stichting
mathematisch
centrum

AFDELING INFORMATICA IW 194/82 MAART
(DEPARTMENT OF COMPUTER SCIENCE)

J.A. BERGSTRA & J.W. KLOP

FORMAL PROOF SYSTEMS FOR PROGRAM EQUIVALENCE

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics subject classification: 03D45, 03D80, 68B15,
03D35, 03D75, 68B10

1982 CR Categories: 4.34, 5.24

Formal proof systems for program equivalence*)

by

J.A. Bergstra**) & J.W. Klop

ABSTRACT

We explore conservative refinements of specifications. These form an appropriate framework for a proof theory for program equivalence that is based on a logic for partial program correctness.

We propose two formalized proof methods for program equivalence (inclusion). Both are sound w.r.t. the most general semantics of first order specifications. In spite of being incomplete the methods cover many natural examples.

KEY WORDS & PHRASES: *data type specification, program correctness, conservative refinement, program inclusion, program equivalence, prototype proof, logical completion*

---

*) This report will be submitted for publication elsewhere.

This report is a revised and abridged version of the report IW 176/81 'Proving program inclusion using Hoare's logic'. Several topics, discussed in IW 176/81, are not touched here; also some changes in the order of presentation have been made and alternative arguments have been used in some proofs.

**) Department of Computer Science, University of Leiden, Wassenaarseweg 80, 2300 RA Leiden, The Netherlands

0. INTRODUCTION

This paper aims at a detailed study of program equivalence, seen from the point of view of Hoare's logic for program correctness. Because program inclusion is just halfway program equivalence we can safely restrict our attention to program inclusion. This moreover has the advantage of connecting closely to the theory of programming using stepwise refinements as described in BACK [2].

Our work can be seen as belonging to the subject of axiomatic semantics for programs. Its novelty lies in the precise mathematical analysis of the situation, in addition to a rather strict adherence to first order proof systems and first order semantics for data type specifications.

Deriving program equivalence from program correctness properties is not a new idea, of course. It occurs in compiler correctness proofs, for instance HEMERIK [12], and RUSSELL [16], as well as in the general theory of program correctness as in HAREL, PNUELI & STAVI [11], COUSINEAU - ENJALBERT [9].

Because of our interest in a proper theoretical analysis, we try to minimize the semantical problems by working with <u>while</u>- programs only; this by no means trivializes the problem. We expect that the present theory can be generalized to more powerful programming concepts, although not without some effort.

It appears to us to be a worth-while but nontrivial project to relate our proof systems to the methods of algebraic semantics, as explained e.g. in GUESSARIAN [10].

One might expect a close relationship between the present work and MEYER-HALPERN [14], which also describes program equivalence from the point of view of first order logic. It is an important difference however that their paper focuses on semantics, whereas our main interest is in proof systems.

In the sequel of this introduction an intuitive account is given of the key definitions that underly the paper.

<u>Intuition</u>. Suppose that for $S_1, S_2 \in WP(\Sigma)$ we have

(1) $\quad Alg(\Sigma,T) \models S_1 \sqsubseteq S_2$ $\qquad$ (*semantical inclusion*)

and that we wish to prove this fact. Now obviously, (1) implies

(2) $\text{Alg}(\Sigma,T) \models \{p\}S_2\{q\} \Rightarrow \text{Alg}(\Sigma,T) \models \{p\}S_1\{q\}$, for all $p,q \in L(\Sigma)$.

However, there is no reason to expect that the reverse implication (2) ⇒ (1) will hold, since (2) states only roughly that $S_1 \sqsubseteq S_2$, where 'roughly' refers to the limited expressive power of $L(\Sigma)$. (In fact, one can show that indeed (2) $\not\Rightarrow$ (1).) Now consider

(3) $\forall(\Sigma',T') \geq (\Sigma,T) \;\; \forall p,q \in L(\Sigma')$
$$\text{Alg}(\Sigma',T') \models \{p\}S_2\{q\} \Rightarrow \text{Alg}(\Sigma',T') \models \{p\}S_1\{q\}.$$

Clearly (1) ⇒ (3) ⇒ (2). (For (1) ⇒ (3), note that if $(\Sigma',T') \geq (\Sigma,T)$, then the reducts of (Σ',T')-algebras to Σ form a subset of $\text{Alg}(\Sigma,T)$; hence $\text{Alg}(\Sigma,T) \models S_1 \sqsubseteq S_2 \Rightarrow \text{Alg}(\Sigma',T') \models S_1 \sqsubseteq S_2$.)
In fact we will restrict our attention to a subclass of all refinements (≥) of (Σ,T), namely to the *conservative* refinements (⊵) of (Σ,T), for reasons which will be clear later. So consider

(4) $\forall(\Sigma',T') \trianglerighteq (\Sigma,T) \;\; \forall p,q \in L(\Sigma')$
$$\text{Alg}(\Sigma',T') \models \{p\}S_2\{q\} \Rightarrow \text{Alg}(\Sigma',T') \models \{p\}S_1\{q\}.$$

Now we have (1) ⇒ (3) ⇒ (4) ⇒ (2); and it can be shown that (4) ⇒ (1). The conclusion is that one can treat the 'semantical' inclusion (1) by considering only first order properties of S_1, S_2 (i.e. asserted programs $\{p\}S_i\{q\}$, $i = 1,2$), provided one is willing to consider not only (Σ,T), but all its (conservative) refinements.

This observation prepares the way for an approach via Hoare's logic of proving asserted programs. First of all, define

(5) $S_1 \sqsubseteq_{HL(\Sigma,T)} S_2$ iff $\forall p,q \in L(\Sigma)$
$$HL(\Sigma,T) \vdash \{p\}S_2\{q\} \Rightarrow HL(\Sigma,T) \vdash \{p\}S_1\{q\}$$
(*prooftheoretical inclusion*)

and consider

(6) $\forall(\Sigma',T') \trianglerighteq (\Sigma,T) \;\; S_1 \sqsubseteq_{HL(\Sigma',T')} S_2$ (*derivable inclusion*)

the prooftheoretical analogue of (4). Indeed, it will turn out that this 'derivable' inclusion, written as $HL(\Sigma,T) \vdash S_1 \sqsubseteq S_2$, implies the semantical inclusion (1). This is our first "proof system" for proving semantical inclusion; we will prove that (6), as a relation of S_1, S_2, is semi-decidable in T.

