the Hoare logics follow from the completeness theorem for assertion logic and the theorem stating that assertion logic is a conservative extension of both Hoare logics. The strong completeness theorems imply the usual relative completeness (in the sense of Cook) theorems. As a corollary we obtain relative completeness theorems for the Hoare logic of imperative (non-)deterministic programming languages. Another corollary is that the semantics induced by the Hoare logics can now be axiomatized in the style of Process Algebra.

Keywords and phrases: process, programming languages, Hoare logic, recursion, infinitary logic, partial correctness, semantics, completeness.
1985 Mathematics subject classification: 68Q60, 68Q55, 68N15, 68Q10, 03C70
1987 CR categories: D.3.1., F.3.1, F.3.3, F.4.1, F.1.2

Contents:

Report CS-R8957
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. INTRODUCTION.

1.1. SHORT OVERVIEW.

Abstracting from data in programs we prove two strong completeness theorems for the Hoare logic of non-deterministic processes with recursion. The two theorems correspond with two options for a Hoare rule concerning recursion: the infinitary induction rule and Scott's induction rule. For the infinitary induction rule we prove completeness with respect to the general class of context-free processes, for Scott's induction rule we prove completeness for the restricted class of regular processes. Scott's induction rule is not complete for the full class of processes as we can show by a counter example (namely the process defined by the recursive equation x=axa+aa). The main tools of the proofs are an infinitary assertion logic in which the partial correctness formulas are expressible together with its semantics. The infinitary assertion logic can be seen as a conservative extension of two Hoare logics in the following way:

- If Γ is a set of partial correctness formulas and/or implications, then
  $\Gamma\vdash_{\text{HoareLogic}}\{\alpha\}p\{\beta\} \Rightarrow \Gamma\vdash_{\text{AssertionLogic}}\{\alpha\}p\{\beta\}$.
- If Γ is a countable set of formulas in the assertion language, then
  $\Gamma\vdash_{\text{AssertionLogic}}\{\alpha\}p\{\beta\} \Rightarrow \text{Th}(\Gamma)\vdash_{\text{HoareLogic}}\{\alpha\}p\{\beta\}$, where $\text{Th}(\Gamma):=\{\alpha\rightarrow\beta \mid \Gamma\vdash\forall s(\alpha(s)\rightarrow\beta(s))\}$.

The strong completeness theorems follow immediately from these theorems and Karp's well-known completeness theorem for the infinitary logic $\mathcal{L}_{\omega_1\omega}$. Weak and relative completeness are easy corollaries of the strong completeness theorems. For programming languages that are instances of the introduced process language we obtain several traditional completeness theorems in the sense of Cook. As an example we consider the languages treated in the recent monograph of Tucker and Zucker. The semantics of the infinitary assertion logic induces in a natural way a semantics for the process language we abstract the programming language to. We give a process algebra axiomatization of this semantics in the style of Bergstra and Klop. The semantics coincides with the semantics that the Hoare logic induces on processes. We finish this paper relating our results to the literature on Hoare logic.

In the remainder of this section we introduce some of the above notions in more detail.

1.2. HOARE LOGIC FOR RECURSIVE PROCESSES.

Hoare logics are axiomatic methods for proving programs correct. The subject was introduced by Hoare in 1969 and its tenth anniversary was celebrated by two survey papers by Apt, to which we refer for historic background (cf. [A1] and [A2]). Hoare logic of partial correctness is concerned with partial correctness formulas of the form $\{\alpha\}p\{\beta\}$: if the assertion α describes the situation before execution of p, then any successful terminating execution of p results in a situation satisfying β.
In Hoare logic two languages play a role: the language of the programs and the language of the assertions on these programs. In this paper we abstract from variables and datatypes, and concentrate on the Hoare logic of processes. Then interpreting programs as processes built from atomic assignments we can apply the results to programs again.

First we consider Hoare logics for recursion free processes covering features of programming languages not related to data types: sequential composition, non-deterministic choice and tests (hence including conditionals). Using ingredients of the process language we define an $\mathcal{L}_{\omega_1\omega}$–assertion language for which a strong completeness theorem exists by the standard theory of infinitary logic (the idea to use such languages is not new in the field of Hoare logic and goes back to Engeler (cf. [E] and section 8); Back recognized their expressibility for weakest preconditions in (cf. [Bac1&2] and

section 8)). We prove that the assertion logic ($\vdash$) is a conservative extension of Hoare logic ($\vdash_H$) from which the soundness and adequacy theorems for the Hoare logic of recursion free processes follow by an appeal to the completeness theorem of the infinitary assertion logic.
Secondly, we introduce recursion. Now there are two different rules to extend Hoare logic with recursion (and hence with while statements): the infinitary induction rule and Scott's induction rule, respectively (for one variable):

$$\frac{\{\alpha\}t^n(\delta)\{\beta\} \quad \text{for all } n\in \mathbf{N}}{\{\alpha\}X^{x=t(x)}\{\beta\}} \qquad \text{and} \qquad \frac{\begin{array}{c}[\{\alpha\}x\{\beta\}]\\ \vdots \\ \{\alpha\}t(x)\{\beta\}\end{array}}{\{\alpha\}X^{x=t(x)}\{\beta\}}\ .$$

For both systems we have unrestricted soundness theorems. Completeness of the Hoare logic with the infinitary induction rule will be shown for arbitrary processes. In case of completeness of the Hoare logic with Scott's induction rule we have to restrict to regular (or linear) processes, i.e., processes in which linearly specified recursive equations may occur.

1.3. COMPLETENESS THEOREMS FOR HOARE LOGIC.

In a completeness theorem one tries to match in a sound and adequate way a logic of a particular type with a class of models (possible consisting of only one model). If the fit is not proper then one could reconsider the logic or the class of models. Hoare logics with finitary assertion languages mismatch easily. In [BT2] Bergstra and Tucker give some strong examples of incompleteness: they describe models A (Pressburger arithmetic, the field of algebraic numbers and the field of algebraic numbers) and sound Hoare logics over A such that the set of provable correctness formulas is r.e. but not recursive, and the set of correctness formulas true in A is co-r.e. but not recursive. These models are all computable algebraic structures with decidable first order theory $Th(A) := \{\phi \mid A\models\phi\}$.

And only under a constraint concerning expressivity there is:

COOK'S RELATIVE COMPLETENESS THEOREM (cf. [C] or [A1]):
Given some model K, *finitary* assertion logic L and, if L is expressive for K (meaning that semantically defined weakest preconditions (cf. section 4) over K are definable in L) then

$$K\models\{\alpha\}p\{\beta\} \Leftrightarrow \{\phi \mid K\models\phi\}\vdash_H\{\alpha\}p\{\beta\}.$$

If we consider infinitary assertion languages of the type $\mathcal{L}^{\omega}_{\omega_1}$ the mismatch disappears. The incompleteness arguments are circumvented because the infinitary assertion logic is an conservative extension of the Hoare logic considered in this paper. We can now prove strong completeness theorems of the form:

STRONG COMPLETENESS THEOREM:

(i) (soundness) If Γ is a set of partial correctness formulas and/or implications, then
$\Gamma\vdash_H\{\alpha\}p\{\beta\} \Rightarrow \Gamma\models\{\alpha\}p\{\beta\}$.

(ii) (adequacy) If Γ is a countable set of formulas in the (infinitary) assertion language, then
$\Gamma\models\{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma)\vdash_H\{\alpha\}p\{\beta\}$,
where $Th(\Gamma):=\{\alpha\rightarrow\beta \mid \Gamma\vdash\forall s(\alpha(s)\rightarrow\beta(s))\}$.

In particular:

COROLLARY: for one's favourite standard model K it holds that

$$K\models\{\alpha\}p\{\beta\} \Leftrightarrow Th(K)\vdash_H\{\alpha\}p\{\beta\},$$

where

$$Th(K):=\{\{\alpha\}p\{\beta\} \mid K\models\{\alpha\}p\{\beta\}, \text{ for all } \alpha, \beta \text{ and closed process expressions } p\}$$

(PROOF. $Th(K)\models\{\alpha\}p\{\beta\} \Leftrightarrow K\models\{\alpha\}p\{\beta\}$ and Th(K) is closed under provability by soundness. □)

An infinitary assertion language is expressive by nature in contrast to arbitrary finitary assertion languages. Cook's condition on expressibility is just the question whether there exist finitary equivalents of weakest preconditions and is irrelevant for the question of completeness! One can imagine that finitary equivalents of weakest preconditions exist if the finitary part of the infinitary assertion logic has sufficient coding properties, corresponding to the coding properties of the natural numbers. See Harel's analysis in [Ha] an Bergstra and Tucker in [BT2].

Our strong completeness theorems and the corollary differ from the traditional relative completeness results for Hoare logic:

- *strong* completeness, that is, we take sets with hypotheses into account.
- Th(Γ) is syntactically defined, i.e. Th(Γ) is model/interpretation independent.
- depending on the chosen recursion rule there are different completeness theorems with respect to different classes of processes.
- we don't need all usual rules for completeness of the Hoare logics with recursion rules (in terms of [dB] we don't need the conjunction and invariance rule nor the substitution rules).

1.4. HOW ADEQUACY OF HOARE LOGIC DEPENDS ON ADEQUACY OF ASSERTION LOGIC.

We now give a heuristic argument how the *adequacy of Hoare logic* depends on the *adequacy of the assertion logic*. The consequence rule of Hoare logic is not exclusively concerned with correctness formulas:

$$\frac{\alpha(s)\rightarrow\alpha'(s) \qquad \{\alpha'\}p\{\beta'\} \qquad \beta'(s)\rightarrow\beta(s)}{\{\alpha\}p\{\beta\}} \text{ consequence rule}$$

This rule has expressions of the form $\alpha(s)\rightarrow\alpha'(s)$ among its hypotheses. Following the tradition of logic one would like to prove *adequacy*: $\Gamma\models\{\alpha\}p\{\beta\} \Rightarrow \Gamma\vdash_H\{\alpha\}p\{\beta\}$, that is: if any of our models in which all formulas of a set Γ of correctness formulas and/or universal closures of implications are valid also satisfies $\{\alpha\}p\{\beta\}$, then we have a proof in Hoare logic of $\{\alpha\}p\{\beta\}$ based on the set of hypotheses Γ.

This formulation of adequacy is not adequate. Suppose one can prove $\{\alpha\}p\{\beta\}$, and suppose $\beta\rightarrow\gamma$ holds in all models in which $\{\alpha\}p\{\beta\}$ is valid. Then $\{\alpha\}p\{\gamma\}$ holds there as well. Completeness would imply that we can prove $\{\alpha\}p\{\gamma\}$ from Γ in Hoare logic. In general this is only the case if in the assertion logic we can prove $\beta\rightarrow\gamma$ necessary for the Hoare logic proof of $\{\alpha\}p\{\gamma\}$. One way of proving $\beta\rightarrow\gamma$ would be via an adequacy theorem for assertion logic. But then the notion of adequacy for Hoare logic is:

$$\Gamma\models\{\alpha\}p\{\beta\}\Rightarrow Th(\Gamma)\vdash_H\{\alpha\}p\{\beta\},$$

where Th(Γ) is the set of assertion formulas valid in all models satisfying the set of hypotheses Γ. Using the completeness theorem for assertion logic we can equivalently define Th(Γ) syntactically: let Th(Γ) be the set of formulas ϕ such that $\Gamma\vdash_{AssLog}\phi$.

2. THE PROCESS LANGUAGE.

Processes are programs in which we have abstracted from data and variables. One should compare this in analogy to propositions and predicates. Different theories for processes have been developed. In this section we will encounter a variant of Process Algebra of Bergstra and Klop (cf. [BK3&4]).

In order to incorporate conditionals and iterations (if-then-else and while-loops) from programming languages we consider a process language in which tests occur as processes. In Process Algebra one already has felt a need for tests: tests can be found in disguise as state operators (cf. [BB1&2]). Of course the notion of tests in (even non-deterministic) programming languages is well known (cf. e.g. [dB]).

Programs are often thought of as state transformers, and states are usually interpreted as valuation functions assigning a value to variables of the programming language or leaving them undefined. Hence, in this view states are structured objects. Processes abstract from the structure of states. The structure of the states is not observable, only the actions performed on them. The abstraction from programs to processes preserves the idea of state transformers.

Following this line of thought it is natural to introduce a state semantics for processes, as in [Po]. The intuitive picture is a system M being in some state s. The execution of an *atomic action* on M transforms the state s into a state s'. Failure (notation $\perp$) can be interpreted as a special state of the system, it can be understood as a state which reports a failure in the execution of the process (cf. [A2]). When a system enters failure it remains in failure: failure is a kind of deadlock. The failure state is a fixed point for all processes. The semantics of a *test* ϕ is simple: we interpret ϕ as a predicate on M. The effect of ϕ on a state s is the state s itself if $\phi(s)$ holds in M and $\perp$ otherwise. That is, depending on the state a test skips or deadlocks to failure.
We can now define equality on processes p and q: if for all models and all interpretations of the process language in the models, whenever process p is able to transform a state s into s', the process q can do the same, and vice versa. We show that a standard set of axioms from Process Algebra together with Trace axioms and Test axioms is a complete axiomatization of this equality relation.
The axiomatization of tests is of independent interest, as it does not use any kind of reference model where the tests have to be performed. It should not come as a surprise that the set of processes which are constructed from tests only is a boolean algebra.

2.1. SYNTAX OF PROCESS ALGEBRA WITH TESTS AND RECURSION.

We will now give syntax definitions of three languages of processes, that we will use in this paper. These languages are constructed from three ingredients:

- a set A of atomic actions A (its typical elements are usually denoted by a,b,...),
- a set Ψ of atomic tests or atomic propositions (typical element ϕ),
- a countable set Var of process variables (typical element x).

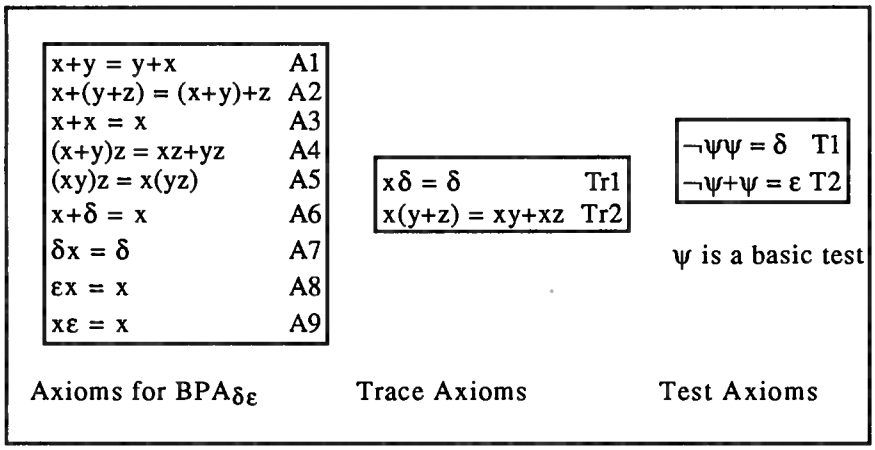
Basic processes are constructed from these ingredients by sequential composition and alternative composition (or non-deterministic choice).

2.1.1. DEFINITION. (syntax of basic tests and processes)

(i) The class $\Psi^{\#}$ of basic tests (typical element ψ) is given by
$\psi ::= \phi \mid \neg\psi \mid \delta \mid \varepsilon$

(ii) The class of basic processes BPT(A,Ψ) (typical element q) is given by
$q ::= a \mid \psi \mid q_1q_2 \mid (q_1+q_2)$

We have introduced ε and δ as tests: ε will be the test that always succeeds, whereupon the process continues, and δ will be the test that always fails, whereupon the process deadlocks. The axioms for BPT(A,Ψ) are the following



Axioms for BPT(A,Ψ)

The axioms A1-A9 are standard for basic process algebra (cf. [BK3&4]) with empty process (ε) and deadlock constant (δ). The trace axioms Tr1 and Tr2 do not belong to standard axiomatizations of process algebra. We need them for the complete axiomatization for the state semantics, in which we will interpret Hoare logic. The axioms T1-T2 for atomic tests are new for process algebra. The axioms for tests may seem not complete, the whole axiom system of process algebra is strong enough to prove that BPT($\varnothing$,Ψ) is a boolean algebra. We will sketch a proof of this fact at the end of this section 2.1. In a different context Manes and Arbib (cf. [MA]) construct a Boolean algebra of guards in a partial additive semantics using similar axioms.