Of course the proof system given by (6) is sound, i.e. (6) ⇒ (1); otherwise it did not deserve the name. Some simple program inclusions that are in its scope, are program equivalences like 'loop-unwinding', and the kind of program equivalences considered in MANNA [13]. This proof system is not yet complete, however. In order to prove semantical inclusion (1), it is sufficient that:

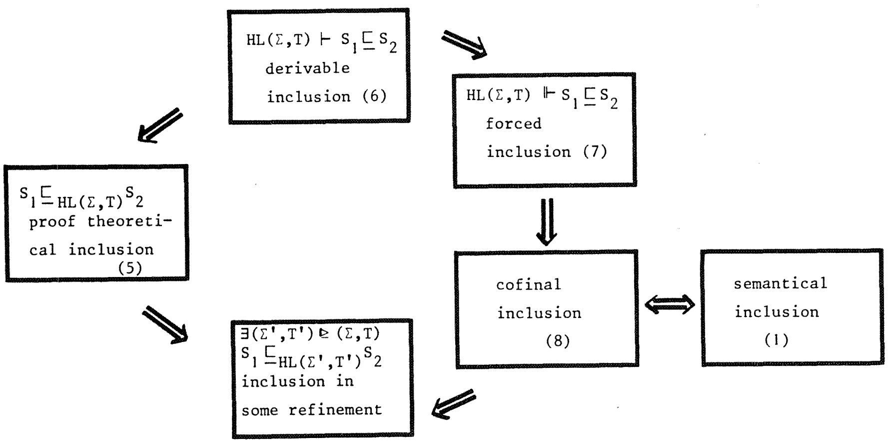
(7) $\exists(\Sigma',T') \trianglerighteq (\Sigma,T) \;\; \forall(\Sigma'',T'') \trianglerighteq (\Sigma',T') \;\; S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2$.

(Notation: $HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_2$, in words: *forced inclusion*.)

The proof system embodied by (7) is stronger than that of derivable inclusion (6), and we will give an example of program inclusion which requires the extra strength of this last proof system.
Still, (7) is not 'complete'. One can prove, however, that the following 'cofinal' inclusion is equivalent to semantical inclusion:

(8) $\forall(\Sigma',T') \trianglerighteq (\Sigma,T) \;\; \exists(\Sigma'',T'') \trianglerighteq (\Sigma',T') \;\; S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2$.

One could suspect that there is a multitude of such relations obtained by repeated alternating quantification $\forall\exists\forall$... from the basic relation $\sqsubseteq_{HL(\Sigma,T)}$ (prooftheoretical inclusion). It is a pleasant surprise, suggesting the naturalness of the notions involved, that this possible hierarchy does in fact not exist, and that one has no more relations than in the following diagram:



1. PRELIMINARIES ABOUT PROGRAMS AND LOGIC

The notions of first order language, derivability ($\vdash$) and satisfiability ($\models$) are supposed known.

In this paper we will exclusively deal with <u>while</u>-programs. For a signature Σ the set $WP(\Sigma)$ of <u>while</u>-programs over Σ is defined inductively as follows:

$$S ::= x := t \mid S_1;S_2 \mid \underline{\text{if}}\ b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\ \underline{\text{fi}} \mid \underline{\text{while}}\ b\ \underline{\text{do}}\ S\ \underline{\text{od}},$$

where $t \in Ter(\Sigma)$, the set of terms over the signature Σ, b is a boolean (i.e. quantifier free) assertion $\in L(\Sigma)$, the first order language determined by Σ.

A *specification* is a pair (Σ,T) where $T \subseteq L(\Sigma)$; the semantics of a specification is just the collection $Alg(\Sigma,T)$ of Σ-structures $\mathcal{A}$ such that $\mathcal{A} \models T$. We write $Alg(\Sigma)$ for $Alg(\Sigma,\emptyset)$.

$\mathcal{A},\mathcal{B} \in Alg(\Sigma,T)$ will be written as $\mathcal{A} = (A,\ldots)$, $\mathcal{B} = (B,\ldots)$ where A, B are the underlying sets.

For $\mathcal{A} \in Alg(\Sigma)$ and $S \in WP(\Sigma)$ with variables $x_1,\ldots,x_k$ the meaning of S in $\mathcal{A}$ is a partial function $M_{\mathcal{A}}(S)\colon A^k \to A^k$. $M_{\mathcal{A}}(S)$ can be defined using conventional methods of operational or denotational semantics.

We write $S(\vec{a}) = \vec{b}$ for $M_{\mathcal{A}}(S)(\vec{a}) = \vec{b}$; if $S(\vec{a}) = \vec{b}$ for some $\vec{b}$ we write $S(\vec{a})\downarrow$ (otherwise $S(\vec{a})\uparrow$).

Important is the following

1.1. <u>COMPUTATION LEMMA</u>. *Let* $\vec{x} = x_1,\ldots,x_k$ *and* $\vec{y} = y_1,\ldots,y_k$. *Let* $S = S(\vec{x}) \in WP(\Sigma)$ (i.e. S *contains precisely the variables* $\vec{x}$).
Then for all $n \in \mathbb{N}$ *there is a quantifier free assertion* $Comp_{S,n}(\vec{x}) = \vec{y}$ *in* $L(\Sigma)$ *such that for every* $\mathcal{A} \in Alg(\Sigma)$ *and all* $\vec{a},\vec{b} \in A$:

$$\mathcal{A} \models Comp_{S,n}(\underline{\vec{a}}) = \underline{\vec{b}} \iff |S(\vec{a})| \le n \text{ and } S(\vec{a}) = \vec{b}.$$

Here $\underline{\vec{a}}$, $\underline{\vec{b}}$ *are constant symbols denoting* $\vec{a}$, $\vec{b}$ *and* $|S(\vec{a})|$ *denotes the length of the* *computation of* S *on* $\vec{a}$.

1.2. Preliminaries on Hoare's logic.

Let $p,q \in L(\Sigma)$ and $S \in WP(\Sigma)$. Then the syntactic object $\{p\}S\{q\}$ is called an '*asserted program*'. For $A \in Alg(\Sigma)$, we define:

$$A \models \{p\}S\{q\} \text{ iff } \forall \vec{a},\vec{b} \in A\colon S(\vec{a})\downarrow \text{ and } S(\vec{a}) = \vec{b} \Rightarrow (A \models p(\underline{\vec{a}}) \rightarrow q(\underline{\vec{b}})).$$

Furthermore we define

$$Alg(\Sigma,T) \models \{p\}S\{q\} \Longleftrightarrow \forall A \in Alg(\Sigma,T)\ A \models \{p\}S\{q\}.$$

Hoare's logic w.r.t. (Σ,T) is a well-known proof system designed to prove facts like $Alg(\Sigma,T) \models \{p\}S\{q\}$. We will call this proof system $HL(\Sigma,T)$; it provides one axiom (*assignment* axiom) and four rules:

(1) *Assignment axiom scheme:* $\{p[t/x]\}\ \ x:=t\ \{p\}$

(2) *Composition rule:* $\dfrac{\{p\}S_1\{r\} \quad \{r\}S_2\{q\}}{\{p\}S_1;S_2\{q\}}$

(3) *Conditional rule:* $\dfrac{\{p\wedge b\}S_1\{q\} \quad \{p\wedge\neg b\}S_2\{q\}}{\{p\}\ \underline{\text{if}}\ b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\ \underline{\text{fi}}\ \{q\}}$

(4) *Iteration rule:* $\dfrac{\{p\wedge b\}\ S\ \{p\}}{\{p\}\ \underline{\text{while}}\ b\ \underline{\text{do}}\ S\ \underline{\text{od}}\ \{p\wedge\neg b\}}$

(5) *Consequence rule:* $\dfrac{p\rightarrow p_1 \quad \{p_1\}S\{q_1\} \quad q_1\rightarrow q}{\{p\}S\{q\}}$

where $(\Sigma,T) \vdash p \rightarrow p_1$ and $(\Sigma,T) \vdash q_1 \rightarrow q$.

These rules constitute an inductive definition of a relation $HL(\Sigma,T) \vdash \{p\}S\{q\}$; we assume familiarity with this proof system.

$HL(\Sigma,T)$ is sound in the following sense: for all $p,q \in L(\Sigma)$ and $S \in WP(\Sigma)$: $HL(\Sigma,T) \vdash \{p\}S\{q\} \Rightarrow Alg(\Sigma,T) \models \{p\}S\{q\}$.

1.2.1. DEFINITION. $HL(\Sigma,T)$ is *logically complete* iff for all $p,q \in L(\Sigma)$ and $S \in WP(\Sigma)$: $HL(\Sigma,T) \vdash \{p\}S\{q\} \Longleftrightarrow Alg(\Sigma,T) \models \{p\}S\{q\}$.

(In general, $HL(\Sigma,T)$ is not logically complete. The notion of logical completeness is studied in BERGSTRA-TUCKER [6].)

2. REFINEMENTS OF SPECIFICATIONS

In this section we will collect some facts concerning the notion of *refinement* and especially, *conservative* refinement. These notions will be of fundamental importance in the sequel. All the material in this section is standard in Mathematical Logic and can be found (e.g.) in SHOENFIELD [17] and MONK [15].

2.1. DEFINITION (refinements)
(i) If $\Sigma' \supseteq \Sigma$ and $\overline{T'} \supseteq \overline{T}$ we write $(\Sigma',T') \geq (\Sigma,T)$ and call (Σ',T') a *refinement* of (Σ,T). Here $\overline{T} = \{p \in L(\Sigma) \mid T \vdash p\}$.
We will always suppose that T, T' are consistent.
(ii) Let A be some algebra. Then Σ_A is the *signature of* A and T_A is the *theory of* A: $T_A = \{p \in L(\Sigma_A) \mid A \models p\}$.
Note that $A \models p \Longleftrightarrow Alg(\Sigma_A,T_A) \models p$.
(iii) Let (Σ,T) be a specification. Then T is *complete* if $\forall p \in L(\Sigma)$, $T \vdash p$ or $T \vdash \neg p$.

2.2. DEFINITION (conservative refinements)
Let $(\Sigma',T') \geq (\Sigma,T)$ be a refinement such that: $\forall p \in L(\Sigma)\ T' \vdash p \Leftrightarrow T \vdash p$. In other words, such that $T' \cap L(\Sigma) = T$. Then this refinement is called *conservative* over (Σ,T).
(So a conservative refinement does not yield more theorems in the 'original' language $L(\Sigma)$.)
Notation: $(\Sigma',T') \trianglerighteq (\Sigma,T)$.

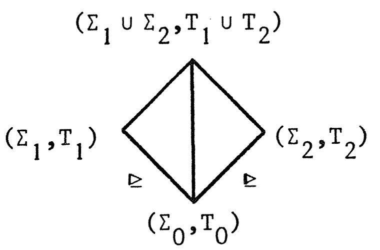
2.2.1. Note that if T is complete: $(\Sigma',T') \geq (\Sigma,T) \Rightarrow (\Sigma',T') \trianglerighteq (\Sigma,T)$.

2.3. DEFINITION. (Expansions and restrictions). Let $\Sigma' \supseteq \Sigma$.
(i) If (Σ',T') is a specification, then the *restriction* of (Σ',T') to the signature Σ is (Σ,T) where $T = T' \cap L(\Sigma)$.
(ii) If $A' \in Alg(\Sigma',T')$, then the *restriction* of A' to Σ is obtained by deleting all constants, functions, predicates in A' corresponding to symbols in $\Sigma' - \Sigma$. The resulting A is also called a *reduct* of A'; and A' is called an *expansion* of A. We will also write $A \leq A'$.

2.3.1. Note that if $A' \geq A$, then $(\Sigma_{A'},T_{A'}) \trianglerighteq (\Sigma_A,T_A)$.

In the sequel we will always deal with conservative refinements ($\trianglerighteq$). They have the pleasant property that two refinements $(\Sigma_i,T_i) \trianglerighteq (\Sigma,T)$ $(i = 1,2)$ can be joined to a refinement $(\Sigma_1 \cup \Sigma_2,\ T_1 \cup T_2) \trianglerighteq (\Sigma,T)$, provided the requirement $\Sigma_1 \cap \Sigma_2 = \Sigma$ is satisfied. This is a (strong) form of A. Robinson's Consistency Theorem (RCT).

2.4. ROBINSON'S CONSISTENCY THEOREM.



Let $(\Sigma_i,T_i) \trianglerighteq (\Sigma_0,T_0)$, $i = 1,2$, *such that* $\Sigma_1 \cap \Sigma_2 = \Sigma_0$. *Then*
(i) $T_1 \cup T_2$ *is consistent, and moreover*
(ii) $(\Sigma_1 \cup \Sigma_2,\ T_1 \cup T_2) \trianglerighteq (\Sigma_0,T_0)$.

PROOF. See Exercise 22.15 p.375 MONK [15] or BOOLOS - JEFFREY [8] p.244. □

We conclude this section with a useful criterion for conservativity.

2.5. DEFINITION. Let (Σ',T') be a refinement such that every $A \in Alg(\Sigma,T)$ can be expanded to an $A' \in Alg(\Sigma',T')$. Then this refinement is called *simple*.

2.6. PROPOSITION. (Criterion for conservativity). *Simple refinements are conservative.*

PROOF. Suppose (Σ',T') is a simple refinement of (Σ,T), i.e. $\forall A \in Alg(\Sigma,T)$ $\exists A' \in Alg(\Sigma',T')\ A' \geq A$. Let $T \nvdash p$ for some closed assertion p. Then by Gödel's Completeness Theorem, $A \nvDash p$ for some $A \in Alg(\Sigma,T)$. So there is an $A' \in Alg(\Sigma',T')$ such that $A' \geq A$. Hence $A' \vDash \neg p$; and reasoning backwards we have $T' \nvdash p$. □

3. PROGRAM INCLUSIONS

We will now introduce the various notions of inclusion ($\sqsubseteq$) between programs S_1, $S_2 \in WP(\Sigma)$ which we will study, and prove some important facts about them.

Let $S \in WP(\Sigma)$ and $A = (A,\ldots) \in Alg(\Sigma,T)$. Let S contain the variables $x_1,\ldots,x_n$ $(n \geq 1)$. Then $M_A(S): A^n \to A^n$ is the partial function defined in Section 1.