2.1.2. DEFINITION. (syntax of processes with recursive specifications)

(i) The class of open process terms (typical element g)
$g ::= a \mid \psi \mid x \mid g_1g_2 \mid (g_1+g_2)$

(ii) A recursively specified process is a syntactic construct of the form X_1^E,
where $E = \{x_1=g_1,...,x_n=g_n\}$ is a set of n equations and the g_i are process terms with variables from $\{x_1,...,x_n\}$.
In the context of programs the equations are often interpreted as declarations and written as $\{x_1\Leftarrow g_1,...,x_n\Leftarrow g_n\}$.
The variable ranging over the (minimal, as will follow from the axioms) solution(s) of x_i in E will be denoted by X_i^E,

(iii) The class $BPT_{rec}(A,\Psi)$ of processes with recursive specifications (typical element p) is given by
$p ::= a \mid \psi \mid X_i^E \mid p_1+p_2 \mid p_1p_2 \mid \pi_n(p)$, where E is a recursive specification and $n\geq 0$.
The π_n are projections restricting the number of steps a process can make (cf. the axioms at the next page).

2.1.3. DEFINITION. (syntax of processes with linear recursive specifications)

(i) The class of linear process terms is defined by the grammar
$k ::= q \mid qx \mid qk \mid k_1+k_2$

(ii) A linear recursively specified process is a syntactic construct of the form $X_{i|}^{E}$, where
$E_{lin} = \{x_1=k_1,...,x_n=k_n\}$, and the k_i are linear process terms with variables from $\{x_1,...,x_n\}$

(iii) The class $BPT_{linrec}(A,\Psi)$ of processes with linear recursive specifications (typical element p) is given by
$p ::= a \mid \psi \mid X_i^E \mid p_1+p_2 \mid p_1p_2 \mid \pi_n(p)$, where E is a linear recursive specification and $n\geq 0$.

Remark: the definitions exclude nested (linear) recursion. (If one insists on nested recursion one should iterate ad infinitum, i.e., ad ω, this construction, enriching at each iteration step the set of atomic actions with the constructs X_i^E constructed at the previous step (cf. [Po]).)

Comparing these two definitions with the standard ones for process algebra (cf. [BK3&4]), one should note that we don't insist on the guardedness of the recursive specifications. Recall that a recursive specification is guarded if the occurrence of a recursion variable cannot be accessed without passing an atomic action (cf. [BK2]). We have added atomic tests. And there is an inessential change in notation (X_i^E instead of $\langle x_i|E\rangle$).
We have a pragmatic reason for not considering guards: eventually we want to be able to extend the process language again to a programming language by treating assignments x:=t as atomic actions. However, it is difficult to distinguish an assignment like x:=x from ε. In general it even does not need to be decidable that x:=t acts like ε. The notion of guardedness is used in standard process algebra to ensure that in the presence of a finite approximation principle (AIP) guarded equations like x=ax+b do have a unique solution in contrast to unguarded equations like x=x+b. We will solve this uniqueness problem differently: the axioms we shall formulate for recursion will guarantee the existence and uniqueness of solutions (e.g. the solution of x=x+b will be δ+b (=b)) (the solution can be recognized as minimal in terms of the usual metric on the terms of process algebra (cf. [BK2] for more details on the distance underlying the metric). An instance of the axiom in case of the recursive specification of a single variable is $\pi_n(X^{x=t(x)}) = \pi_n(t^n(\delta))$ for some $n\geq 0$.

Processes in $BPT_{rec}(A,\Psi)$ are subject to the axioms of $BPT(A,\Psi)$ together with the following axioms:

$\pi_0(\psi) = \psi$ for $\psi\in\Psi^{\#}$	PR1
$\pi_0(ax) = \delta$ for $a\in A$	PR2
$\pi_n(\psi x) = \psi\pi_n(x)$ for $\psi\in\Psi^{\#}$	PR3
$\pi_{n+1}(ax) = a\pi_n(x)$ for $a\in A$	PR4
$\pi_n(x+y) = \pi_n(x)+\pi_n(y)$	PR5
$\pi_n(X_i^E) = \pi_n(t_i^n)$	PR6
where for $1\leq i\leq k$	
$t_i^0 := \delta$	
$t_i^{n+1} := t_i(t_1^n,...,t_k^n)$	

$X_i^E = t_i(X_1^E,\ldots,X_k^E)$	RDP
$\pi_n(x) = \pi_n(y)$ for all $n\geq 0 \Rightarrow x=y$	AIP

The axioms PR1-PR5, RDP and AIP are a standard extension to basic process algebra with recursion (cf. [Gl], where the projection interacts in a similar way with τ as here with tests). PR6 seems to be new in the process algebra tradition. Of course in case of guarded equations PR6 is derivable in basic process algebra with recursion.
Examples:

$\pi_n(X^{x=x}) := \pi_n(\delta) = \delta$ for all $n\geq 0$, hence $X^{x=x} = \delta$ by AIP.
$\pi_n(X^{x=xa}) := \pi_n(\delta a^n) = \delta$ for all $n\geq 0$, hence $X^{x=xa} = \delta$ by AIP.
$\pi_2(X^{x=xa+b}) := \pi_2((\delta a+b)a+b) = \pi_2(ba+b) = ba+b$
$\pi_2(X^{x=ax+b}) := \pi_2(a(a\delta+b)+b) = \pi_2(ab+b) = ab+b$
$\pi_n(X^{x=ax}) := \pi_n(a^n\delta) = a^n\pi_0(\delta) = \delta$ for all $n\geq 0$, hence $X^{x=ax} = \delta$ by AIP.

In the remainder of this section (2.1) on syntax we will sketch how to prove that BPT(Ψ) (more precisely: equivalence classes of all processes constructed from Ψ (no atomic actions), where we take provable equality as equivalence relation) is a *Boolean algebra* (cf. for instance [Ja]). This fact plays no role in other parts of the article, and can be skipped. In the next section 2.2 we will consider a state semantics for the above process algebra. The existence of such a semantics shows the consistency of the axioms..

2.1.4. EASY FACT.
Since the following statements are equivalent

(i) there exist a proces p' such that $p+p' = q$,
(ii) $p+q = q$.

we denote either of them by $p\leq q$.

2.1.5. LEMMA. The foregoing axioms for BPT(A,Ψ) imply:

(i) Let ψ be a basic test, and ϕ some arbitrary compound test.
If $\phi\leq\psi$ then $\psi\phi = \phi$ and $\neg\psi\phi = \delta$.
(ii) $\psi(\neg\psi) = \delta$ for basic tests ψ
(iii) Let ψ and ϕ as in (i).
If $\phi\leq\psi$ then also $\delta = \phi(\neg\psi)$.

PROOF.

(i) Let ψ be an basic test, and ϕ some arbitrary compound test. Assume $\phi\leq\psi$, i.e. $\phi+\psi = \psi$. Then $\delta = \neg\psi\psi = \neg\psi(\phi+\psi) = \neg\psi\phi+\neg\psi\psi = \neg\psi\phi+\delta = \neg\psi\phi$. It follows that $\phi = \varepsilon\phi = (\psi+\neg\psi)\phi = \psi\phi+\delta = \psi\phi$.

(ii) Since $\neg\psi = \varepsilon\neg\psi = \psi\neg\psi+\neg\psi(\neg\psi)$ we have $\psi\neg\psi\leq\neg\psi$ by (2.1.4). Hence by (i) $\psi\neg\psi = \neg\psi(\psi\neg\psi) = (\neg\psi\psi)\neg\psi = \delta\neg\psi = \delta$.

(iii) Make assumptions as for (i). Then using (ii) $\delta = \psi(\neg\psi) = (\phi+\psi)(\neg\psi) = \phi(\neg\psi)+\psi(\neg\psi) = \phi(\neg\psi)+\delta = \phi(\neg\psi)$.

□

Remark: a similar statement is proved in a rather different context and a different way by Manes and Arbib (cf. (3.19) in [MA]). There they make use of an interesting principle $x+y = \delta \Rightarrow x = \delta$. In process algebra one can prove this principle in one line: assume $x+y = \delta$ then $x = x+\delta = x+x+y = x+y = \delta$.

Negation of *basic* tests can be extended to negation of arbitrary compound tests:

$-\psi := \neg\psi$
$-(x+y) := (-x)(-y)$
$-(xy) := (-x)+(-y)$

One can now check that the axioms for basic tests hold also for arbitrary compound tests, i.e., terms involving no atomic actions.

2.1.6. LEMMA. For x,y in BPT(Ψ) it holds:

(i) $xy = yx$
(ii) $-xx = \delta$
(iii) $-x+x = \varepsilon$

PROOF: (i) is proven by induction to both x and y. (ii) and (iii) are proven simultaneously by induction on the structure of x. □

Now the following lemma can be proved by simple algebraic manipulations.

2.1.7. LEMMA. For elements of BPT(Ψ) the following holds:

(i) $\varepsilon = -\delta$
(ii) $\delta = -\varepsilon$
(iii) $xx = x$
(iv) $--x = x$
(v) $xy+x = x$
(vi) $(x+y)x = x$
(vii) $x(y+z) = xy+xz$
(viii) $xy+z = (x+z)(y+z)$

PROOF. For example:

(iv) $--y = (y+-y)(--y) = y(--y)+\delta = (--y)y = (--y)y+\delta = (--y)y+(-y)y = (--y+-y)y = y$
(v) $x = \varepsilon x = (-y+y)x = ((-y+y)+y)x = (\varepsilon+y)x = x+yx = x+xy$

□

Collecting all results we conclude that BPT(∅,Ψ) is a *Boolean algebra*, i.e., a model for classical propositional logic (for a definition see [Ja] or [MA]).
The fact that mathematical structures have subsets that form boolean algebras is well known, as Manes and Arbib remark. The case of process algebra forms yet another example. It differs from the examples mentioned in [MA]: in process algebra we don't have full distribution: $x+(yz) = (x+y)(x+z)$ does not hold.

It is also possible, but outside the range of this paper, to construct a subset representing a Heyting algebra. Then instead of a unary operation $\neg$ one uses a binary operation $\rightarrow$ on atomic tests satisfying:

$\phi\rightarrow\phi = \varepsilon$
$\phi\psi = \psi\phi$
$\phi(\phi\rightarrow\psi) = \phi\psi$
$\psi(\phi\rightarrow\psi) = \psi$
$\phi\rightarrow(\psi_1\psi_2) = (\phi\rightarrow\psi_1)(\phi\rightarrow\psi_2)$

For more information on Heyting algebras, see for instance [Jo].

2.2. State Semantics for Processes including Tests

In the context of Hoare logic it is natural to think of processes as state transformers. That is, we will interpret a process p as a function from a state space M into its powerset $P(M)$. The execution of an atomic action or test results in the transformation of a state. This will be modeled by effect functions. In this section we will give a precise definition of the state semantics for processes and prove a completeness result with respect to the axiomatization of the previous section.

2.2.1. DEFINITION. A model $<M,effect,test,\perp>$ (shortly M) for processes as state transformers consists of three ingredients:

- a non-empty set M containing a fixed base point $\perp$ (like in flat cpo constructions),
- a function $effect: M\times(A\cup\{\delta,\varepsilon\})\rightarrow M$ such that: (a) $effect(\perp,a)=\perp$, for all $a\in A$, (b) $effect(s,\delta)=\perp$, for all $s\in M$, (c) $effect(s,\varepsilon)=s$, for all $s\in M$,
- a function $test:\Psi\rightarrow(M\rightarrow\{0,1\})$

We extend the domain of the effect function to atomic tests and their negations:

$$effect(s,\psi) = \begin{cases} s & \text{if } test(\psi)(s)=1 \\ \perp & \text{otherwise} \end{cases} \quad \text{for } s\in M \text{ and } \psi\in\Psi.$$

$$effect(s,\neg\psi) = \begin{cases} s & \text{if } test(\psi)(s)=0 \\ \perp & \text{otherwise} \end{cases} \quad \text{for } s\in M \text{ and } \psi\in\Psi.$$

Furthermore, we extend the effect function to finite non-empty sequences:

$eff_a(s) := effect(s,a)$
$eff_{a\sigma}(s) := eff_\sigma(eff_a(s))$, where σ is a sequence of elements from $A\cup\Psi\cup\{\neg\phi|\phi\in\Psi\}\cup\{\delta,\varepsilon\}$.

A process embodies a family of successful computations, i.e., complete, successful behaviour of a process observable in a finite amount of time: one performance of a process is just the execution of a finite sequence of atomic actions and atomic tests *not including* δ and/or ε. It is standard to formalize finite successful computations with help of termination traces (cf. [Gl] for an overview of trace notions). The undecided dual character of the atomic tests forces us to include them in the standard notion of trace. We will call this extended notion *path*. The possible paths of one particular process p form the *path set* Paths(p) of p. The *termination trace set* TerTr(p) (in [Gl] one finds the notation $\surd$-tr(p)) can be defined with help of paths sets: replace in p all tests ϕ by ε and then take the path set.

Examples: $Paths((a\varepsilon\delta+b+\delta+\phi)a+(\phi+\varepsilon)) = \{ba,\phi a,\phi,<>\}$
$TerTr((a\varepsilon\delta+b+\delta+\phi)a+(\phi+\varepsilon)) = \{ba,a,<>\}$
$Paths(X^{x=\phi ax+\neg\phi}) = \{(\phi a)^{n+1}\neg\phi \mid n\in \mathbf{N}\}$
$TerTr(X^{x=\phi ax+\neg\phi}) = \{a^{n+1} \mid n\in \mathbf{N}\}$

With help of the notion of path we can define a semantic equality (cf. 2.2.4) between processes: p=q if and only if the process p transforms a state s into a state s' only if q does as well, and vice versa, where p transforms s into s' if there is a path in the path set of p such that the effect of this path on s is s'.

2.2.2. DEFINITION. By induction on the structure of $p\in BPT_{rec}(A,\Psi)$ we define
$Paths: BPT_{rec}(A,\Psi)\to P((A\cup\Psi^{\#}\setminus\{\delta,\varepsilon\})^*)$,
where $P((A\cup\Psi^{\#}\setminus\{\delta,\varepsilon\})^*)$ stands for the power set of $(A\cup\Psi^{\#}\setminus\{\delta,\varepsilon\})^*$:

(i) $Paths(a) := \{a\}$ for $a\in A$
(ii) $Paths((\neg)^{2n}\psi) := \{\psi\}$ for $\psi\in\Psi$
(iii) $Paths((\neg)^{2n+1}\psi) := \{\neg\psi\}$ for $\psi\in\Psi$
(iv) $Paths(\delta) := \varnothing$
(v) $Paths(\varepsilon) := \{<>\}$ (i.e., the empty sequence)
(vi) $Paths(p+q) := Paths(p)\cup Paths(q)$
(vii) $Paths(pq) := \{\sigma\sigma' \mid \sigma\in Paths(p)\wedge\sigma'\in Paths(q)\} \approx Paths(p)xPaths(q)$
(viii) $Paths(\pi_n(p)) := \{\sigma\in Paths(p) \mid \sigma \text{ contains at most n atomic actions}\}$
(ix) If $E = \{x_i=t_i(x_1,...,x_k) \mid 1\le i\le k\}$, define for $1\le i\le k$ and $n\in\mathbf{N}$
$t_i^0 := \delta$
$t_i^{n+1} := t_i(t_1^n,...t_k^n)$
then
$Paths(X_i^E) := \bigcup_{n\in\mathbf{N}} Paths(t_i^n)$ for $1\le i\le k$.

Paths will be treated as processes by identifying the empty sequence <> with ε again, and non-empty words with the corresponding sequential compositions.

We give some examples stressing that path sets contain only the succesful terminated (i.e. finite) paths:
$Paths(X^{x=ax+b}) = \{a^nb \mid n\in\mathbf{N}\}$
$Paths(X^{x=xa+b}) = \{ba^n \mid n\in\mathbf{N}\}$
$Paths(X^{x=x+b}) = \{b\}$
$Paths(X^{x=ax}) = \bigcup_{n\in\mathbf{N}} Paths(a^n\delta) = \varnothing$
$Paths(X^{x=b}) = \{b\}$

Paths nor traces are immediately suitable to distinguish processes. Traces ignore tests, and paths sometimes pay too much attention to them:
$TerTr(a\phi+b) = \{a,b\} = TerTr(a+b)$
$Paths(a) = \{a\} \neq \{\phi a,\neg\phi a\} = Paths((\phi+\neg\phi)a)$.