3.1. DEFINITION. Let S_1, $S_2 \in WP(\Sigma)$.

(i) *Semantical inclusion:*

$$Alg(\Sigma,T) \models S_1 \sqsubseteq S_2 \iff M_A(S_1) \subseteq M_A(S_2), \text{ for all } A \in Alg(\Sigma,T).$$

(ii) *Prooftheoretical inclusion:*

$$S_1 \sqsubseteq_{HL(\Sigma,T)} S_2 \text{ iff for all } p,q \in L(\Sigma)\colon HL(\Sigma,T) \vdash \{p\}S_2\{q\} \Rightarrow HL(\Sigma,T) \vdash \{p\}S_1\{q\}.$$

(iii) *Derivable inclusion:*

$$HL(\Sigma,T) \vdash S_1 \sqsubseteq S_2 \iff \forall(\Sigma',T') \trianglerighteq (\Sigma,T) \quad S_1 \sqsubseteq_{HL(\Sigma',T')} S_2.$$

(iv) *Forced inclusion:*

$$HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_2 \iff \exists(\Sigma',T') \trianglerighteq (\Sigma,T) \quad HL(\Sigma',T') \vdash S_1 \sqsubseteq S_2.$$

(v) *Cofinal inclusion:* the inclusion $S_1 \sqsubseteq S_2$ is *cofinal*, if

$$\forall(\Sigma',T') \trianglerighteq (\Sigma,T) \quad \exists(\Sigma'',T'') \trianglerighteq (\Sigma',T') \quad S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2.$$

3.2. REMARK. (i) Note the direction of the implication in 3.1 (ii). Intuitively: S_1 is less defined than S_2, so $\{p\}S_1\{q\}$ is more often trivially true.
(ii) The phrase 'derivable' in 3.1 (iii) and the choice of the notation ' $\vdash$ ' is justified by results in Section 5: it will be proved that derivable inclusion w.r.t. (Σ,T) is semi-decidable in T.
(iii) In all cases 3.1(i) - (v) there is the corresponding notion of equivalence, defined in the obvious way; e.g. for forced equivalence:

$$HL(\Sigma,T) \Vdash S_1 \equiv S_2 \iff HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_2 \text{ and } HL(\Sigma,T) \Vdash S_2 \sqsubseteq S_1.$$

It is clear that all inclusions ($\sqsubseteq$) defined above are partial orders and that all equivalences ($\equiv$) are equivalence relations, except for forced and cofinal inclusion resp. equivalence. For the last case, 'cofinal', we will prove in Section 5 that cofinal inclusion coincides with semantical inclusion, hence cofinal inclusion is indeed transitive.

3.3. PROPOSITION. *Forced inclusion is transitive. (Hence it is a partial order and forced equivalence is an equivalence relation.)*

PROOF. Let $S_1,S_2,S_3 \in WP(\Sigma)$, $HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_2$ and $HL(\Sigma,T) \Vdash S_2 \sqsubseteq S_3$. Then for $i = 1,2$:

$$\exists(\Sigma_i',T_i') \trianglerighteq (\Sigma,T) \quad \forall(\Sigma_i'',T_i'') \trianglerighteq (\Sigma_i',T_i') \quad S_i \sqsubseteq_{HL(\Sigma_i'',T_i'')} S_{i+1}.$$

Now consider such (Σ_i',T_i'), $i = 1,2$. We may suppose that $\Sigma_1' \cap \Sigma_2' = \Sigma$.
Now by Robinson's Consistency Theorem 2.4,

$$(\Sigma^*,T^*) = (\Sigma_1' \cup \Sigma_2',\ T_1' \cup T_2') \trianglerighteq (\Sigma,T).$$

Evidently, $HL(\Sigma^*,T^*) \vdash S_1 \sqsubseteq S_2$ and $HL(\Sigma^*,T^*) \vdash S_2 \sqsubseteq S_3$.
By transitivity of derivable inclusion, therefore $HL(\Sigma^*,T^*) \vdash S_1 \sqsubseteq S_3$. Hence $HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_3$. □

The main result of this section consists in establishing the various logical interrelationships between the previously defined notions of inclusion (and equivalence), as they are displayed in the diagram in the Introduction. There are only three nontrivial cases and two of them are dealt with in the following proposition.

3.4. PROPOSITION. (i) *Forced inclusion implies cofinal inclusion.*

(ii) *Semantical inclusion implies cofinal inclusion.*
(See Proposition (5.1) for the other direction.)

PROOF. (i) Suppose $HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_2$, i.e.:

$$(1) \qquad \exists(\Sigma',T') \trianglerighteq (\Sigma,T) \quad \forall(\Sigma'',T'') \trianglerighteq (\Sigma',T') \quad S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2$$

To prove:

$$(2) \qquad \forall(\Sigma_1',T_1') \trianglerighteq (\Sigma,T) \quad \exists(\Sigma_1'',T_1'') \trianglerighteq (\Sigma_1',T_1') \quad S_1 \sqsubseteq_{HL(\Sigma_1'',T_1'')} S_2$$

Take (Σ',T') as in (1), and consider a (Σ_1',T_1') as in (2).
We may assume that $\Sigma' \cap \Sigma_1' = \Sigma$. Then take (Σ_1'',T_1'') in (2) as the union of (Σ_1',T_1') and (Σ',T'); by RCT 2.4 this is possible.
(ii) To prove: $Alg(\Sigma,T) \models S_1 \sqsubseteq S_2 \Rightarrow$

$$\forall(\Sigma',T') \trianglerighteq (\Sigma,T) \quad \exists(\Sigma'',T'') \trianglerighteq (\Sigma',T') \quad S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2.$$

Suppose $Alg(\Sigma,T) \models S_1 \sqsubseteq S_2$, and consider $(\Sigma',T') \trianglerighteq (\Sigma,T)$.
According to BERGSTRA - TUCKER [7] there is a $(\Sigma'',T'') \trianglerighteq (\Sigma',T')$ for which HL is logically complete (See Def. 1.2.1).
Consequently: $S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2$. □

3.5. REMARK. All inclusions introduced above, except semantical inclusion, were obtained by quantification over the 'basic' prooftheoretical inclusion $\sqsubseteq_{HL}$. This suggests looking at all inclusions of the following general form:

$$S_1 \sqsubseteq^{\forall\exists\forall\ldots\exists}_{HL(\Sigma,T} S_2 \iff \forall(\Sigma_1,T_1) \trianglerighteq (\Sigma,T) \quad \exists(\Sigma_2,T_2) \trianglerighteq (\Sigma_1,T_1)$$

$$\forall(\Sigma_3,T_3) \trianglerighteq (\Sigma_2,T_2)\ldots\exists(\Sigma_{2n},T_{2n}) \trianglerighteq (\Sigma_{2n-1},T_{2n-1}) \quad S_1 \sqsubseteq_{HL(\Sigma_{2n},T_{2n})} S_2,$$

and likewise $S_1 \sqsubseteq^{\forall\exists\forall\ldots\forall}_{HL(\Sigma,T)} S_2$, and the dual notions obtained by interchanging ∃,A.
(Note that only alternating strings of quantifiers are interesting, since clearly $--\forall\forall-- = --\forall--$ and likewise for ∃.) So derivable inclusion w.r.t. (Σ,T) is $\sqsubseteq^{\forall}_{HL(\Sigma,T)}$, forced inclusion is $\sqsubseteq^{\exists\forall}_{HL(\Sigma,T)}$, and cofinal inclusion is $\sqsubseteq^{\forall\exists}_{HL(\Sigma,T)}$. (Inclusion in some refinement, $\sqsubseteq^{\exists}_{HL(\Sigma,T)}$, was not mentioned in this Section, because it seems to be of less importance).