We solve this problem by considering instead of Paths(p) a set $Paths_\Phi(p)$ of all paths that are Φ-*consistent* with some path of p, where Φ is some fixed, finite set of atomic tests. We will introduce this concept next.

A path of p is of the following form:

$$(\phi_{01}\cdots\phi_{0n_0})a_1(\phi_{11}\cdots\phi_{1n_1})a_2\ldots a_k(\phi_{k1}\cdots\phi_{kn_k}) \quad (k\geq 0,\ n_i\geq 0)$$

or, for short

$$\Phi_0 a_1 \Phi_1 a_2 \ldots a_k \Phi_k.$$

Let $\Phi=\{\phi_1,\ldots,\phi_n\}$ be a fixed finite set of atomic tests. Let us call a sequential composite of basic tests $\psi_1\ldots\psi_n$, where ψ_i equals either ϕ_i or $\neg\phi_i$ for all $1\leq i\leq n$ a *valuation* v over Φ is .

We say that

$$v_0 a_1 v_1 a_2 \ldots a_m v_m, \text{ where } v_i \text{ are valuations of } \Phi \qquad (*)$$

is Φ-*consistent* with

$$\Phi_0 a_1 \Phi_1 a_2 \ldots a_m \Phi_m$$

if all elements of Φ_i occur with the same sign (more precise: parity of the number of negations) in the valuation v_i for $0\leq i\leq m$.

Finally we will need one more notation for path sets: $\text{Paths}_p(r)$ will denote all nonempty sequences in $(A\cup\Psi)^*$ that have the form of (*) and are Φ-consistent with a path in Path(r), where Φ now is the set of all atomic tests that occur in p.

It is a well-known result of basic process algebra that with help of additional *trace axioms* processes without tests can be rewritten as a sum of traces. In case of recursion one has to be careful with possible infinite sums. For processes with tests we have a similar result for paths:

2.2.3. LEMMA. Let empty sums be equal to δ by definition.

(i) $\text{BPT}_{\text{rec}}(A,\varnothing)\vdash \pi_n(p) = \sum_{\sigma\in \text{TerTr}(\pi_n(p))}\sigma$ for any process p of $\text{BPT}_{\text{rec}}(A,\varnothing)$ and $n\in\mathbb{N}$.

(ii) $\text{BPT}_{\text{rec}}(A,\Psi)\vdash \pi_n(p) = \sum_{\sigma\in \text{Paths}_p(\pi_n(p))}\sigma$ for any process p of $\text{BPT}_{\text{rec}}(A,\Psi)$ and $n\in\mathbb{N}$.

□

With help of paths we give a semantical definition of a binary predicate p(s,s') on M×M for each closed process expression p in $\text{BPT}_{\text{rec}}(A,\Psi)$. This enables us also to define a semantic notion of equality of processes. In the following section we will extend this logic to give a syntactic definition of p(s,s').

2.2.4. DEFINITION.

(i) $M\models p(s,s')$ if there exists a path σ in $\text{Paths}_p(p)$ such that $M\models s'=\text{eff}_\sigma(s)$.

(ii) $M\models p=q$ if $M\models\forall s,s'\neq\perp[p(s,s')\leftrightarrow q(s,s')]$.

We have the following relationship between provability from the axioms of the process algebra $\text{BPT}_{\text{rec}}(A,\Psi)$, state semantics and path semantics:

2.2.5. COMPLETENESS THEOREM. The following are equivalent:

(i) $BPT_{rec}(A,\Psi) \vdash p=q$

(ii) $Paths_{pq}(p) = Paths_{pq}(q)$

(iii) $M \models p=q$ for all models M.

PROOF.

Given a fixed, finite reference set Φ of atomic tests, as before, we define $<M_\Phi,effect,test,\perp>$ as follows.

The states of M_Φ are $\perp$ together with all pairs of the form $<a_1...a_n,V>$ where $a_1...a_n \in A^*$ is a (possible empty) sequence of atomic actions, and V is a function from natural numbers to valuations over Φ.

We define

$eff_a(<a_1...a_n,V>):=<a_1...a_n a,V>$

$test_\phi(<a_1...a_n,V>):=1$ if ϕ occurs positively in the valuation V(n)

$test_{\neg\phi}(<a_1...a_n,V>):=1$ if ϕ occurs negatively in the valuation V(n)

By definition it follows:

$$eff_\phi(<a_1...a_n,V>) = \begin{cases} <a_1...a_n,V> & \phi \in V(n) \\ \perp & \text{otherwise} \end{cases}$$

$$eff_{\neg\phi}(<a_1...a_n,V>) = \begin{cases} <a_1...a_n,V> & \neg\phi \in V(n) \\ \perp & \text{otherwise} \end{cases}$$

Let M_{pq} denote the model constructed for the set of atomic tests that occur in the processes p and q.

We can now give the proof that (iii)⇒(ii)⇒(i). Soundness, that is (i)⇒(iii), remains. This is but a standard verification of all axioms.

$$\begin{aligned}
M_{pq} \models p=q &\Leftrightarrow M_{pq} \models \forall s,s' \neq \perp [p(s,s') \leftrightarrow q(s,s')] \\
&\Leftrightarrow \forall s,s' \in M_{pq}\backslash\{\perp\} (M_{pq} \models p(s,s') \Leftrightarrow M_{pq} \models q(s,s')) \\
&\Leftrightarrow \forall s,s' \in M_{pq}\backslash\{\perp\} (\exists \sigma \in Paths_{pq}(p)\ eff_\sigma(s)=s' \Leftrightarrow \exists \sigma \in Paths_{pq}(q)\ eff_\sigma(s)=s') && (\$) \\
&\Rightarrow Paths_{pq}(p)=Paths_{pq}(q) && (\$\$) \\
&\Rightarrow Paths_{pq}(\pi_n(p))=Paths_{pq}(\pi_n(q)) \text{ for all } n \in \mathbf{N} \\
&\Rightarrow BPT_{rec}(A,\Psi) \vdash \sum_{\sigma \in Paths_{pq}(\pi_n(p))} \sigma = \sum_{\sigma \in Paths_{pq}(\pi_n(q))} \sigma \text{ for all } n \in \mathbf{N} \\
&\Leftrightarrow BPT_{rec}(A,\Psi) \vdash \pi_n(p)=\pi_n(q) \text{ for all } n \in \mathbf{N} && \text{(lemma (2.2.4))} \\
&\Leftrightarrow BPT_{rec}(A,\Psi) \vdash p=q && \text{(AIP)}
\end{aligned}$$

Proof of ($)⇒($$). Suppose ($). To prove ($$) it suffices by symmetry to show that $Paths_{pq}(p) \subseteq Paths_{pq}(q)$. So, let $v_0a_1v_1a_2...a_nv_n$ be a path in $Paths_{pq}(p)$. Consider some element $<<>,V>$ in M_{pq} such that $V(i)=v_i$ for $1 \leq i \leq n$.

Clearly

$$eff_{v_0a_1v_1a_2...a_nv_n}(<<>,V>) = <<a_1...a_n>,V>.$$

Now by ($) we get $\sigma=w_0b_1w_1b_2...b_mw_m$ in $Paths_{pq}(q)$ such that

$$eff_{w_0b_1w_1b_2...b_mw_m}(<<>,V>) = <<a_1...a_n>,V>.$$

By construction $eff_{w_0b_1w_1b_2...b_mw_m}(<<>,V>) = \perp$ or $eff_{w_0b_1w_1b_2...b_mw_m}(<<>,V>) = <b_1...b_m,V>$. The first possibility is ruled out since $<<a_1...a_n>,V> \neq \perp$. Thus we get n=m, and $a_i=b_i$ for $1 \leq i \leq n$. Moreover, it follows from $eff_{w_0b_1w_1b_2...b_mw_m}(<<>,V>) \neq \perp$ that w_i is pq-consistent with V(i), hence because w_i and V(i) are valuations we conclude that $V(i)=w_i$ for $0 \leq i \leq n$.

Therefore $v_0a_1v_1a_2...a_nv_n = w_0b_1w_1b_2...b_mw_m \in Paths_{pq}(q)$.

Remark that if our initial set of atomic tests Ψ is infinite then this completeness proof requires for each pair p and q the construction of its own model M_{pq}. In case of a finite set Ψ of atomic tests one single model M_Φ suffices. However it is possible to modify the proof such that only one model is needed. Just take the union of all models M_Φ for all finite subsets Φ of Ψ.

□

3. THE ASSERTION LANGUAGE

What is the right logic to express correctness formulas of the form $\{\alpha\}p\{\beta\}$ in?
Suppose we have a language in which α and β are expressed and we can express with the formula p(s,s') the semantic concept *there exists a path* σ *in* Paths(p) *such that* $s'=eff_\sigma(s)$. Then the semantic definition of $\{\alpha\}p\{\beta\}$ is the interpretation in M of the formula

$$\forall s{\neq}\perp[\alpha(s){\rightarrow}\forall s'{\neq}\perp[p(s,s'){\rightarrow}\beta(s')]].$$

The naive way of writing p(s,s') is $s'=eff_{\sigma_1}(s)\vee s'=eff_{\sigma_2}(s)\vee s'=eff_{\sigma_3}(s)\vee\dots$, where $\sigma_1, \sigma_2, \sigma_3, \dots$ is an enumeration of the at most countable many paths of p. Infinitary logic provides us with a shorthand for countable disjunctions and conjunctions. Then p(s,s') becomes $\bigvee_{\sigma\in Paths(p)} s'=eff_\sigma(s)$.
$\mathcal{L}_{\omega_1\omega}$-languages (cf. [Ka] and [Ke]) are finitary first order languages to which countable disjunctions and conjunctions are added, respectively denoted by $\bigvee_{n\in\mathbb{N}}$ and $\bigwedge_{n\in\mathbb{N}}$. As for first order finitary predicate logics, there is a completeness theorem for $\mathcal{L}_{\omega_1\omega}$-languages. We will now give the definition of the $\mathcal{L}_{\omega_1\omega}$-assertion logic we will use here:

3.1. DEFINITION. The assertion language $\mathcal{L}$ for process correctness consists of the following ingredients:

(i) a sort S containing states
(ii) variables s,s',s",...
(iii) a constant $\perp$ for unsuccessful termination
(iv) unary functions eff_a for each $a\in A\cup\Psi\cup\{\neg\phi \mid \phi\in\Phi\}\cup\{\delta,\varepsilon\}$
(v) unary predicates ψ for each $\psi\in\Psi$
(vi) terms are generated by the grammar:

$$t ::= \perp \mid s \mid eff_a(t)$$

(vii) formulas are generated by the grammar:

$$\phi ::= t_1=t_2 \mid \psi(t) \mid \mathbf{true} \mid \mathbf{false} \mid \neg\phi \mid \phi_1\wedge\phi_2 \mid \phi_1\vee\phi_2 \mid \phi_1\rightarrow\phi_2 \mid \forall s\ \phi(s) \mid \exists s\ \phi(s)$$
$$\mid \bigwedge_{i\in I}\phi_i \mid \bigvee_{i\in I}\phi_i \quad \text{(where I is an infinite but countable index set)}$$

NOTATION.

- Sometimes we will use the notation $\bigwedge_{i\in I}\phi_i$ and $\bigvee_{i\in I}\phi_i$ if I is a finite set.
  In case of the empty set the $\bigwedge_{i\in I}\phi_i$ is understood to be **true** and $\bigvee_{i\in I}\phi_i$ is taken to be **false**.
- For paths σ we define

$$eff_\sigma(s):=\begin{cases} eff_a(s) & \text{if } \sigma=a \\ eff_{\sigma'}(eff_a(s)) & \text{if } \sigma=a\sigma' \end{cases}$$

- We will use p(s,s') as abbreviation for $\bigvee_{\sigma\in Paths(p)} s'=eff_\sigma(s)$ (recall the semantic definition (2.2.4.i)).

This infinitary logic can be interpreted in models in the same way as predicate logic. We will give a quick overview.

3.2. DEFINITION. Let a valuation of variables $v:Var\to M$ be given. By induction on the structure of terms we define valuation of terms in $<M,effect,test,\bot>$:

(i) $[\![\bot]\!]_v^M := \bot$

(ii) $[\![s]\!]_v^M := v(s)$

(iii) $[\![eff_a(t)]\!]_v^M := effect([\![t]\!]_v^M,a)$, where $a\in A\cup\Psi\cup\{\neg\phi|\phi\in\Psi\}\cup\{\delta,\varepsilon\}$

And extending the notation of (2.2.1)

(iv) $[\![eff_\sigma(t)]\!]_v^M = eff_\sigma([\![t]\!]_v^M)$ (sequences are always paths)

3.3. DEFINITION. We define satisfaction $\models$ of formulas in $\mathcal{L}$ by induction on the structure of formulas in $M = <M,effect,test,\bot>$:

(i) $M,v\models t_1=t_2 := [\![t_1]\!]_v^M = [\![t_2]\!]_v^M$

(ii) $M,v\models\psi(t)$ if $test(\psi)([\![t]\!]_v^M)=1$ (cf. (2.2.1))

(iii) $M,v\models\wedge_{i\in I}\phi_i$ if for any $i\in I$ we have $M,v\models\phi_i$

(iv) $M,v\models\vee_{i\in I}\phi_i$ if for some $i\in I$ we have $M,v\models\phi_i$

(cf. any textbook on elementary logic, e.g. [vD] for the interpretation of the not mentioned symbols)

We write $\Gamma\models\phi$ if for all M and v it holds that if for all $\psi\in\Gamma$ we have $M,v\models\psi$ then $M,v\models\phi$.

3.4. DEFINITION.

(i) A natural deduction calculus for the assertion language $\mathcal{L}$ consists of the usual rules of natural deduction for finitary first order predicate logic (cf. e.g. [vD]) to which are added the following introduction and elimination rules for the countable disjunctions and conjunctions (cf. [MR]):

$$\frac{\phi_1 \quad \phi_2 \quad \phi_3 \quad \cdots}{\wedge_{i\in\mathbb{N}}\phi_i}\ (\wedge\text{ introduction})$$

$$\frac{\wedge_{i\in\mathbb{N}}\phi_i}{\phi_i}\ \text{for any } i\ (\wedge\text{ elimination})$$

$$\frac{\phi_i}{\vee_{i\in I}\phi_i}\ \text{for any } i\ (\vee\text{ introduction})$$

$$\frac{\vee_{i\in I}\phi_i \qquad \begin{matrix}[\phi_1] & [\phi_2] & [\phi_3] & \\ \vdots & \vdots & \vdots & \cdots \\ \phi & \phi & \phi & \end{matrix}}{\phi}\ (\vee\text{ elimination})$$

- The square brackets [ ] indicate that $\{\alpha\}p\{\beta\}$ is a canceled hypothesis (cf. [vD] for this convention of natural deduction.

(ii) Axioms for our assertion logic concern the effect functions:

$\forall s\ \mathrm{eff}_\delta(s)=\bot$,
$\forall s\ \mathrm{eff}_\varepsilon(s)=s$,
$\forall s\forall s'\ [\mathrm{eff}_\psi(s)=s' \leftrightarrow ((\psi(s)\rightarrow s=s')\wedge(\neg\psi(s)\rightarrow s'=\bot))]$,
$\mathrm{eff}_a(\bot)=\bot$ for each $a\in A\cup\Psi\cup\{\neg\phi|\phi\in\Psi\}\cup\{\delta,\varepsilon\}$.

We write $\Gamma\vdash\phi$ if there is a natural deduction derivation of ϕ with hypotheses in Γ in this calculus.

3.5. DEDUCTION THEOREM (Karp) (cf. [Ka]). For any at most countable set of formulas Γ and for any formula ϕ it holds that

$\Gamma\vdash\phi \Leftrightarrow \vdash(\wedge_{\psi\in\Gamma}\psi)\rightarrow\phi$.

□

3.6. COMPLETENESS THEOREM (Karp) (cf. [Ka]). For any at most countable set of formulas Γ and for any formula ϕ it holds that

$\Gamma\vdash\phi \Leftrightarrow \Gamma\models\phi$.

□

The following lemma summarizes a number of useful properties of infinitary logic.