Now it is easy to show (using RCT 2.4) that (dropping the subscript $HL(\Sigma,T)$) $\sqsubseteq^{\exists\forall} = \sqsubseteq^{\forall\exists\forall}$ and $\sqsubseteq^{\forall\exists} = \sqsubseteq^{\exists\forall\exists}$, which implies that only five essentially different inclusions exist, viz $\sqsubseteq^{i}$ where i = empty, ∀,∃,∀∃,∃∀.

4. PROTOTYPE PROOFS

In this section we will define the notion of 'prototype proof', which will play an important role in the sequel. Its main property is that every proof of some $\{p\}S\{q\}$ is a substitution instance of the prototype proof $\pi(S)$ corresponding to S. First we need an auxiliary concept.

4.1. DEFINITION. The class $IWP(\Sigma)$ (with typical elements S^*, S^{**},...) of interpolated while-programs is inductively defined by

$$S^* ::= S \mid \{p\}S^* \mid S^*\{p\} \mid \underline{if}\ b\ \underline{then}\ S_1^*\ \underline{else}\ S_2^*\ \underline{fi} \mid$$
$$\underline{while}\ b\ \underline{do}\ S^*\ \underline{od}.$$

Here $S \in WP(\Sigma)$. So the class of interpolated statements contains next to the usual statements also asserted statements and statements interlaced with assertions in an arbitrary way; but it contains also *proofs* of asserted statements. These will be singled out by means of the following extended proof rules.

4.2. DEFINITION. By means of the following axioms and extended proof rules we can derive proofs of asserted programs:

(1) *Assignment axiom scheme:* $\{p(t)\}\ \ x := t\ \ \{p\}$

(2) *Extended composition rule:* $$\frac{\{p\}S_1^*\{r\} \quad \{r\}S_2^*\{q\}}{\{p\}S_1^*\{r\}S_2^*\{q\}}$$

(3) *Extended contitional rule:*

$$\frac{\{p\wedge b\}\ S_1^*\ \{q\} \qquad\qquad \{p\wedge\neg b\}\ S_2^*\ \{q\}}{\{p\}\ \underline{\text{if}}\ b\ \underline{\text{then}}\ \{p\wedge b\}\ S_1^*\{q\}\ \underline{\text{else}}\ \{p\wedge\neg b\}\ \ S_2^*\{q\}\ \text{fi}\ \{q\}}$$

(4) *Extended iteration rule:* $$\frac{\{p\wedge b\}\ S^*\{p\}}{\{p\}\ \text{while}\ b\ \text{do}\{p\wedge b\}\ S^*\{p\}\ \text{od}\ \{p\wedge\neg b\}}$$

(5) *Extended consequence rule:* $$\frac{p\rightarrow p_1\ \ \{p_1\}\ S^*\ \{q_1\}\ \ q_1\rightarrow q}{\{p\}\{p_1\}S^*\{q_1\}\{q\}}$$

4.3. DEFINITION AND NOTATION.

(i) Let PR(Σ,T) be the class of proofs (interpolated programs) which can be derived using this axiom scheme and extended proof rules, such that in (5) only implications provable from T are used.
(ii) If $S^* \in$ IWP(Σ), then $\sigma(S^*)$ will denote the underlying program obtained by erasing all {p} in S^*.
(iii) If $S^* \in$ PR(Σ,T), then $\kappa(S^*)$ will denote the set of implications $p \rightarrow p'$ used in the derivation of S^*. Note that these implications can be read of directly from S^*:

$$\kappa(S^*) = \{p \rightarrow p' \mid \{p\}\ \{p'\} \subseteq S^*\}.$$

(Here "$\subseteq$" denotes the relation of being contained as a 'subword'.)
(iv) If $S^* \in$ PR(Σ,T) and $S^* = \{p\}\ S_1^*\ \{q\}$, then pre($S^*$) = p and post($S^*$) = q.
(v) Let $S^* \in$ PR(Σ,T). Then S^* is called a *reduced* proof, iff it contains no occurrence of a triple {p} {q} {r}.
(By the transitivity of $\rightarrow$, every proof may be supposed reduced, up to equivalence.)
(vi) Two interpolated programs S^*, S^{**} such that $\sigma(S^*) = \sigma(S^{**})$ = S are called *matching* if at every place the same number of assertions occur in S^*, S^{**}.
(vii) Let S^* = --{p}-- be an interpolated statement containing {p}. Then S^{**} = = --{p} {p}-- is called a *trivial expansion* of S^*.

In the following definition we will use a set of n-ary relation symbols $\{r_i \mid i\geq 0\}$. If $S^* \in$ IWP contains some of these r-symbols, $[S^*]_j$ will be the result of replacing each occurrence of r_i in S^* by $r_{(i,j)}$ where (,): $\mathbb{N}^2 \rightarrow \mathbb{N}$ is the usual bijective pairing function. (This device merely serves to 'refresh' the r-symbols where necessary.)

4.4. DEFINITION.

(i) Let S $\in$ WP(Σ) involve the variables $\vec{x}$ (= $x_1,\ldots,x_n$). By induction on the structure of S we define π'(S) as follows:

(1) $$\pi'(x_i{:=}t) = \{r_0(\vec{x})\ [t/x_i]\}\ \ x_i{:=}t\ \ \{r_0(\vec{x})\}.$$

(2) $$\pi'(S_1;S_2) = [\pi'(S_1)]_0\ [\pi'(S_2)]_1.$$

(That is, $\pi'(S_1)$ and $\pi'(S_2)$ are concatenated, without infix. Moreover, the r-symbols in $[\pi'(S_1)]_0$ are made distinct from those in $[\pi'(S_2)]_1$.)

(3) $\pi'(\underline{\text{if}}\ b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\ \underline{\text{fi}}) =$

$\{r_0(\vec{x})\}$ **if** b **then** $\{r_0(\vec{x})\wedge b\}[\pi'(S_1)]_2\ \{r_1(\vec{x})\}$

else $\{r_0(\vec{x})\wedge\neg b\}\ [\pi'(S_2)]_3\ \{r_1(\vec{x})\}$

fi $\{r_1(\vec{x})\}$.

(4) $\pi'(\underline{\text{while}}\ b\ \underline{\text{do}}\ S\ \underline{\text{od}}) =$

$\{r_0(\vec{x})\}$ **while** b do $\{r_0(\vec{x})\wedge b\}\ S^*$ od $\{r_0(\vec{x})\wedge\neg b\}\ \ \{r_1(\vec{x})\}$

where $S^* = [\pi'(S)]_4$ and $r_0(\vec{x}) = post(S^*)$.