3.7. LEMMA. Let I and J be countable index sets:

(i) $\vdash\wedge_{i\in I\cap J}\phi_i \leftrightarrow (\wedge_{i\in I}\phi_i \wedge\wedge_{i\in J}\phi_i)$
(ii) $\vdash\vee_{i\in I\cup J}\phi_i \leftrightarrow (\vee_{i\in I}\phi_i \vee\vee_{i\in J}\phi_i)$
(iii) $\vdash\wedge_{(i,j)\in I\times J}\phi_{ij} \leftrightarrow \wedge_{i\in I}\wedge_{j\in J}\phi_{ij}$
(iv) $\vdash\vee_{(i,j)\in I\times J}\phi_{ij} \leftrightarrow \vee_{i\in I}\vee_{j\in J}\phi_{ij}$
(v) $\vdash\vee_{i\in I}(\phi\wedge\phi_i) \leftrightarrow (\phi\wedge\vee_{i\in I}\phi_i)$
(vi) $\vdash\wedge_{i\in I}(\phi\vee\phi_i) \leftrightarrow (\phi\vee\wedge_{i\in I}\phi_i)$
(vii) $\vdash\vee_{i\in I}\exists x\phi_i \leftrightarrow \exists x\vee_{i\in I}\phi_i$
(viii) $\vdash\wedge_{i\in I}\forall x\phi_i \leftrightarrow \forall x\wedge_{i\in I}\phi_i$
(ix) $\vdash(\vee_{i\in I}\phi_i\rightarrow\phi) \leftrightarrow \wedge_{i\in I}(\phi_i\rightarrow\phi)$
(x) $\vdash(\phi\rightarrow\wedge_{i\in I}\phi_i) \leftrightarrow \wedge_{i\in I}(\phi\rightarrow\phi_i)$

□

4. PARTIAL CORRECTNESS FORMULAS AND WEAKEST PRECONDITIONS

Before we introduce Hoare logic for processes in the next section, and before we prove a completeness theorem for it, we will first define the concept of weakest precondition for closed process terms and derive some useful properties using the infinitary assertion language of the previous section.

We can express the partial correctness formula $\{\alpha\}p\{\beta\}$ where p is a closed process expression, and α and β are formulas in $\mathcal{L}$ by the formula of the infinitary assertion logic $\forall s{\neq}\bot(\alpha(s){\rightarrow}\forall s'{\neq}\bot[p(s,s'){\rightarrow}\beta(s')])$. The interpretation of $\{\alpha\}p\{\beta\}$ in a model <M,effect,test,$\bot$> corresponds to the usual semantic formulation: any successfully terminating execution of p from a state $s{\neq}\bot$ satisfying α results in a state s' satisfying β.

4.1. DEFINITION. Let p be a closed process expression, and α and β be formulas in $\mathcal{L}$.
We denote by $\{\alpha\}p\{\beta\}$ any of the three equivalent formulas

- $\forall s{\neq}\bot(\alpha(s){\rightarrow}\forall s'{\neq}\bot[p(s,s'){\rightarrow}\beta(s')])$
- $\forall s(\alpha(s){\rightarrow}\forall s'[p(s,s'){\rightarrow}(s{=}\bot{\vee}s'{=}\bot{\vee}\beta(s'))])$
- $\forall s(\alpha(s){\rightarrow}\bigwedge_{\sigma\in Paths(p)}[s{=}\bot{\vee}eff_\sigma(s){=}\bot{\vee}\beta(eff_\sigma(s))])$.

REMARK:

(i) p has to be a closed term, otherwise Paths(p) is undefined.

(ii) the interpretation of $\{\alpha\}p\{\beta\}$ in a model <M,effect,test,$\bot$> corresponds to the intended semantic formulation: any successfully terminating execution of p from a state $s{\neq}\bot$ satisfying α results in a state s' satisfying β.

Weakest preconditions for program correctness are usually defined semantically: a *weakest precondition* for partial correctness of a *closed* process expression p with respect to an assertion β is an assertion $w(p,\beta)(s)$ such that for any model M and valuation v it holds that $M,v{\models}\{w(p,\beta)\}p\{\beta\}$ and $M,v{\models}\{\alpha\}p\{\beta\} \Leftrightarrow M,v{\models}\forall s(\alpha(s){\rightarrow}w(p,\beta)(s))$ for all α. Inspection of the syntactic definition of $\{\alpha\}p\{\beta\}$ reveals what the syntactic definition of $w(p,\beta)$ should be:

4.2. DEFINITION. Let p be a closed process expression and β be some assertion.
The weakest precondition for p w.r.t. β is the assertion:

$w(p,\beta)(s) := \forall s'\,[p(s,s'){\rightarrow}(s{=}\bot{\vee}s'{=}\bot{\vee}\beta(s'))]$.

In the remainder of the section we will prove a number of lemmas for weakest preconditions.

4.3. LEMMA. Let β be an assertion and p a closed process expression.

$\vdash\forall s\,[w(p,\beta)(s){\leftrightarrow}\bigwedge_{\sigma\in Paths(p)}w(\sigma,\beta)(s)]$

PROOF. The proof is the trivial observation that

$\vdash w(p,\beta)(s) \Leftrightarrow$
$\vdash\forall s'[p(s,s'){\rightarrow}(s{=}\bot{\vee}s'{=}\bot{\vee}\beta(s'))] \Leftrightarrow$
$\vdash\bigwedge_{\sigma\in Paths(p)}[s{=}\bot{\vee}eff_\sigma(s){=}\bot{\vee}\beta(eff_\sigma(s))] \Leftrightarrow$
$\vdash\bigwedge_{\sigma\in Paths(p)}w(\sigma,\beta)(s)$ □

4.4. LEMMA. Let β_i be assertions for $i \in I$ (I at most countable) and p a closed process expression.
$\vdash \forall s\ [w(p,\wedge_{i\in I}\beta_i)(s) \leftrightarrow \wedge_{i\in I}\, w(p,\beta_i)(s)]$

PROOF. The proof uses properties of the infinitary logic as listed in (3.8).

$\vdash w(p,\wedge_{i\in I}\beta_i)(s) \Leftrightarrow$
$\vdash \forall s'\ [p(s,s') \rightarrow (s=\bot \vee s'=\bot \vee \wedge_{i\in I}\beta_i(s')] \Leftrightarrow$
$\vdash \forall s'\ [p(s,s') \rightarrow \wedge_{i\in I}(s=\bot \vee s'=\bot \vee \beta_i(s')] \Leftrightarrow$
$\vdash \forall s' \wedge_{i\in I}\ [p(s,s') \rightarrow (s=\bot \vee s'=\bot \vee \beta_i(s')] \Leftrightarrow$
$\vdash \wedge_{i\in I} \forall s'\ [p(s,s') \rightarrow (s=\bot \vee s'=\bot \vee \beta_i(s')] \Leftrightarrow$
$\vdash \wedge_{i\in I}\, w(p,\beta_i)(s)$

□

4.5. LEMMA. Let α and β be assertions.

(i) $\vdash \forall s(w(\delta,\beta)(s) \leftrightarrow \mathbf{true})$
(ii) $\vdash \forall s(w(\varepsilon,\beta)(s) \leftrightarrow \beta(s))$
(iii) $\vdash \forall s(w(a,\beta)(s) \leftrightarrow \beta(\mathrm{eff}_a(s)))$
(iv) $\vdash \forall s(w(\psi,\beta)(s) \leftrightarrow (\psi(s) \rightarrow \beta(s)))$
(v) $\vdash \forall s(w(p+q,\beta)(s) \leftrightarrow (w(p,\beta)(s) \wedge w(q,\beta)(s)))$
(vi) $\vdash \forall s(w(pq,\beta)(s) \leftrightarrow w(p,w(q,\beta))(s))$

PROOF. (i)-(iv) are similar, for instance:

(iv) $\vdash w(\psi,\beta)(s) \Leftrightarrow$
$\vdash \wedge_{\sigma\in \mathrm{Paths}(\psi)}[s=\bot \vee \mathrm{eff}_\sigma(s)=\bot \vee \beta(\mathrm{eff}_\sigma(s))] \Leftrightarrow$ (recall (3.4.ii))
$\vdash (\psi(s) \rightarrow \beta(s)) \vee (\neg\psi(s) \rightarrow \mathbf{true}) \Leftrightarrow$
$\vdash \psi(s) \rightarrow \beta(s)$

(v) $\vdash w(p+q,\beta)(s) \Leftrightarrow$
$\vdash \wedge_{\sigma\in \mathrm{Paths}(p+q)}\, w(\sigma,\beta)(s) \Leftrightarrow$
$\vdash \wedge_{\sigma\in \mathrm{Paths}(p)\cup\mathrm{Paths}(q)}\, w(\sigma,\beta)(s) \Leftrightarrow$
$\vdash \wedge_{\sigma\in \mathrm{Paths}(p)}\, w(\sigma,\beta)(s) \wedge \wedge_{\sigma\in \mathrm{Paths}(q)}\, w(\sigma,\beta)(s) \Leftrightarrow$
$\vdash w(p,\beta)(s) \wedge w(q,\beta)(s)$

(vi) $\vdash w(pq,\beta)(s) \Leftrightarrow$
$\vdash \wedge_{\sigma\in \mathrm{Paths}(pq)}\, w(\sigma,\beta)(s) \Leftrightarrow$
$\vdash \wedge_{(\sigma',\sigma'')\in \mathrm{Paths}(p)\times\mathrm{Paths}(q)}\, w(\sigma'\sigma'',\beta)(s) \Leftrightarrow$
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)} \wedge_{\sigma''\in \mathrm{Paths}(q)}\, w(\sigma'\sigma'',\beta)(s) \Leftrightarrow$
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)} \wedge_{\sigma''\in \mathrm{Paths}(q)}\, (s=\bot \vee \mathrm{eff}_{\sigma'\sigma''}(s)=\bot \vee \beta(\mathrm{eff}_{\sigma'\sigma''}(s))) \Leftrightarrow$
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)} \wedge_{\sigma''\in \mathrm{Paths}(q)}\, (s=\bot \vee \mathrm{eff}_{\sigma''}(\mathrm{eff}_{\sigma'}(s))=\bot \vee \beta(\mathrm{eff}_{\sigma''}(\mathrm{eff}_{\sigma'}(s)))) \Leftrightarrow$ (note: $s=\bot \vdash \mathrm{eff}_{\sigma''}(\mathrm{eff}_{\sigma'}(s))=\bot$)
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)} \wedge_{\sigma''\in \mathrm{Paths}(q)}\, (s=\bot \vee \mathrm{eff}_{\sigma'}(s)=\bot \vee (\mathrm{eff}_{\sigma''}(\mathrm{eff}_{\sigma'}(s))=\bot \vee \beta(\mathrm{eff}_{\sigma''}(\mathrm{eff}_{\sigma'}(s)))) \Leftrightarrow$
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)} \wedge_{\sigma''\in \mathrm{Paths}(q)}\, (s=\bot \vee \mathrm{eff}_{\sigma'}(s)=\bot \vee w(\sigma'',\beta)(\mathrm{eff}_{\sigma'}(s))) \Leftrightarrow$
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)} \wedge_{\sigma''\in \mathrm{Paths}(q)}\, w(\sigma',w(\sigma'',\beta))(s) \Leftrightarrow$
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)}\, w(\sigma',\wedge_{\sigma''\in \mathrm{Paths}(q)} w(\sigma'',\beta))(s) \Leftrightarrow$
$\vdash \wedge_{\sigma'\in \mathrm{Paths}(p)}\, w(\sigma',w(q,\beta))(s) \Leftrightarrow$

$\vdash w(p,w(q,\beta))(s)$

We do not claim this is the most elegant way to prove (vi), but within the formalism of the infinitary assertion language the proof is a long one-liner that can not be missed.

□

5. HOARE LOGIC FOR RECURSION FREE PROCESSES

In this section we will define a Hoare logic for *partial correctness* formulas of recursion free closed process expressions in BPT(A,Ψ). From the semantics for the assertion logic and the interpretation of partial correctness formulas we obtain a semantics for the Hoare logic. We will prove strong completeness for the Hoare logic using the completeness theorem of the assertion logic and the relation between the Hoare logic and the assertion logic.

Being able to express our partial correctness formulas in the assertion language it is natural to investigate whether we the derivation of the various axioms and rules of Hoare logic (cf. 5.2.) in our assertion logic, instead of checking them semantically:

5.1 LEMMA.

(i) $\vdash \{w(p,\beta)\}p\{\beta\}$ for $p \in A \cup \Psi \cup \{\neg\phi | \phi \in \Psi\} \cup \{\varepsilon,\delta\}$

(ii) $\vdash \{\alpha\}p\{\beta\}$ and $\vdash \{\alpha\}q\{\beta\}$ if and only if $\vdash \{\alpha\}p+q\{\beta\}$

(iii) if $\vdash \{\alpha\}p\{\beta\}$ and $\vdash \{\beta\}q\{\gamma\}$ then $\vdash \{\alpha\}pq\{\gamma\}$

(iv) if $\vdash \alpha \to \alpha'$ and $\vdash \{\alpha'\}p\{\beta'\}$ and $\vdash \beta' \to \beta$ then $\vdash \{\alpha\}p\{\beta\}$

PROOF.

(i) Trivial, using the definitions of $p(s,s')$ and $w(p,\beta)$ for $p \in A \cup \Psi \cup \{\neg\phi | \phi \in \Psi\} \cup \{\varepsilon,\delta\}$.

(ii) $\vdash \{\alpha\}p\{\beta\}$ and $\vdash \{\alpha\}q\{\beta\} \Leftrightarrow$
$\vdash \{\alpha\}p\{\beta\} \wedge \{\alpha\}q\{\beta\} \Leftrightarrow$
$\vdash \forall s(\alpha(s) \to (w(p,\beta)(s)) \wedge w(q,\beta)(s))) \Leftrightarrow$
$\vdash \forall s(\alpha(s) \to w(p+q,\beta)(s)) \Leftrightarrow$
$\vdash \{\alpha\}p+q\{\beta\}$

(iii) $\vdash \{\alpha\}p\{\beta\}$ and $\vdash \{\beta\}q\{\gamma\} \Rightarrow$
$\vdash \{\alpha\}p\{\beta\} \wedge \{\beta\}q\{\gamma\} \Rightarrow$
$\vdash \forall s(\alpha(s) \to \forall s'[p(s,s') \to (s=\bot \vee s'=\bot \vee \beta(s'))]) \wedge \forall s'(\beta(s') \to w(q,\gamma)(s')) \Rightarrow$
$\vdash \forall s(\alpha(s) \to \forall s'[p(s,s') \to (s=\bot \vee s'=\bot \vee w(q,\gamma)(s'))] \Rightarrow$
$\vdash \forall s(\alpha(s) \to w(p,(w(q,\gamma))(s))) \Rightarrow$
$\vdash \forall s(\alpha(s) \to w(pq,\gamma)(s)) \Rightarrow$
$\vdash \{\alpha\}pq\{\gamma\}$

(iv) $\vdash \alpha \to \alpha'$ and $\vdash \{\alpha'\}p\{\beta'\}$ and $\vdash \beta' \to \beta \Rightarrow$
$\vdash \forall s[(\alpha(s) \to \alpha'(s)) \wedge (\alpha'(s) \to \forall s'[p(s,s') \to (s=\bot \vee s'=\bot \vee \beta'(s'))]) \wedge (\beta'(s) \to \beta(s))] \Rightarrow$
$\vdash \forall s(\alpha(s) \to \forall s'[p(s,s') \to (s=\bot \vee s'=\bot \vee \beta(s'))]) \Rightarrow$

$\vdash \{\alpha\}p\{\beta\}$.

□

5.2 DEFINITION of a proof system for Hoare logic for closed process expressions without recursion in BPT(A,Ψ). The building elements of the proofs are partial correctness formulas $\{\alpha\}p\{\beta\}$ and closed formulas of the form $\forall s(\alpha(s)\rightarrow\beta(s))$ which we will abbreviate by $\alpha\rightarrow\beta$:

(i) $\dfrac{}{\{\alpha\}\delta\{\beta\}}$

(ii) $\dfrac{}{\{\alpha\}\varepsilon\{\alpha\}}$

(iii) $\dfrac{}{\{\gamma\}a\{\beta\}}$, where γ stands for the predicate $\phi(s)\leftrightarrow\beta(\mathrm{eff}_a(s))$ and $a\in A$

(iv) $\dfrac{}{\{\psi\rightarrow\beta\}\psi\{\beta\}}$, for each basic test $\psi\in\Psi^{\#}\setminus\{\varepsilon,\delta\}$

(v) $\dfrac{\{\alpha\}p\{\beta\}\quad\{\alpha\}q\{\beta\}}{\{\alpha\}p+q\{\beta\}}$ *sum introduction*

(vi) $\dfrac{\{\alpha\}p\{\beta\}\quad\{\beta\}q\{\gamma\}}{\{\alpha\}pq\{\gamma\}}$ *sequential composition introduction*

(vii) $\dfrac{\alpha\rightarrow\alpha'\quad\{\alpha'\}p\{\beta'\}\quad\beta'\rightarrow\beta}{\{\alpha\}p\{\beta\}}$ *consequence rule*

Let us denote derivability for partial correctness formulas in this proof system by $\vdash_F$ in contrast to the derivability $\vdash$ of the assertion logic and two extions of $\vdash_F$ in the next section. Let Th(Γ) denote $\{\alpha\rightarrow\beta \mid \Gamma\vdash\forall s(\alpha(s)\rightarrow\beta(s))\}$.

5.3. EXAMPLE:

We derive $\dfrac{}{\{\alpha\}\psi\{\alpha\wedge\psi\}}$ for any atomic test $\psi\in\Psi$ by the following application of the consequence rule:

$$\frac{\alpha\rightarrow(\psi\rightarrow(\alpha\wedge\psi))\quad\dfrac{}{\{\psi\rightarrow(\alpha\wedge\psi)\}\psi\{\alpha\wedge\psi\}}}{\{\alpha\}\psi\{\alpha\wedge\psi\}}$$

(ii) Define: **if** ψ **then** p **else** q := $\psi p+\neg\psi q$

We derive the Hoare rule for **if then else**: $\dfrac{\{\alpha\wedge\psi\}p\{\beta\}\quad\{\alpha\wedge\neg\psi\}q\{\beta\}}{\{\alpha\}\textbf{if }\psi\textbf{ then p else q}\{\beta\}}$ as follows

$$\frac{\dfrac{\dfrac{}{\{\alpha\}\psi\{\alpha\wedge\psi\}}\quad\{\alpha\wedge\psi\}p\{\beta\}}{\{\alpha\}\psi p\{\beta\}}\qquad\dfrac{\dfrac{}{\{\alpha\}\neg\psi\{\alpha\wedge\neg\psi\}}\quad\{\alpha\wedge\neg\psi\}q\{\beta\}}{\{\alpha\}\neg\psi q\{\beta\}}}{\dfrac{\{\alpha\}\psi p+\neg\psi q\{\beta\}}{\{\alpha\}\textbf{if }\psi\textbf{ then p else q}\{\beta\}}\text{(def)}}$$

5.4. TRANSLATION THEOREM. For Hoare logic of closed process expressions in BPT(A,Ψ):

(i) If Γ is a set of partial correctness formulas and/or implications, then
$\Gamma\vdash_F\{\alpha\}p\{\beta\} \Rightarrow \Gamma\vdash\{\alpha\}p\{\beta\}$.

(ii) If Γ is a set of formulas in the assertion language, then
$\Gamma\vdash\{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma)\vdash_F\{\alpha\}p\{\beta\}$.

PROOF.

(i) This follows by induction on the structure of the *proof* of $\Gamma\vdash_F\{\alpha\}p\{\beta\}$ using the lemmas (5.1) and (4.5).

(ii) By induction on the structure of the *process* p:

- case p, where $p\in A\cup\Psi\cup\{\varepsilon,\delta\}$: We transform a derivation $\Gamma\vdash\{\alpha\}p\{\beta\}$ into a derivation of $\Gamma\vdash\alpha\rightarrow w(p,\beta)$.

Consider the following proof in $\vdash_F$:

$$\frac{\alpha\rightarrow w(p,\beta) \quad \overline{\{w(p,\beta)\}p\{\beta\}}}{\{\alpha\}p\{\beta\}}.$$

By definition $\alpha\rightarrow w(p,\beta)\in Th(\Gamma)$ for $p\in A\cup\Psi\cup\{\delta\}$. Hence for all p in $A\cup\Psi\cup\{\varepsilon\}$ we have
$\Gamma\vdash\{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma)\vdash_F\{\alpha\}p\{\beta\}$.

- case p+q: Note that $\Gamma\vdash\{\alpha\}p+q\{\beta\}$ implies both $\Gamma\vdash\{\alpha\}p\{\beta\}$ and $\Gamma\vdash\{\alpha\}q\{\beta\}$. By induction we now have derivations $\Gamma\vdash_F\{\alpha\}p\{\beta\}$ and $\Gamma\vdash_F\{\alpha\}q\{\beta\}$. From these we get a derivation for $\Gamma\vdash_F\{\alpha\}p+q\{\beta\}$ by applying +-introduction.

- case pq: Assume $\Gamma\vdash\{\alpha\}pq\{\beta\}$. Hence $\Gamma\vdash\{\alpha\}p\{w(q,\beta)\}$ and $\vdash\{w(q,\beta)\}q\{\beta\}$.

By induction hypothesis we now have proofs
$\Gamma\vdash_F\{\alpha\}p\{w(q,\beta)\}$.
and
$\Gamma\vdash_F\{w(q,\beta)\}q\{\beta\}$
Now apply sequential composition introduction to get a derivation for $\Gamma\vdash_F\{\alpha\}pq\{\beta\}$.

□

5.5. COROLLARY: Completeness for Hoare logic of closed process expressions in BPT(A,Ψ):

(i) (soundness) If Γ is a set of partial correctness formulas and/or implications, then
$\Gamma\vdash_F\{\alpha\}p\{\beta\} \Rightarrow \Gamma\models\{\alpha\}p\{\beta\}$.

(ii) (adequacy) If Γ is a countable set of formulas in the assertion language, then
$\Gamma\models\{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma)\vdash_F\{\alpha\}p\{\beta\}$,
where $Th(\Gamma):=\{\alpha\rightarrow\beta \mid \Gamma\vdash\forall s(\alpha(s)\rightarrow\beta(s))\}$.

PROOF.
Combine the translation theorem (5.4) with the completeness theorem (3.6) of the infinitary assertion logic.

□

6. HOARE LOGICS FOR PROCESSES WITH RECURSION

We will present two reasonable ways of extending the previous Hoare logic for processes with rules for recursion. Surprisingly enough the different extensions are not equivalent: they are complete for different classes of processes. We will consider generalized versions of the rules:

$$\frac{\{\alpha\}t^n(\delta)\{\beta\} \text{ for all } n\in\mathbb{N}}{\{\alpha\}X^{x=t(x)}\{\beta\}}$$ *infinitary induction rule for one variable*

and

$$\frac{\begin{array}{c}[\{\alpha\}x\{\beta\}]\\ \vdots\\ \{\alpha\}t(x)\{\beta\}\end{array}}{\{\alpha\}X^{x=t(x)}\{\beta\}}$$ Scott's *induction rule for one variable* (cf. [dBS] or [dB]).

Comments on the natural deduction style notation in Scott's induction rule (cf. for more explanation a textbook of logic, e.g. [vD]):

- the square brackets [ ] indicate that $\{\alpha\}p\{\beta\}$ is a canceled hypothesis
- 'linear' reformulation: if $\Gamma,\{\alpha\}x\{\beta\}\vdash_G\{\alpha\}t(x)\{\beta\}$, then $\Gamma\vdash_G\{\alpha\}X^{x=t(x)}\{\beta\}$)

In the style of the previous section it is not difficult to prove a completeness theorem for Hoare logic with the infinitary induction rule, where the partial correctness formulas deal with closed process expressions p in $BPT_{rec}(A,\Psi)$.

Scott's induction rule introduces open process expressions in the Hoare calculus, which we avoided carefully in the previous section. We can deal with process variables if we interpret the proof scheme $\{\alpha\}x\{\beta\}\vdash_H\{\alpha\}t(x)\{\beta\}$ which occurs as hypothesis in Scott's induction rule as an abbreviation for: $\{\alpha\}p\{\beta\}\vdash_H\{\alpha\}t(p)\{\beta\}$ for all closed process expressions p, i.e., "taking the universal closure of $\{\alpha\}x\{\beta\}\vdash_H\{\alpha\}t(x)\{\beta\}$". Then Hoare logic with Scott's induction rule is not complete for processes involving arbitrary recursive specifications: the solution of the non-linear equation $x = axa+aa$ is a counter example, as we will show.

As in the previous section we first look at what the assertion logic suggests us for recursion:

6.1 LEMMA.

(i) $\vdash\{\alpha\}t^n(\delta)\{\beta\}$ for all $n\in\mathbb{N} \Leftrightarrow \vdash\{\alpha\}X^{x=t(x)}\{\beta\}$,

(ii) For a set of m specifying equations $\{x_i=t_i(x_1,...,x_m) \mid 1\le i\le m\}$ define $t_i^0 := \delta$ and $t_i^{n+1} := t_i(t_1^n,...t_m^n)$. If $\vdash\{\alpha_i\}t_i^n\{\beta_i\}$ for all $n\in\mathbb{N}$, all $1\le j\le m$, then $\vdash\{\alpha_k\}X_k\{\beta_k\}$ for all $1\le k\le m$.

PROOF.

(i) $\vdash\{\alpha\}t^n(\delta)\{\beta\}$ for all $n\in\mathbb{N} \Leftrightarrow$ (4.3)

$\vdash\bigwedge_{n\in\mathbb{N}}\bigwedge_{\sigma\in Paths(t^n(\delta))}\{\alpha\}\sigma\{\beta\} \Leftrightarrow$

$\vdash\bigwedge_{\sigma\in\bigcup_{n\in\mathbb{N}}Paths(t^n(\delta))}\{\alpha\}\sigma\{\beta\} \Leftrightarrow$ (2.2.2)&(4.3)

$\vdash\{\alpha\}X^{x=t(x)}\{\beta\}$.

(ii) Similar as (i). □

6.2 DEFINITION. We extend the Hoare logic of (5.2) to closed process expressions in $BPT_{rec}(A,\Psi)$ with the following rules (we extend the notation $\vdash_G$ to this first extension of $\vdash_F$ of the previous section 5):

(viii) $\dfrac{\{\alpha\}t^n(\delta)\{\beta\} \text{ for all } n\in \mathbf{N}}{\{\alpha\}X^{x=t(x)}\{\beta\}}$ *infinitary induction rule for one variable*

(ix) For a set of equations $\{x_i=t_i(x_1,...,x_m) \mid 1\le i\le m\}$:

$$\frac{\{\alpha_i\}t_i^n\{\beta_i\} \text{ for all } n\in \mathbf{N},\ \text{all } 1\le i\le m}{\{\alpha_k\}X_k\{\beta_k\}} \text{ for all } 1\le k\le m$$ *infinitary induction rule for* m *variables*

where $t_i^0 := \delta$ and $t_i^{n+1} := t_i(t_1^n,...t_m^n)$.

6.3. EXAMPLE.
Define: **while** ψ **do** p **od** := $X^{x=t(x)}$, where $t(x) = \psi px+\neg\psi$.
Then we can derive the corresponding Hoare rule

$$\frac{\{\alpha\wedge\psi\}p\{\alpha\}}{\{\alpha\}\textbf{while } \psi \textbf{ do } p \textbf{ od}\{\alpha\wedge\neg\psi\}}$$

as follows: We have the axiom $\{\alpha\}\delta\{\alpha\wedge\neg\psi\}$ and we also have $\{\alpha\}t^n(\delta)\{\alpha\wedge\neg\psi\}\vdash\{\alpha\}t^{n+1}(\delta)\{\alpha\wedge\neg\psi\}$ by the following argument using example (5.3.i):

$$\dfrac{\dfrac{\dfrac{(5.3.i)}{\{\alpha\}\psi\{\alpha\wedge\psi\}} \quad \dfrac{\{\alpha\wedge\psi\}p\{\alpha\} \quad [\{\alpha\}t^n(\delta)\{\alpha\wedge\neg\psi\}]}{\{\alpha\wedge\psi\}pt^n(\delta)\{\alpha\wedge\neg\psi\}}}{\{\alpha\}\psi pt^n(\delta)\{\alpha\wedge\neg\psi\}} \quad \dfrac{(5.3.i)}{\{\alpha\}\neg\psi\{\alpha\wedge\neg\psi\}}}{\{\alpha\}t^{n+1}(\delta)\{\alpha\wedge\neg\psi\}}$$

Hence by an induction argument we get $\vdash_G\{\alpha\}t^n(\delta)\{\alpha\wedge\neg\psi\}$ for all $n\in\mathbf{N}$, from which $\vdash\{\alpha\}X^{x=\psi px+\neg\psi}\{\alpha\wedge\neg\psi\}$ follows by application of the infinitary induction rule. So we have shown $\{\alpha\wedge\psi\}p\{\alpha\}\vdash_G\{\alpha\}$**while** ψ **do** p **od**$\{\alpha\wedge\neg\psi\}$. We will come back to this example when we have introduced Scott's induction rule (6.7).

6.4. TRANSLATION THEOREM. For Hoare logic of closed process expressions in $BPT_{rec}(A,\Psi)$:

(i) If Γ is a set of partial correctness formulas and/or implications, then
$\Gamma\vdash_G\{\alpha\}p\{\beta\} \Rightarrow \Gamma\vdash\{\alpha\}p\{\beta\}$.

(ii) If Γ is a countable set of formulas in the assertion language, then
$\Gamma\vdash\{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma)\vdash_G\{\alpha\}p\{\beta\}$,
where as in (5.2) $Th(\Gamma):=\{\alpha\to\beta \mid \Gamma\vdash\forall s(\alpha(s)\to\beta(s))\}$.

PROOF. We only have to add the induction steps concerning recursion to the proof of the translation theorem (5.4) for non-recursive processes.

(ii) case $X^{x=t(x)}$ (one variable only).

Assume we have a proof $\Gamma\vdash\{\alpha\}X\{\beta\}$. Then by $\wedge$-elimination we have a proof of $\Gamma\vdash\{\alpha\}\sigma\{\beta\}$ for all $\sigma\in$ Paths(X). By induction hypothesis we have $\Gamma\vdash_G\{\alpha\}\sigma\{\beta\}$ for any path $\sigma\in$ Paths(X)=$\bigcup_{n\in\mathbb{N}}$Paths($t^n(\delta)$). Hence using $\wedge$-introduction we get $\Gamma\vdash_G\{\alpha\}t^n(\delta)\{\beta\}$ for all $n\in\mathbb{N}$. By the recursion rule we derive $\Gamma\vdash_G\{\alpha\}X\{\beta\}$.
The general case can be proved similar.

□

6.5. COROLLARY. Strong completeness theorem for Hoare logic of closed process expressions in $BPT_{rec}(A,\Psi)$:

(i) (soundness) If Γ is a set of partial correctness formulas and/or implications, then
$\Gamma\vdash_G\{\alpha\}p\{\beta\}\Rightarrow\Gamma\models\{\alpha\}p\{\beta\}$.

(ii) (adequacy) If Γ is a countable set of formulas in the assertion language, then
$\Gamma\models\{\alpha\}p\{\beta\}\Rightarrow Th(\Gamma)\vdash_G\{\alpha\}p\{\beta\}$,
where $Th(\Gamma):=\{\alpha\to\beta\mid\Gamma\vdash\forall s(\alpha(s)\to\beta(s))\}$.

PROOF. Combine the completeness theorem for infinitary logic (3.6) with the translation theorem (6.4).

□

6.6. LEMMA. The following rule is a derived rule in the proof system $\vdash_G$:

$$\frac{\begin{array}{c}[\{\alpha\}p\{\beta\}]\\ \vdots\\ \{\alpha\}t(p)\{\beta\}\quad\text{for all closed processes p}\end{array}}{\{\alpha\}X^{x=t(x)}\{\beta\}}\ \textit{cp-induction rule}\ .$$

PROOF. Straightforward, by induction. To load the induction process we recall that the axiom $\{\alpha\}\delta\{\beta\}$. Hence, if $\{\alpha\}p\{\beta\}\vdash_G\{\alpha\}t(p)\{\beta\}$ for any closed p, then we find $\{\alpha\}t^n(\delta)\{\beta\}$ for all natural numbers n. Application of the infinitary induction rule gives us the desired conclusion $\{\alpha\}X^{x=t(x)}\{\beta\}$.