(ii) Now $\pi(S) = \{r_0(\vec{x})\}\ [\pi'(S)]_0\ \{r_1(\vec{x})\}$.

$\pi(S)$ is called the *prototype proof* of S.

4.5. EXAMPLE. Let S be:

$x_1 := 0;$
$x_2 := 1;$
while $x_2 > x_3$
do **if** $x_1 = 0$
  then $x_3 := 0$
  else $x_1 := x_2+1$
  fi
od;
$x_1 := x_1+x_2$

Then $\pi(S)$ is as follows. (The assertions to the right of the vertical bar are for use in Example 4.7.1.)

$\pi(S) =$	ϕ
$\{r_1(x_1,x_2,x_3)\}$	$\{\underline{\text{true}}\}$
$\{r_2(0,x_2,x_3)\}$	$\{0=0\}$
$x_1:=0$	
$\{r_2(x_1,x_2,x_3)\}$	$\{x_1=0\}$
$\{r_3(x_1,1,x_3)\}$	$\{x_1=0\ \wedge\ 1=1\}$
$x_2:=1$	
$\{r_3(x_1,x_2,x_3)\}$	$\{x_1=0\ \wedge\ x_2=1\}$
$\{r_6(x_1,x_2,x_3)\}$	$\{x_1=0\ \wedge\ x_2=1\}$
while $x_2 > x_3$ **do**	
$\{r_6(x_1,x_2,x_3)\wedge x_2>x_3\}$	$\{x_1=0\ \wedge\ x_2=1\ \wedge\ x_2>x_3\}$
$\{r_4(x_1,x_2,x_3)\}$	$\{x_1=0\ \wedge\ x_2=1\ \wedge\ x_2>x_3\}$
if $x_1=0$ **then**	
$\{r_4(x_1,x_2,x_3)\wedge x_1=0\}$	$\{x_1=0\ \wedge\ x_2=1\ \wedge\ x_2>x_3\ \wedge\ x_1=0\}$
$\{r_5(x_1,x_2,0)\}$	$\{x_1=0\ \wedge\ x_2=1\ \wedge\ 0=0\}$

$x_3:=0$	
$\{r_5(x_1,x_2,x_3)\}$	$\{x_1=0 \wedge x_2=1 \wedge x_3=0\}$
$\{r_6(x_1,x_2,x_3)\}$	$\{x_1=0 \wedge x_2=1\}$
else	
$\{r_4(x_1,x_2,x_3) \wedge \neg x_1 = 0\}$	$\{x_1=0 \wedge x_2=1 \wedge x_2>x_3 \wedge \neg x_1 = 0\}$
$\{r_7(x_2+1,x_2,x_3)\}$	$\{x_2+1=0 \wedge x_2=1 \wedge x_3=0\}$
$x_1:=x_2+1$	
$\{r_7(x_1,x_2,x_3)\}$	$\{x_1=0 \wedge x_2=1 \wedge x_3=0\}$
$\{r_6(x_1,x_2,x_3)\}$	$\{x_1=0 \wedge x_2=1\}$
fi	
$\{r_6(x_1,x_2,x_3)\}$	$\{x_1=0 \wedge x_2=1\}$
od	
$\{r_6(x_1,x_2,x_3) \wedge \neg x_2>x_3\}$	$\{x_1=0 \wedge x_2=1 \wedge \neg x_2>x_3\}$
$\{r_8(x_1+x_2,x_2,x_3)\}$	$\{x_1+x_2=1 \wedge x_2=1 \wedge x_3\geq 1\}$
$x_1:=x_1+x_2$	
$\{r_8(x_1,x_2,x_3)\}$	$\{x_1=1 \wedge x_2=1 \wedge x_3\geq 1\}$
$\{r_9(x_1,x_2,x_3)\}$	$\{x_1=1 \wedge x_2=1 \wedge x_3\geq 1\}$

4.6. DEFINITION. Let $S^* \in IWP(\Sigma)$ contain the n-ary relation symbol r, and let $p = p(x_1,\ldots,x_n) \in L(\Sigma)$. (Note: p may contain other variables than those displayed.)

Then $\phi_r^p(S^*)$ is the result of replacing each $r(t_1,\ldots,t_n)$, occurring in S^*, by $p(t_1,\ldots,t_n)$. Likewise we define $\phi_{r_1,\ldots,r_n}^{p_1,\ldots,p_n}(S^*)$.

4.7. LEMMA. *Let* $S^* \in PR(\Sigma,T)$ *be a reduced proof such that* $\sigma(S^*) = S$. *Then* $\phi:\pi(S) \to S^*$ *for some substitution* ϕ *as in Definition 4.6.* (So every proof is an instance of the prototype proof.)

PROOF. Take S, S^* as in the lemma. We may suppose S^* and $\pi(S)$ are matching; otherwise only some trivial expansions (Definition 3.3) of S^* are required. Then we can construct by induction on the structure of S a substitution as required. This construction is entirely straightforward and routine; it will be left to the reader. □

4.7.1. EXAMPLE. Let S be as in Example 4.5; we use the abbreviations

$$S'' = \underline{if}\ x_1=0\ \underline{then}\ x_3:= 0\ \underline{else}\ x_1:=x_2+1\ \underline{fi}$$

$$S' = while\ x_2>x_3\ \underline{do}\ S''\ \underline{od}$$

$$S = x_1:=0;\ x_2:=1;\ S';\ x_1:=x_1+x_2.$$

Then the following proof of $\{\underline{true}\}S\{x_1=1 \wedge x_2=1 \wedge x_3\geq 1\}$, written as a column of asserted programs and implications, is a substitution instance of $\pi(S)$ as in Example 4.5, via the substitution ϕ displayed there (see the assertions to the right of the bar).

	0.	$\underline{\text{true}} \to 0=1$
	1.	$\{0=0\}x_1:=0\{x_1=0\}$
0,1:	2	$\{\underline{\text{true}}\}x_1:=0\{x_1=0\}$
	3	$x_1=0 \to x_1=0 \wedge 1=1$
2,3:	4	$\{\underline{\text{true}}\}x_1:=0\{x_1=0 \wedge 1=1\}$
	5	$\{x_1=0 \wedge 1=1\}x_2:=1\{x_1=0 \wedge x_2=1\}$
4,5:	6	$\{\underline{\text{true}}\}x_1:=0;\ x_2:=1\ \{x_1=0 \wedge x_2=1\}$
	7	$x_1=0 \wedge x_2=1 \wedge x_2>x_3 \wedge x_1=0 \to x_1=0 \wedge x_2=1 \wedge 0=0$
	8	$\{x_1=0 \wedge x_2=1 \wedge 0=0\}x_3:=0\{x_1=0 \wedge x_2=1 \wedge x_3=0\}$
7,8:	9	$\{x_1=0 \wedge x_2=1 \wedge x_2>x_3 \wedge x_1=0\}x_3:=0\{x_1=0 \wedge x_2=1 \wedge x_3=0\}$
	10	$\{x_2+1=0 \wedge x_2=1 \wedge x_3=0\}x_1:=x_2+1\{x_1=0 \wedge x_2=1 \wedge x_3=0\}$
	11	$x_1=0 \wedge x_2=1 \wedge x_2>x_3 \wedge x_1\neq 0 \to x_2+1=0 \wedge x_2=1 \wedge x_3=0$
10,11:	12	$\{x_1=0 \wedge x_2=1 \wedge x_2>x_3 \wedge x_1\neq 0\}x_1:=x_2+1\{x_1=0 \wedge x_2=1 \wedge x_3=0\}$
9,12:	13	$\{x_1=0 \wedge x_2=1 \wedge x_2>x_3\}S''\{x_1=0 \wedge x_2=1 \wedge x_3=0\}$
	14	$x_1=0 \wedge x_2=1 \wedge x_3=0 \to x_1=0 \wedge x_2=1$
13,14:	15	$\{x_1=0 \wedge x_2=1 \wedge x_2>x_3\}S''\{x_1=0 \wedge x_2=1\}$
15:	16	$\{x_1=0 \wedge x_2=1\}S'\{x_1=0 \wedge x_2=1 \wedge \neg x_2>x_3\}$
6,16:	17	$\{\underline{\text{true}}\}x_1:=0;\ x_2:=1;\ S'\{x_1=0 \wedge x_2=1 \wedge \neg x_2>x_3\}$
	18	$x_1=0 \wedge x_2=1 \wedge \neg x_2>x_3 \to x_1+x_2=1 \wedge x_2=1 \wedge x_3\geq 1$
	19	$\{x_1+x_2=1 \wedge x_2=1 \wedge x_3\geq 1\}x_1:=x_1+x_2\{x_1=1 \wedge x_2=1 \wedge x_3\geq 1\}$
18,19:	20	$\{x_1=0 \wedge x_2=1 \wedge \neg x_2>x_3\}\ x_1:=x_1+x_2\{x_1=1 \wedge x_2=1 \wedge x_3\geq 1\}$
17,20:	21	$\{\underline{\text{true}}\}S\{x_1=1 \wedge x_2=1 \wedge x_3\geq 1\}$.

4.8. PROPOSITION. *Let* $\Sigma^0 = \Sigma \cup \Sigma_{\pi(S)}$ *and* $T^0 = T \cup \kappa(\pi(S))$. *Then* $(\Sigma^0,T^0) \trianglerighteq (\Sigma,T)$.

PROOF. Take arbitrary p,q such that $HL(\Sigma,T) \vdash \{p\}S\{q\}$. (E.g. take q = true.) Let $\{p\}S^*\{q\} \in PR(\Sigma,T)$ be the corresponding proof; we may suppose it matches $\pi(S)$.

Now let $A \in Alg(\Sigma,T)$, so by soundness of HL we have $A \models \{p\}S\{q\}$. Further, it is not hard to see that the $r_i(\vec{x})$ can be interpreted in A just like the matching assertions in $\{p\}S^*\{q\}$.

Hence every $A \in Alg(\Sigma,T)$ can be expanded to an $A^0 \in Alg(\Sigma^0,T^0)$. So by the conservativity criterion 2.6, we have $(\Sigma^0,T^0) \trianglerighteq (\Sigma,T)$. □

5. PROOF SYSTEMS

Our interest is in formal criteria that imply program inclusion. The diagram described in the Introduction contains three such concepts: $\sqsubseteq^{\forall}$, $\sqsubseteq^{\exists\forall}$ and $\sqsubseteq^{\forall\exists}$ (in the notation of Remark 3.5). Now $\sqsubseteq^{\forall\exists}$ coincides with semantical program inclusion (3.4 plus 5.1) and therefore $\sqsubseteq^{\exists\forall}$ is a sufficient criterion (3.4(i)) as well as $\sqsubseteq^{\forall}$.

$HL(\Sigma,T) \vdash S_1 \sqsubseteq S_2$ is a semicomputable relation (5.2). It constitutes a formal proof system of a conventional nature. $\vdash$ is quite natural and suffices for many examples.

$\vdash (\sqsubseteq^{\forall})$ is not complete however (5.4(i)). The proof system $\Vdash (\sqsubseteq^{\forall\exists})$ provides a less effective but considerably stronger method (5.4(ii)). In fact $\Vdash$ is also incomplete (5.3). Because $\sqsubseteq^{\exists\forall}$ can hardly be considered a formal proof method, we are left with the problem of finding useful extensions of $\vdash$ and $\Vdash$. This seems to us to be a research topic of considerable importance.

5.1. PROPOSITION. *Cofinal inclusion implies semantical inclusion,* i.e.

$$\forall(\Sigma',T') \trianglerighteq (\Sigma,T) \quad \exists(\Sigma'',T'') \trianglerighteq (\Sigma',T') \quad S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2 \Rightarrow$$

$$Alg(\Sigma,T) \models S_1 \sqsubseteq S_2.$$

PROOF. Suppose $Alg(\Sigma,T) \not\models S_1 \sqsubseteq S_2$. Choose $A \in Alg(\Sigma,T)$, $\vec{a},\vec{b} \in A$ with $A \models S_1(\vec{a}) = \vec{b}$ and $A \not\models S_2(\vec{a}) = \vec{b}$. Let $k = |S_1(\vec{a})|$, i.e. $A \models Comp_{k,S_1}(\vec{a}) = \vec{b}$. One obtains a signature Σ^0 by adding names $\underline{\vec{a}}$ and $\underline{\vec{b}}$ for $\vec{a}$ and $\vec{b}$. Then let $\Sigma' = \Sigma^0 \cup \Sigma_{\pi(S_2)}$, $T' = T \cup \kappa(\pi(S_2))$. One proves $(\Sigma',T') \trianglerighteq (\Sigma,T)$ just like Proposition 4.8. Moreover, let $\theta = Comp_{k,S_1}(\underline{\vec{a}}) = \underline{\vec{b}} \wedge \forall\vec{x}\ (\vec{x}=\underline{\vec{a}} \rightarrow r_0(\vec{x})) \wedge \forall\vec{x}\ (r_1(\vec{x}) \rightarrow \neg\vec{x}=\underline{\vec{b}})$. Here $r_0(x) = pre(\pi(S_2))$ and $r_1(x) = post(\pi(S_2))$ (see Definition 4.3). Then $(\Sigma',T' \cup \{\theta\})$ is consistent (a model is found by expanding A). Clearly

$$HL(\Sigma',T' \cup \{\theta\})\ \{\vec{x}=\underline{\vec{a}}\}S_2\{\neg\vec{x}=\underline{\vec{b}}\}\ ;\ \text{it follows that}$$

$$HL(\Sigma',T') \vdash \{\theta\} \wedge \vec{x}=\vec{a}\}S_2\{\neg\vec{x}=\vec{b}\}.$$