□

6.7. DEFINITION. We extend the Hoare logic of (5.2) to open process expressions in $BPT_{linrec}(A,\Psi)$ and we add the following rules (we extend the notation $\vdash_H$ to this second extension of $\vdash_F$ of the previous section 5):

(viii) $$\frac{\begin{array}{c}[\{\alpha\}x\{\beta\}]\\ \vdots\\ \{\alpha\}t(x)\{\beta\}\end{array}}{\{\alpha\}X^{x=t(x)}\{\beta\}}$$ Scott's *induction rule for one variable.*

And, more generally:

(ix) For a set of equations $\{x_i=t_i(x_1,\ldots,x_m)\mid 1\le i\le m\}$:

$$\frac{\begin{array}{c}[\{\alpha_1\}x_1\{\beta_1\},\ldots,\{\alpha_m\}x_m\{\beta_m\}]\\ \vdots\\ \{\alpha_k\}t_k(x_1,\ldots,x_m)\{\beta_k\}\end{array}}{\{\alpha_k\}X_k\{\beta_k\}}$$ for $1\le k\le m$, *Scott's rule for* m *variables*

6.8. EXAMPLE.

Define: **while** ψ **do** p **od** := X^E, where E is the *linear* equation $x = \psi px + \neg\psi$. Then we can derive the corresponding

Hoare rule $\dfrac{\{\alpha\wedge\psi\}p\{\alpha\}}{\{\alpha\}\textbf{while } \psi \textbf{ do } p \textbf{ od}\{\alpha\wedge\neg\psi\}}$ as follows (we use example (5.3.i)):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(5.3.i)}{\{\alpha\}\psi\{\alpha\wedge\psi\}} \quad \dfrac{\{\alpha\wedge\psi\}p\{\alpha\} \quad [\{\alpha\}x\{\alpha\wedge\neg\psi\}]}{\{\alpha\wedge\psi\}px\{\alpha\wedge\neg\psi\}}}{\{\alpha\}\psi px\{\alpha\wedge\neg\psi\}} \quad \dfrac{(5.3.i)}{\{\alpha\}\neg\psi\{\alpha\wedge\neg\psi\}}}{\{\alpha\}\psi px+\neg\psi\{\alpha\wedge\neg\psi\}}}{\{\alpha\}X^{x=\psi px+\neg\psi}\{\alpha\wedge\neg\psi\}}}{\{\alpha\}\textbf{while } \psi \textbf{ do } p \textbf{ od}\{\alpha\wedge\neg\psi\}}\,(\text{def})$$

We have to give a new interpretation to $\Gamma\models\{\alpha\}p\{\beta\}$ in case p and correctness formulas in Γ contain free process variables:

6.9. DEFINITION. We define $\Gamma\models\{\alpha\}p\{\beta\}$ to mean that for all substitutions σ of closed process expressions for all free process variables in $\Gamma\cup\{\alpha\}p\{\beta\}$ we have $\Gamma^\sigma\models\{\alpha\}p^\sigma\{\beta\}$.

The following lemma contains some properties for derivations in the Hoare calculus $\vdash_H$. The properties will be used in the proof of the translation theorem for the present Hoare calculus.

6.10. LEMMA.

(i) If there are derivations of $\Gamma\vdash_H\{\alpha\}p+q\{\gamma\}$ and $\Gamma\vdash_H\{\alpha\}pq\{\gamma\}$, then there are derivations of these formulas where the last rule is an instance of an introduction rule and not an instance of the consequence rule.

Let Γ be some set of partial correctness formulas.

(ii) If $Th(\Gamma)\vdash_H\{\alpha\}p\{\beta\}$ and $Th(\Gamma)\vdash_H\{\alpha\}p\{\gamma\}$ for a closed process term p in BPT(A,Ψ) then $Th(\Gamma)\vdash\{\alpha\}p\{\beta\wedge\gamma\}$

(iii) Let t and t' be linear terms with free variables $x_1,\ldots,x_n$ such that $BPT(A,\Psi)\vdash t=t'$. Then

$Th(\Gamma),\{\alpha_1\}x_1\{\beta_1\},\ldots,\{\alpha_n\}x_n\{\beta_n\}\vdash_H\{\alpha\}t\{\beta\}$

if and only if

$Th(\Gamma),\{\alpha_1\}x_1\{\beta_1\},\ldots,\{\alpha_n\}x_n\{\beta_n\}\vdash_H\{\alpha\}t'\{\beta\}$

(iv) Let $E=\{x_i=t_i \mid 1\le i\le n\}$ and $E'=\{x_i=t'_i \mid 1\le i\le n\}$ be sets of linear equations with free variables $x_1,\ldots,x_n$ such that $BPT(A,\Psi)\vdash t_i=t'_i$ for all $1\le i\le n$. Then

$Th(\Gamma)\vdash_H\{\alpha\}X_1^E\{\beta\}\Leftrightarrow Th(\Gamma)\vdash_H\{\alpha\}X_1^{E'}\{\beta\}$.

PROOF.

(i) Trivial.

(ii) By induction to the structure of the proof of the closed process expression p. The atomic case is crucial: Suppose we have $Th(\Gamma)\vdash_H\{\alpha\}a\{\beta\}$ and $Th(\Gamma)\vdash_H\{\alpha\}a\{\gamma\}$. By definition of $Th(\Gamma)$ this implies that $\Gamma\vdash\{\alpha\}a\{\beta\}$ and $\Gamma\vdash\{\alpha\}a\{\gamma\}$. But in the assertion logic one easily verifies that now $\Gamma\vdash\{\alpha\}a\{\beta\wedge\gamma\}$. Hence by the translation theorem for $\vdash_G$ (we are working recursion free) we now get $Th(\Gamma)\vdash_H\{\alpha\}a\{\beta\wedge\gamma\}$.

(iii) It is easy to prove by induction using distributivity that any linear term $t(x_1,...,x_n)$ is equivalent to a term in normal linear form:

$$t(x_1,...,x_n) = q + \sum_{1\le i\le n} q_i x_i, \text{ where } q \text{ and } q_i \text{ are closed processes.}$$

Example:

$$q(qx_1+p)+r+rx_1 = (qp+r)+(qq+r)x_1$$

It suffices to assume that t' is in normal linear form. A normal form can be found by repeated use of the axioms: $x(y+z)=xy+xz$ and $(x+y)z=xz+yz$. A formal proof goes by induction to the structure of t. We present the essential step of the proof for one variable: $Th(\Gamma),\{\alpha\}x\{\beta\} \vdash_H \{\alpha\}p(t_1(x)+t_2(x))\{\beta\} \Leftrightarrow Th(\Gamma),\{\alpha\}x\{\beta\} \vdash_H \{\alpha\}pt_1(x)+pt_2(x)\{\beta\}$.
Assume $Th(\Gamma),\{\alpha\}x\{\beta\} \vdash_H \{\alpha\}p(t_1(x)+t_2(x))\{\beta\}$, then we can construct a proof of the form:

$$\dfrac{Th(\Gamma) \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma\}t_1(x)\{\beta\}} \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma\}t_2(x)\{\beta\}}}{\dfrac{Th(\Gamma)}{\{\alpha\}p\{\gamma\}} \quad \dfrac{\{\gamma\}t_1(x)+t_2(x)\{\beta\}}{\{\alpha\}p(t_1(x)+t_2(x))\{\beta\}}}$$

which we can transform into a derivation of $Th(\Gamma),\{\alpha\}x\{\beta\} \vdash_H \{\alpha\}pt_1(x)+pt_2(x)\{\beta\}$

$$\dfrac{\dfrac{Th(\Gamma)}{\{\alpha\}p\{\gamma\}} \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma\}t_1(x)\{\beta\}}}{\{\alpha\}pt_1(x)\{\beta\}} \quad \dfrac{\dfrac{Th(\Gamma)}{\{\alpha\}p\{\gamma\}} \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma\}t_2(x)\{\beta\}}}{\{\alpha\}pt_2(x)\{\beta\}}}{\{\alpha\}pt_1(x)+pt_2(x)\{\beta\}}$$

Hence $Th(\Gamma),\{\alpha\}x\{\beta\} \vdash_H \{\alpha\}pt_1(x)+pt_2(x)\{\beta\}$. The transformation in the other direction is more of interest:

Assume $Th(\Gamma),\{\alpha\}x\{\beta\} \vdash_H \{\alpha\}pt_1(x)+pt_2(x)\{\beta\}$, then we can find a derivation of the form:

$$\dfrac{\dfrac{Th(\Gamma)}{\{\alpha\}p\{\gamma\}} \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma\}t_1(x)\{\beta\}}}{\{\alpha\}pt_1(x)\{\beta\}} \quad \dfrac{\dfrac{Th(\Gamma)}{\{\alpha\}p\{\gamma'\}} \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma'\}t_2(x)\{\beta\}}}{\{\alpha\}pt_2(x)\{\beta\}}}{\{\alpha\}pt_1(x)+pt_2(x)\{\beta\}}$$

Using (i) we can transform this proof into:

$$\dfrac{\dfrac{Th(\Gamma)}{\{\alpha\}p\{\gamma\wedge\gamma'\}} \quad \dfrac{\gamma\wedge\gamma''\to\gamma \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma\}t_1(x)\{\beta\}}}{\{\gamma\wedge\gamma'\}t_1(x)\{\beta\}} \quad \dfrac{\gamma\wedge\gamma''\to\gamma'' \quad \dfrac{Th(\Gamma) \quad \{\alpha\}x\{\beta\}}{\{\gamma'\}t_2(x)\{\beta\}}}{\{\gamma\wedge\gamma'\}t_2(x)\{\beta\}}}{\{\gamma\wedge\gamma'\}t_1(x)+t_2(x)\{\beta\}}}{\{\alpha\}p(t_1(x)+t_2(x))\{\beta\}}$$

(iv) corollary of (iii)

□

6.11. TRANSLATION THEOREM. For Hoare logic with Scott's induction rule of process terms in $BPT_{linrec}(A,\Psi)$:

(i) If Γ is a set of partial correctness formulas and/or implications, then
$\Gamma \vdash_H \{\alpha\}p\{\beta\} \Rightarrow \Gamma \vdash \{\alpha\}p\{\beta\}$.

(ii) If Γ is a countable set of formulas in the assertion language, then
$\Gamma \vdash \{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma) \vdash_H \{\alpha\}p\{\beta\}$.

PROOF. Again, we only have to add the induction steps concerning recursion to the completeness proof (5.4) for non-recursive processes.

(i) Lemma (6.5) and the soundness of $\vdash_G$ with respect to $\vdash$ deal with the soundness of Scott's rule.

(ii) We consider the case with one variable. Suppose we have $\Gamma\vdash\{\alpha\}X^{x=t(x)}\{\beta\}$, and we want to prove $Th(\Gamma)\vdash_H\{\alpha\}X^{x=t(x)}\{\beta\}$

By axioms $x(y+z)=xy+xz$ and $(x+y)z=xz+yz$ we can find an equation of the form $x=qx+r$, where q and r are closed process expressions in $BPT(A,\Psi)$ equivalent to the linear equation $x=t(x)$. In terms of weakest preconditions $\Gamma\vdash\{\alpha\}X^{x=t(x)}\{\beta\}$ is equivalent to $\Gamma\vdash\forall s[\alpha(s)\rightarrow\bigwedge_{n\in\mathbf{N}}w(t^n(\delta),\beta)(s)]$. Let us abbreviate $\forall s[\bigwedge_{n\in\mathbf{N}}w(t^n(\delta),\beta)]$ by γ. So we have $\Gamma\vdash\forall s[\alpha(s)\rightarrow\gamma(s)]$.

Since $\Gamma\vdash\forall s[\gamma(s)\rightarrow\bigwedge_{n\in\mathbf{N}}w(t^{n+1}(\delta),\beta)]$
$\Rightarrow\Gamma\vdash\forall s[\gamma(s)\rightarrow w(r,\beta)\wedge\bigwedge_{n\in\mathbf{N}}w(q,w(t^n(\delta),\beta)]$
$\Rightarrow\Gamma\vdash\forall s[\gamma(s)\rightarrow w(r,\beta)]$ and $\Gamma\vdash\forall s[\gamma(s)\rightarrow w(q,\bigwedge_{n\in\mathbf{N}}w(t^n(\delta),\beta)]$
$\Rightarrow\Gamma\vdash\{\gamma\}r\{\beta\}$ and $\Gamma\vdash\{\gamma\}q\{\bigwedge_{n\in\mathbf{N}}w(t^n(\delta),\beta)\}$
$\Rightarrow Th(\Gamma)\vdash_H\{\gamma\}r\{\beta\}$ and $Th(\Gamma)\vdash_H\{\gamma\}q\{\gamma\}$ (ind. hyp.)

$$\Rightarrow\ \dfrac{\dfrac{\dfrac{Th(\Gamma)}{\{\gamma\}q\{\gamma\}}\quad[\{\gamma\}x\{\beta\}]}{\{\gamma\}qx\{\beta\}}\quad\dfrac{Th(\Gamma)}{\{\gamma\}r\{\beta\}}}{\dfrac{Th(\Gamma)}{\alpha\rightarrow\gamma}\quad\dfrac{\{\gamma\}qx+r\{\beta\}}{\{\gamma\}X^{x=qx+r}\{\beta\}}}{\{\alpha\}X^{x=qx+r}\{\beta\}}$$

we see that $\Gamma\vdash\{\alpha\}X^{x=t(x)}\{\beta\}$ implies $Th(\Gamma)\vdash_H\{\alpha\}X^{x=qx+r}\{\beta\}$ and also $Th(\Gamma)\vdash_H\{\alpha\}X^{x=t(x)}\{\beta\}$ by lemma (6.10). □

6.12. COROLLARY. Completeness theorem for Hoare logic with Scott's induction rule of process expressions in $BPT_{linrec}(A,\Psi)$:

(i) (soundness) If Γ is a set of partial correctness formulas and/or implications, then $\Gamma\vdash_H\{\alpha\}p\{\beta\}\Rightarrow\Gamma\models\{\alpha\}p\{\beta\}$.

(ii) (adequacy) If Γ is a countable set of formulas in the assertion language, then $\Gamma\models\{\alpha\}p\{\beta\}\Rightarrow Th(\Gamma)\vdash_H\{\alpha\}p\{\beta\}$.

PROOF. Combine the completeness theorem for infinitary logic (3.6) with the above translation theorem (6.11). □

If we compare the present proof system $\vdash_H$ with the related proof system discussed in [Po], we note a number of differences in the chosen assertion logic and semantics, but the main difference is the nature of the completeness result. [Po] gives a completeness theorem relative to one model only. His argument that linearity is necessary in order to obtain a completeness result for Hoare logic with Scott's induction rule carries over to the present situation:

6.13. COUNTER EXAMPLE.

The restriction to processes of $BPT_{linrec}(A,\Psi)$ is necessary, as the following example shows.

Consider the non-linear equation $x=t(x)$ where $t(x)=axa+aa$ in a process language with $A=\{a\}$ and $\Psi=\{\alpha,\beta\}$. Take $\Gamma=\{\{\alpha\}a\{\beta\},\{\beta\}a\{\alpha\}\}$. Then for all $n\in\mathbb{N}$ we can show by simultaneous induction: $\Gamma\vdash\{\alpha\}t^n(\delta)\{\alpha\}$ and $\Gamma\vdash\{\beta\}t^n(\delta)\{\beta\}$. The induction steps are respectively $\Gamma,\ \{\alpha\}t^n(\delta)\{\alpha\}\vdash\{\beta\}t^{n+1}(\delta)\{\beta\}$ and $\Gamma,\ \{\beta\}t^n(\delta)\{\beta\}\vdash\{\alpha\}t^{n+1}(\delta)\{\alpha\}$. The first can be proved as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{\{\beta\}a\{\alpha\}}\quad\{\alpha\}t^n(\delta)\{\alpha\}}{\{\beta\}at^n(\delta)\{\alpha\}}\quad\dfrac{Th(\Gamma)}{\{\alpha\}a\{\beta\}}}{\{\beta\}at^n(\delta)a\{\beta\}}\quad\dfrac{\dfrac{\Gamma}{\{\beta\}a\{\alpha\}}\quad\dfrac{\Gamma}{\{\alpha\}a\{\beta\}}}{\{\beta\}aa\{\beta\}}}{\{\beta\}t^{n+1}(\delta)\{\beta\}}$$

The proof of the second induction step goes similar. So we conclude that $\Gamma\vdash\{\alpha\}X^{x=axa+aa}\{\alpha\}$.
By analysing the form of derivations $Th(\Gamma)\vdash_H\{\alpha\}X^{x=axa+aa}\{\alpha\}$ we will show that the existence of such a derivation implies $\Gamma\vdash\{\alpha\}aa\{\alpha\}$ and $\Gamma\vdash\{\alpha\}aaa\{\alpha\}$. This we can show to be contradictory by means of a countermodel. Therefore $Th(\Gamma)\nvdash_H\{\alpha\}X^{x=axa+aa}\{\alpha\}$, despite $\Gamma\vdash\{\alpha\}X^{x=axa+aa}\{\alpha\}$.
Assume now that there is a derivation of $Th(\Gamma)\vdash_H\{\alpha\}X^{x=axa+aa}\{\alpha\}$. The rules of Hoare logic dictate its form: if we push the applications of the consequence rule upwards as far as possible, there are two possible proofs, of which we show one:

$$\dfrac{\dfrac{Th(\Gamma)}{\alpha\to\beta}\quad\dfrac{\dfrac{\dfrac{\dfrac{Th(\Gamma)}{\{\beta\}a\{\beta''\}}\quad\dfrac{\dfrac{Th(\Gamma)}{\beta''\to\beta}\quad[\{\beta\}x\{\gamma\}]\quad\dfrac{Th(\Gamma)}{\gamma\to\gamma''}}{\{\beta''\}x\{\gamma''\}}}{\{\beta\}ax\{\gamma''\}}\quad\dfrac{Th(\Gamma)}{\{\gamma''\}a\{\gamma\}}}{\{\beta\}axa\{\gamma\}}\quad\dfrac{Th(\Gamma)}{\{\beta\}aa\{\gamma\}}}{\{\beta\}axa+aa\{\gamma\}}}{\{\beta\}X^{x=axa+aa}\{\gamma\}}\quad\dfrac{Th(\Gamma)}{\gamma\to\alpha}}{\{\alpha\}X^{x=axa+aa}\{\alpha\}}$$

The other proof is characterized by the following subproof of $\{\beta\}axa\{\gamma\}$.

$$\dfrac{\dfrac{Th(\Gamma)}{\{\beta\}a\{\beta''\}}\quad\dfrac{\dfrac{\dfrac{Th(\Gamma)}{\beta''\to\beta}\quad[\{\beta\}x\{\gamma\}]\quad\dfrac{Th(\Gamma)}{\gamma\to\gamma''}}{\{\beta''\}x\{\gamma''\}}\quad\dfrac{Th(\Gamma)}{\{\gamma''\}a\{\gamma\}}}{\{\beta''\}xa\{\gamma\}}}{\{\beta\}axa\{\gamma\}}$$

Rearranging some of the information in these derivations we obtain $Th(\Gamma)\vdash_H\{\alpha\}aa\{\alpha\}$ and $Th(\Gamma)\vdash_H\{\alpha\}aaa\{\alpha\}$ as follows:

$$\dfrac{\dfrac{Th(\Gamma)}{\alpha\to\beta}\quad\dfrac{Th(\Gamma)}{\{\beta\}aa\{\gamma\}}\quad\dfrac{Th(\Gamma)}{\gamma\to\alpha}}{\{\alpha\}aa\{\alpha\}}\qquad\dfrac{\dfrac{Th(\Gamma)}{\alpha\to\beta}\quad\dfrac{\dfrac{\dfrac{Th(\Gamma)}{\{\beta\}aa\{\gamma\}}\quad\dfrac{Th(\Gamma)}{\gamma\to\gamma''}}{\{\beta\}aa\{\gamma'\}}\quad\dfrac{Th(\Gamma)}{\{\gamma'\}a\{\gamma\}}}{\{\beta\}aaa\{\gamma\}}\quad\dfrac{Th(\Gamma)}{\gamma\to\alpha}}{\{\alpha\}aaa\{\alpha\}}$$

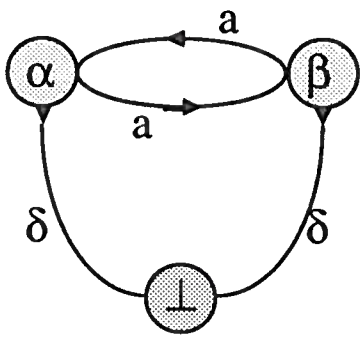
We will present a model M and assertions α such that

$M\models\Gamma$ and $M\models\{\alpha\}X^{x=axa+aa}\{\alpha\}$ and $M\models\{\alpha\}aa\{\alpha\}$

but

not: $M\models\{\alpha\}aaa\{\alpha\}$.

Consider a model M, consisting of three states, α, β and $\perp$. Define $eff_a(\alpha)=\beta$ and $eff_a(\beta)=\alpha$. Let α (β) be the assertion that is true only for state α (β):



Clearly $w(a,\alpha)=\beta$ and $w(a,\beta)=\alpha$. Hence $M\models\Gamma$ and also $w(t^n(\delta),\alpha)=w((aa)^n+...+aa,\alpha)=\alpha$.
Therefore $M\models\{\alpha\}X^{x=axa+aa}\{\alpha\}$ and $M\models\{\alpha\}aa\{\alpha\}$. However, $M\models\{\alpha\}aaa\{\alpha\}$ implies $M\models\forall s\neq\perp(\alpha(s)\rightarrow\beta(s))$, and so $M\models$**false**. Hence $M\not\models\{\alpha\}aaa\{\alpha\}$.

□

In the next section we will apply these results to some programming languages.
We end this section with a short discussion of semantics for regular processes induced by the Hoare logic $\vdash_H$ (cf. [M] and [BK1]. The discussion will hold verbatim (replace $\vdash_H$ by $\vdash_G$ and BPT_{linrec} by BPT_{rec}) for the other Hoare logic $\vdash_G$ with respect to the more general contextfree processes.

Consider process equivalence $p\leftrightarrows_H q$ defined by $Th(\varnothing)\vdash_H\{\alpha\}p\{\beta\}\Leftrightarrow Th(\varnothing)\vdash_H\{\alpha\}q\{\beta\}$ for all assertions α, and β.
The following argument shows that $p\leftrightarrows_H q$ is equivalent to $\models p=q$ and hence by the completeness theorem for process algebra $p\leftrightarrows_H q$ is also equivalent to $BPT_{linrec}(A,\Psi)\vdash p=q$. This means that the semantics induced by the Hoare logic $\vdash_H$ of partial correctness is axiomatized by the axioms $BPT_{linrec}(A,\Psi)$ of process algebra.

$p\leftrightarrows_H q \Leftrightarrow Th(\varnothing)\vdash_H\{\alpha\}p\{\beta\}\Leftrightarrow Th(\varnothing)\vdash_H\{\alpha\}q\{\beta\}$ for all assertions α, and β (by definition)

$\Leftrightarrow Th(\varnothing)\vdash\{\alpha\}p\{\beta\}\Leftrightarrow Th(\varnothing)\vdash\{\alpha\}q\{\beta\}$ for all assertions α, and β ((6.11)+$Th(\varnothing)=Th(Th(\varnothing))$

$\Leftrightarrow\vdash\{\alpha\}p\{\beta\}\Leftrightarrow\vdash\{\alpha\}q\{\beta\}$ for all assertions α, and β

$\Leftrightarrow^*\vdash\forall s,s'\neq\perp(p(s,s')\leftrightarrow q(s,s'))$ (see explanantion)

$\Leftrightarrow\models p=q$ (definition (2.2.4))

$\Leftrightarrow BPT_{linrec}(A,\Psi)\vdash p=q$ (completeness theorem for process algebra (2.2.5))

($\Rightarrow^*$) is the non-trivial step. Fix $s_0\neq\perp$. Consider the assertions $\alpha(s):=$**true** and $\beta(s)=q(s_0,s)$.
Clearly $\vdash\{\textbf{true}\}p\{q(s_0,s)\}\rightarrow\forall s'\neq\perp(p(s,s')\rightarrow q(s_0,s'))$.
Then from $\vdash\{\textbf{true}\}p\{p(s_0,s)\}\Leftrightarrow\vdash\{\textbf{true}\}q\{p(s_0,s)\}$ we derive that $\vdash\forall s'\neq\perp(p(s_0,s')\rightarrow q(s_0,s'))$ for this fixed $s_0\neq\perp$.
Hence $\vdash\forall s,s'\neq\perp(p(s,s')\rightarrow q(s,s'))$. Similarly one proves $\vdash\forall s,s'\neq\perp(q(s,s')\rightarrow p(s,s'))$.

□

7. HOARE LOGICS FOR (NON-)DETERMINISTIC PROGRAMS

Up until now we were concerned with languages for processes involving the constructs: atomic actions, atomic tests, sequential composition, alternative composition, (linear) recursive specifications, if then else, while statements.

Traditionally Hoare logic (cf. [A1], [dB] and [TZ]) is a theory about the partial correctness of programs instead of processes. A programming language usually is a process language which atomic actions are assignments x:=t, where x is a program variable and t is a term in the (finitary, many-sorted) first order predicate logic based on some given signature Σ. The variables that occur in assertions in traditional Hoare logic are program or assertion variables in contrast to our global "state"-variables.

It is not difficult to see that a model in the traditional theory (for each sort a domain etc. (cf. [A1], [TZ])) can be transformed into a state semantics model by taking the valuations of values in the respective sorts to the variables as states. Let us call this a valuation model. The present approach to the old models is useful: the explicit use of effect functions prevents us to step in a well-known pitfall: in imperative programming languages the main atomic action is the *assignment* x:=t. In the present setting the Hoare axiom for assignment becomes: $\{\alpha(\mathrm{eff}_{x:=t}(s))\}x:=t\{\alpha(s)\}$, where, if s is such a valuation

$$\mathrm{eff}_{x:=t}(s)=y \mapsto \begin{cases} s(y) & \text{if } y\neq x \\ t[x_1/s(x_1),\ldots,x_n/s(x_n)] & \text{if } y=x \end{cases} .$$

How does our theory apply to imperative programming languages? Suppose we have some signature Σ as in [dB] or [TZ]. We now consider the programming language that is the process language with assignments x:=t as atomic actions and equations x=t as atomic tests, where t is a open term over the signature Σ and x some programming variable of the appropriate sort.

The completeness theorems ((5.5), (6.5) and (6.12)) for processes provide us with completeness theorems for possible Hoare logics of partial correctness of this programming language.

As an example we consider a non-deterministic variant of the deterministic programming language in the book [TZ] by Tucker and Zucker. Features of this language are among others: arrays, and array assignments.

7.1. STRONG COMPLETENESS THEOREM for a Hoare logic with Scott's induction rule based on a programming language (compare with the (weak) completeness theorem in (3.7) of [TZ]).

Let Σ be a signature like in [TZ] defining sorts (including sorts for natural numbers and booleans) and functions (relations are functions into the sort of booleans). Let the programming language PL_Σ be the process language $BPT_{linrec}(A_\Sigma,\Psi_\Sigma)$ constructed from:

- program variables of sort $i\in\Sigma$ are either variables over i (notation v^i) or variables over arrays of i (notation a^i)
- program terms of sort i (notation t^i) are given by the grammar
  $t^i ::= v^i \mid \langle a^i,n\rangle [v^{\mathbb{N}}] \mid F^i(t^{i_1}_1,\ldots t^{i_n}_n)$.
- the set A_Σ containing atomic actions: all assignments of forms $v^i:=t^i$ and $a^i[t^{\mathbb{N}}]:=t^i$.
- the set Ψ_Σ containing atomic tests: all formulas of the finitary predicate language based on the signature Σ.

Then for the infinitary assertion logic and Hoare logic based on Pl_Σ as in sections (3) and (6) it holds that:

(i) (soundness) If Γ is a set of partial correctness formulas and/or implications, then

$\Gamma \vdash_H \{\alpha\}p\{\beta\} \Rightarrow \Gamma \models \{\alpha\}p\{\beta\}$.

(ii) (adequacy) If Γ is a countable set of formulas in the assertion language, then $\Gamma \models \{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma) \vdash_H \{\alpha\}p\{\beta\}$.

PROOF.
Apply (6.12). □

In a number of aspects this result differs from from the completeness theorem for the Hoare Logic of the language $Proglang_{ra}(\Sigma)$ in section (3.7) of the Tucker and Zucker's book [TZ], in short:

(i) there are some changes in notation
(ii) the programming language: we have added nondeterminism and in order to arrive at a Hoare calculus with Scott's induction rule restricted recursion to linear recursion.
(iii) the assertion logic: we have global state variables instead of program variables and assertion variables.
(iv) our Hoare logic doesn't have invariance axioms, substitution rules and conjunction rules.
(v) the semantics for the programming language
(vi) we prove strong completeness, instead of relative completeness with respect to a certain standard model.
(vii) in the proof we don't have the problems of expressibility for the completeness theorem.
(viii) the present theory does not treat total correctness.

More detailed commentary:

(ad i) The role of the error and error state in [TZ] seems to be similar to the interplay of δ and $\perp$ in our case.

(ad ii) This causes that our theorem (7.1) does not immediately compare with the theorem in [TZ]. However, we can rebuild the theory developed in this paper in order to treat prove (7.1) for the language without non-deterministic choice but including the constructs **if-then-else** and **while-do-**. This involves a reconsideration of the proofs of the completeness theorems in sections 5 and 6. It is not difficult to check that the proofs can be adapted. Moreover, since without non-deterministic choice minimal solutions of non-linear equations are also minimal solutions of related linear equations (in case of one variable: replace in the specification $x=t(x)$ each subterm of the form $p(x)q$ by $p(x)$, where $p(x)$ does contain x, and q may contain x) the restriction on Scott's induction rule is no longer relevant!

(ad iii) It requires only a different reading to see traditional Hoare clauses as Hoare clauses for a suitable process language together with an infinitary assertion logic. Traditional Hoare clauses for the language of theorem (7.1) are of the form $\{\phi\}P\{\psi\}$, where P is a program and ϕ, ψ are formulas of the finitary first order predicate calculus based the signature Σ of the programming language. Observe that such formulas contain variables that also may occur in the program, i.e., variables ranging over the sorts in the signature of the programming language. We interpret such formulas ψ as tests $<\psi>(s)$, which are unary predicates over the states. Note that this is in accordance with the interpretation of assertions in [dB] and [TZ]. Although we do not make the explicit distinction of [TZ] (cf. (1.3.10)) between assertion variables and programming variables, implicitly it is present. Our interpretation of $\{\phi\}P\{\psi\}$ in a valuation model M is $M \models \forall s \neq \perp(\phi(s) \rightarrow \forall s \neq \perp(P(s,s') \rightarrow \psi(s')))$ where s and s' range over valuations. The programming variables in $\{\phi\}P\{\psi\}$ are the variables that occur on the left in an assignment in the program P: those variables are the only variables which value can be changed by P.

(ad iv) The reason why we have no employ in our Hoare calculi for the invariance axiom and the substitution rules in contrast to for instance [dB] and [TZ] should be better understood: regarding the latter this seems to stem from our use

of state variables (cf. our previous remark ad (iii)). Note that the status of the extra axioms and rules is unsettled also in the literature: compare [TZ] (p. 107) with [dB]. It seems that we circumvent the use of the full conjunction rule by the analysis of Hoare logic proofs resulting in lemma (6.10).

(ad v) Our models differ, as we have global state variables, and not a model that consists of a domain for each sort of the signature. It is easy to see that the soundness theorem includes the valuation models. Given a traditional model, that provides a some set valued structure for each sort in Σ, the corresponding valuation model can be seen as a state semantics model: e.g. the definition of the effect function in case of a[t]:=t':

$$\text{effect}(\langle a^i,n\rangle[t^N]:=s^i,v)(\langle a^i,n\rangle) = \begin{cases} [0,n-1]\to A:k\mapsto \begin{cases} (s^i)^v & \text{if } k=(t^i)^v \\ a^i[k] & \text{otherwise} \end{cases} & \text{provided that } 0\le(t^i)^v\le n-1 \\ \bot & \text{otherwise} \end{cases}$$

Let effect($\langle a^i,n\rangle[t^N]:=s^i,v$) be the identity on the other (array-)variables.
Example (this is a test example in [dB], p. 109): Define v(a)(1)=2, v(a)(2)=2 Then effect(a[a[2]]:=1,v)(a)(1) = 1 and effect(a[a[2]]:=1,v)(a)(2) = 2.
It can be checked that the models now validate the following Hoare axiom (in simplified notation) for assignment for array variables:

$\{\alpha(\text{eff}_{a[t]:=t'}(v))\}a[t]:=t'\{\alpha(v)\}$, where v ranges over valuations.

By this construction, and the rewriting trick for traditional assertions as in (iii) above, plus the remark in (ii) we obtain the theorem (3.7) at page 126 of [TZ] as corollary because:

(ad vi) Strong completeness implies relative completeness. Given ones favourite model K (if traditional take the corresponding valuation model) define

$\Gamma(K):=\{\{\alpha\}p\{\beta\} \mid K\models\{\alpha\}p\{\beta\}$, for all α, β and closed process expressions $p\}$.

Since clearly

$\Gamma(K)\models\{\alpha\}p\{\beta\} \Rightarrow K\models\{\alpha\}p\{\beta\}$

and

$K\models\{\alpha\}p\{\beta\} \Rightarrow \{\alpha\}p\{\beta\}\in\Gamma(K) \Rightarrow \Gamma(K)\models\{\alpha\}p\{\beta\}$

we get

$\Gamma(K)\models\{\alpha\}p\{\beta\} \Leftrightarrow K\models\{\alpha\}p\{\beta\}$

and so by (6.11) and (3.6) (we have to be sure that $Th(\Gamma(K))$ is indeed countable, which is the case if the signature where the programming language is based on contains at most countable many symbols...)

$Th(\Gamma(K))\vdash_H\{\alpha\}p\{\beta\} \Leftrightarrow K\models\{\alpha\}p\{\beta\}$ for $p\in BPT_{linrec}(A,\Psi)$.

Similarly:

$Th(\Gamma(K))\vdash_G\{\alpha\}p\{\beta\} \Leftrightarrow K\models\{\alpha\}p\{\beta\}$ for $p\in BPT_{rec}(A,\Psi)$.

(ad vii) That we don't encounter the traditional difficulties with the expressibility of weakest preconditions in the assertion logic comes from our choice of an infinitary assertion language. The question of expressibility is just the question whether there exists finitary equivalents of weakest preconditions and is irrelevant for the question of completeness! One can imagine that finitary equivalents of weakest preconditions exist if the finitary part of the assertion logic has sufficient coding properties, corresponding to the coding properties of the natural numbers. See beside [TZ] also Harel's analysis in [Ha] and Zucker's appendix in [dB].

8. RELATIONS TO OTHER WORK

In this section we briefly compare our results and methods with literature on Hoare logic. Since the subject is relatively old and well established and literature is vast, we can only hope not to have made serious omissions with respect to the present work.

Section 2

We have taken the formalism of process algebra, i.e., atomic actions, sequential composition, non-deterministic choice and recursion to which we have added the feature of tests, including δ and ε. Tests, however, are one of the traditional features considered in the programming languages whose Hoare logics have been studied (cf. [dB]). The treatment of recursion is notationally a bit different, but essentially the same as in [dB].

The processes in $BPT_{linrec}(A,\Psi)$ are regular processes over $A \cup \Psi$, in contrast to processes that are context-free over $A \cup \Psi$ in $BPT_{rec}(A,\Psi)$ (cf. [HU] for the notions regular and context-free, [BBK] for the relation with process algebra).

The regular programs and processes are usually defined with help of the Kleene star operation (cf. the literature on dynamic logic, e.g. [Ha], algorithmic logic e.g. [MS]) and not with regular/linear recursion as in process algebra.

The process algebra approach with tests implies the well-known fact that conditionals and while statements are features of regular processes: the recursive specification of the while statement is linear/regular.

Dynamic logic has a variant that defines context free programs: Context-free Dynamic Logic (cf. [Ha]) which has full recursion in a style related to process algebra.

In the school of algorithmic logic such context-free extensions seem not to have been considered (cf. [MS]).

Section 3

A first full account of infinitary logic is given by Karp (cf. [Ka]). References to literature can be found in [Ke].

The use of infinitary languages (not to be confused with proof systems for finitary languages that make use of infinitary rules) in the field of semantics for programming languages dates back to Engeler (cf. [E]), who proves with help of an infinitary language that a (regular) program p terminates for all inputs in some model M if and only if for some appropriate infinitary formula ϕ the model M satisfies ϕ.

Back ([Bac1] and [Bac2]) has used infinitary languages to express weakest preconditions for regular programs and to prove soundness of their Hoare logic: "infinitary logic is the most appropriate formal system for expressing weakest preconditions of programs and reasoning" (quote from [Bac1]).

We have used our infinitary assertion logic for the same reason for which dynamic logic (cf. [Ha]) and algorithmic logic (cf. [MS]) have been introduced: to provide a general framework in which it is possible to express a wide variety of notions and concepts...(cf. [Ha]).

Pratt's dynamic logic can be expressed in infinitary assertion logic as shown by Meyer and Parikh (cf. [MP]): the language elements of infinitary assertion logic are of a more elementary character (one could make a comparison: <models, infinitary assertion logic, dynamic logic, Hoare logic> versus <machines, machine language, assembler, higher order language>). With help of our notation we can translate dynamic logic as follows:

$(\langle p \rangle \beta)(s)$:= if there exists a (proper) state reachable from s via p , which satisfies β (cf. [Ha])
:= $\exists s' \neq \perp (p(s,s') \wedge \beta(s'))$

$([p]\beta)(s)$:= every (proper) state reachable from s via p satisfies β (cf. [Ha])
:= $\forall s' \neq \perp (p(s,s') \rightarrow \beta(s'))$

We added proper, as in dynamic logic the error/failure state $\perp$ is not considered. By working in dynamic logic one is syntactically not aware of the underlying infinitary assertion logic, only semantically. This translation is very similar

to the translation of finite test dynamic logic in to the infinitary language $\mathcal{L}^{CK}_{\omega_1\omega}$ by Meyer and Parikh (cf. [MP]), who use the translation to show that the expressive power of some variant of Dynamic Logic is equal to $\mathcal{L}^{CK}_{\omega_1\omega}$.
In a similar way algorithmic logic of Salwicki and Mirkowska (cf. [MS]) can be expressed in the infinitary assertion language, as the essential difference of algorithmic logic and dynamic logic lies in the (infinite respectively finite) proof rules and axioms, not in the language.

Our semantics for the assertion logic is straightforward. The semantics of the above interpretation of dynamic logic in infinitary assertion logic is the usual semantics for dynamic logic (cf. [Ha]). Similar for the algorithmic logics (cf. [MS]). It should be noted that the infinitary assertion logic is simple and straightforward to axiomatize, and does not need a list of axioms concerning <p> and [p] as in dynamic logic or algorithmic logic: these axioms become derivable.

Section 4.
In [Bac1] Back gives an inductive definition of weakest preconditions for guarded commands, a class of regular programs. In contrast we give a general definition from which Back's definition follows as lemmas (4.5) and (4.3). In [Bac2] Back shows that non-deterministic assignments can be expressed in $\mathcal{L}_{\omega_1\omega_1}$, and not in $\mathcal{L}_{\omega_1\omega}$.

Section 5 and 6
Axiomatization: we strive for *strong completeness of Hoare logic*, and base our (therefore relatively simple) proof on the well-known strong completeness of the infinitary assertion logic (cf. [Ka]), a stronger goal then *weak completeness* of algorithmic logic. Dynamic logic (cf. [Pr] and [Ha]) seems to search only for *relative completeness* in the context-free case, and refers to algorithmic logic for *weak completeness*.
For clarity we recall the definition of these notions of completeness:

- strong completeness for Hoare logic:
  $\Gamma\vdash_{HoareLogic}\{\alpha\}p\{\beta\} \Rightarrow \Gamma\models\{\alpha\}p\{\beta\}$ together with $\Gamma\models\{\alpha\}p\{\beta\} \Rightarrow Th(\Gamma)\vdash_{HoareLogic}\{\alpha\}p\{\beta\}$
- weak completeness:
  $Th(\varnothing)\vdash_{HoareLogic}\{\alpha\}p\{\beta\} \Leftrightarrow \models\{\alpha\}p\{\beta\}$ (usually $Th(\varnothing)$ is not mentioned, but hidden as oracle rule, cf. [Ha])
- relative completeness
  $Th(M)\vdash_{HoareLogic}\{\alpha\}p\{\beta\} \Leftrightarrow M\models\{\alpha\}p\{\beta\}$, where Th(M) stands for the formulas true in M.

The Hoare logic for the recursion free processes is standard (cf. [A1&2]).
The infinitary induction rule is the rule the Polish school should have used if they had considered it (cf. [MS]).
Scott's induction rule is the usual one in the context of programs with procedures (cf. [dBS] and [dB]).
The Polish school of algorithmic logic proves weak completeness theorems for Hoare logic from which weak completeness theorems for Hoare logic of bounded non-deterministic, iterative programs, cf. [MS]. Harel uses this method to prove weak completeness for the Hoare logic corresponding to this regular programming language (cf. [Ha]).
Dynamic logic itself provides only relative (often arithmetical) completeness results for regular and context-free programs in dynamic logic (cf. [Ha]).
De Bakker, Tucker & Zucker treat relative completeness for various deterministic programming languages ([dB], [TZ]). For the non-deterministic case de Bakker refers to [Ha].
In [A2] Apt does not treat full recursion for Hoare logic of non-deterministic programs. He treats recursion in the form of while statements only.
In [Bac1] Back uses infinitary logic to show that $\Gamma\models\{\alpha\}p\{\beta\}$ if and only if $\Gamma\vdash_{InfLog}\alpha\rightarrow w(p,\beta)$ for any countable set Γ of sentences in the infinitary logic $\mathcal{L}_{\omega_1\omega}$, and he proves soundness for the Hoare logic of total correctness of guarded command programs.
In [Po] Ponse proves relative completeness for the Hoare logic of a process language that allows for linear, guarded recursive specifications. His assertion language is finitary first order predicate logic, build with predicate symbols

corresponding to the subsets of the underlying set of the model one proves relative completeness for. His Hoare logic resembles the present proof system $\vdash_H$ but for the assertion language.
The incompleteness of Scott's induction rule with respect to $BPT_{rec}(A,\Psi)$ seems not to have been observed before. To our knowledge it is noticed in the context of relative completeness for the first time by Ponse (cf. [Po]).

9. ACKNOWLEDGEMENTS.

We acknowledge the enthusiasm, patience and help of the collegues of CWI group AP and the participants of the thursday morning PAM-sessions. Notably we would like to thank Jan Bergstra, Rob van Glabbeek and Jan Friso Groote.

10. REFERENCES

[A1] K.R. APT, *Ten years of Hoare's Logic: A survey - part I*, ACM-TOPLAS 3(4), 1981, pp. 431-483.

[A2] K.R. APT, *Ten years of Hoare's Logic: A survey - part II: nondeterminism*, Theoretic. Comp. Sci. 28, 1984, pp. 83-109.

[Bac1] R.J.R. BACK, *Correctness preserving program refinements: proof theory and applications*, Mathematical Centre Tracts 131, Centrum voor Wiskunde en Informatica, Amsterdam 1980.

[Bac2] R.J.R. BACK, *Total correctness, nondeterminism and infinitary logic*, Acta Informatica 15, 1981, pp. 233-249.

[BB1] J.C.M. BAETEN and J.A. BERGSTRA, *Global renaming operators over concrete process algebra*, Inf. & Comp. 78 (3), 1988, pp. 205-245.

[BB2] J.C.M. BAETEN and J.A. BERGSTRA, *Recursive process definitions with the state operator*, report CS-R8920, Centrum voor Wiskunde en Informatica, Amsterdam, 1989.

[BBK] J.C.M. BAETEN, J.A. BERGSTRA and J.W. KLOP, *Decidability of bisimulation equivalence for processes generating context-free languages*, in: *PARLE II* (eds. J.W. de Bakker, A.J. Nijman and P.C. Treleaven), Lecture Notes in Computer Science 259, Springer, New York, 1987, pp. 94-111.

[dB] J.W. DE BAKKER, *Mathematical theory of program correctness*, Prentice-Hall International London, 1980.

[dBS] J.W DE BAKKER and D.S. SCOTT, *A theory of programs*, IBM Seminar Vienna 1969, appeared in: *J.W. de Bakker, 25 Jaar Semantiek, Liber Amicorum* (eds. J.W. Klop, J.-J.Ch. Meyer and J.J.M.M. Rutten), CWI, Amsterdam, 1989.

[BK1] J.A. BERGSTRA and J.W. KLOP, *Proving program inclusion using Hoare's logic*, TCS 30, 1984, p.1-48.

[BK2] J.A. BERGSTRA and J.W. KLOP, *A convergence theorem in process algebra*, Report CS-R8733, Centre for Mathematics and Computer Science, CWI, Amsterdam.

[BK3] J.A. BERGSTRA and J.W. KLOP, *A complete inference system for regular processes*, in: *Logic Colloquium '86* (eds. F.R. Drake and J.K. Truss), North-Holland, Amsterdam, 1988, pp. 21-82.

[BK4] J.A. BERGSTRA and J.W. KLOP, *Process theory based on bisimulation semantics*, in: *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency* (eds. J.W. de Bakker, W.-P. de Roever and G. Rozenberg), Lecture Notes in Computer Science 354, Springer, New York, 1989, pp. 50-122.

[BT1] J.A. BERGSTRA and J.V. TUCKER, *Expressiveness and the completeness of Hoare's logic*, Journal of Computer and System Sciences 25(3), 1982, pp. 267-284.

[BT2] J.A. BERGSTRA and J.V. TUCKER, *Some natural structures which fail to possess a sound and decidable Hoare-like logic for their while-programs*, Theoretical Computer Science 17, 1982, pp. 303-315.

[C] S.A. COOK, *Soundness and completeness of an axiom system for program verification*, SIAM. J. Comp. 7, 1978, pp. 70-90; Corrigendum, 10, 1981, p. 612.

[vD] D. VAN DALEN, *Logic and structure*, Springer, New York, 1983.

[E] E. ENGELER, *Algorithmic properties of structures*, Math. Syst. Theory 1(3), 1967, pp. 183-195.

[Gl] R.J. VAN GLABBEEK, *Bounded nondeterminism and the approximation induction principle in process algebra*, in *Proceceedings STACS 87* (eds. F.J. Brandenburg, G. Vidal-Naquet, M. Wirsing), Lecture Notes in Computer Science 247, Springer, New York, 1987, pp. 336-347.

[Ha] D. HAREL, *First-Order Dynamic Logic*, Lecture Notes in Computer Science 68, Springer, New York, 1979.

[HU] J.E. HOPCROFT and J.D. ULLMAN, *Introduction to automata theory, languages and computation*, Addison-Wesley, Reading MA., 1979.

[Ja] N. JACOBSON, *Basic Algebra I*, W.H. Freeman and Company, San Francisco, 1974.

[Jo] P.T. JOHNSTONE, *Stone Spaces*, Cambridge University Press, Cambridge, 1982.

[Ka] C.R. KARP, *Languages with expressions of infinite length*, North-Holland, Amsterdam, 1964.

[Ke] H.J. KEISLER, *Model theory for infinitary logic*, North-Holland, Amsterdam, 1971.

[MR] M. MAKKAI and G.E. REYES, *First Order Categorical Logic*, Lecture Notes in Mathematics 611, Springer, New York, 1977.

[MA] E.G. MANES and M.A. ARBIB, *Algebraic Approaches to Program Semantics*, Springer, New York, 1986.

[M] A.R. Meyer, *Floyd-Hoare Logic defines semantics: preliminary version.*, in proceedings of the LICS 86, IEEE, Washington, p. 44-47.

[MP] A.R. MEYER and R. PARIKH, *Definability in Dynamic Logic*, J. of Comp. and Syst. Sc. 23, 1981, pp. 279-298.

[MS] G. MIRKOWSKA and A. SALWICKI, *Algorithmic Logic*, PWN, Warszawa, Reidel, Dordrecht, 1987.

[Po] A. PONSE, *Process expressions and Hoare's logic*, report CS-R8905, Centrum voor Wiskunde en Informatica, Amsterdam, 1989 (to appear in Information and Computation).

[Pr] V.R. PRATT, *Dynamic logic*, in *Foundations of Computer Science III, part 2*, (eds. J.W. de Bakker and J. van Leeuwen), Mathematical Centre Tracts 109, Centrum voor Wiskunde en Informatica, Amsterdam 1979, pp. 53-83.

[TZ] J.V. TUCKER and J.I. ZUCKER, *Program Correctness over abstract data types, with error-state semantics*, CWI Monographs 6, North-Holland, Amsterdam, 1988.