Suppose $(\Sigma'',T'') \trianglerighteq (\Sigma',T')$, then $T'' \cup \{\theta\}$ is consistent and

$$HL(\Sigma'',T'') \vdash \{\theta \wedge \vec{x}=\underline{\vec{a}}\}S_2\{\neg\vec{x}=\underline{\vec{b}}\}\ .$$

Assume for a contradiction that $S_1 \sqsubseteq_{HL(\Sigma'',T'')} S_2$ then:

$$HL(\Sigma'',T'') \vdash \{\theta \wedge \vec{x}=\underline{\vec{a}}\}S_1\{\neg\vec{x}=\underline{\vec{b}}\}\ .$$

However in a model B of $T'' \cup \{\theta\}$ this asserted program is incorrect because $B \models Comp_{k,S_1}(\underline{\vec{a}}) = \underline{\vec{b}}$. □

5.2. THEOREM. $HL(\Sigma,T) \vdash S_1 \sqsubseteq S_2$ *and* $HL(\Sigma,T) \vdash S_1 \equiv S_2$ *as predicates of* (S_1,S_2) *are semidecidable in* T.

PROOF. Let $\Sigma^0 = \Sigma \cup \Sigma_{\pi(S_2)}$ and $T^0 = T \cup \kappa(\pi(S_2))$. (Σ^0,T^0) is found effectively (Σ,T). Now we claim that $HL(\Sigma,T) \vdash S_1 \sqsubseteq S_2 \Leftrightarrow HL(\Sigma^0,T^0) \vdash \{r_0(\vec{x})\}S_1\{r_1(\vec{x})\}$, which implies the theorem because of the semidecidable character of Hoare's Logic.

To prove the claim: $\Rightarrow$ is immediate. So assume $HL(\Sigma^0,T^0) \vdash \{r_0(\vec{x})\}S_1\{r_1(\vec{x})\}$. Let $\{r_0(\vec{x})\}S_1^*\{r_1(\vec{x})\} \in PR(\Sigma^0,T^0)$. Given some $(\Sigma',T') \trianglerighteq (\Sigma,T)$ assume $HL(\Sigma',T') \vdash \{p\}S_2\{q\}$. Let $\{p\}S_2^*\{q\} \in PR(\Sigma',T')$ be the corresponding proof which we may assume matching with $\pi(S_2)$. By Lemma 4.7, $\{p\}S_2^*\{q\}$ is an instance of $\pi(S_2)$ via some substitution ϕ. Applying the substitution ϕ on $\{r_0(\vec{x})\}S_1^*\{r_1(\vec{x})\}$ we obtain a proof $\{p\}\phi(S_1^*)\{q\}$ in $PR(\Sigma',T')$. Consequently $HL(\Sigma',T') \vdash \{p\}S_1\{q\}$. □

Let $A = (\mathbb{N}, 0,S,P)$, $\Sigma = \Sigma_A$ and $T = T_A$. *These notation conventions will hold until the end of this paper.*

5.3. THEOREM. $\Vdash$ *is incomplete. In fact there are* $S_1,S_2 \in WP(\Sigma)$ *with* $Alg(\Sigma,T) \models S_1 \sqsubseteq S_2$ *but* $HL(\Sigma,T) \not\Vdash S_1 \sqsubseteq S_2$.

PROOF. An essentially straightforward verification shows that $Alg(\Sigma,T) \models S_1 \sqsubseteq S_2$ is a complete Π_2^0 predicate of (S_1,S_2) whereas $HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_2$ is a Σ_2^0 predicate of (S_1,S_2). Recursion theory then tells that both predicates must differ. □

5.4. PROPOSITION. *Let* S_1, S_2 *be the following programs over* Σ:

S_1= y:=0; S' *where* S'= <u>while</u> x ≠ 0 <u>do</u> y:=Sy; x:=Px <u>od</u>

S_2= y:=x; x:=0

then (i) $HL(\Sigma,T) \not\vdash S_1 \sqsubseteq S_2$ *but* (ii) $HL(\Sigma,T) \Vdash S_1 \sqsubseteq S_2$.

PROOF. (i) $S_1 \not\sqsubseteq_{HL(\Sigma,T)} S_2$ because

(1) $HL(\Sigma,T) \vdash \{x=z\}S_2\{x=0 \wedge y=z\}$

(2) $HL(\Sigma,T) \not\vdash \{x=z\}S_1\{x=0 \wedge y=z\}$.

Here (2) requires a proof: suppose not (2), then

$$HL(\Sigma,T) \vdash \{x=z \wedge y=0\}S_1\{x=0 \wedge y=z\}.$$

Hence there must be an invariant $r(x,y,z)$ such that $T \vdash \phi_1 \wedge \phi_2 \wedge \phi_3$ where

$$\phi_1 = x=z \wedge y=0 \rightarrow r(x,y,z)$$

$$\phi_2 = \exists x',y' [x'\neq 0 \wedge x=Px' \wedge y=Sy' \wedge r(x',y',z)] \rightarrow r(x,y,z)$$

$$\phi_3 = x=0 \wedge r(x,y,z) \rightarrow y=z.$$

Also $A \models \phi_1 \wedge \phi_2 \wedge \phi_3$. However, a simple proof shows then that $A \models r(\underline{a},\underline{b},\underline{c}) \Leftrightarrow$ $\Leftrightarrow a+b=c$, in contradiction with the non-definability of + in A.
(ii). Let $A' = (\mathbb{N}, 0,S,P,+)$. Because (Σ,T) is complete, we have $(\Sigma_{A'},T_{A'}) \trianglerighteq (\Sigma,T)$. Using the method of prototype proofs, $HL(\Sigma_{A'},T_{A'}) \vdash S_1 \sqsubseteq S_2$ is established as follows: consider $\pi(S_2)$, this is

$$\{r_0(x,y)\}\{r_1(x,x)\} \quad y:=x \quad \{r_1(x,y)\}\{r_2(0,y)\} \quad x:=0 \quad \{r_2(x,y)\}\{r_3(x,y)\}.$$

So we have to find a proof of $\{r_0(x,y)\}$ S_1 $\{r_3(x,y)\}$ in the theory

$$T_{A'} \cup \{r_0(x,y) \rightarrow r_1(x,x),$$
$$r_1(x,y) \rightarrow r_2(0,y),$$
$$r_2(x,y) \rightarrow r_3(x,y)\}.$$

This is indeed possible:

$\{r_0(x,y)\}\{r_1(x,x)\}\{r_2(0,x)\}\{r_3(0,x)\}$

y:=0

$\{r_3(0,x) \wedge y=0\}$

$\{\exists x_0[r_3(0,x_0) \wedge x=x_0 \wedge y=0]\}$

$\{\exists x_0[r_3(0,x_0) \wedge x+y=x_0]\}$

<u>while</u> $x\neq 0$ <u>do</u>

$\{\exists x_0[r_3(0,x_0) \wedge x+y=x_0 \wedge x\neq 0]\}$

$\{\exists x_0[r_3(0,x_0) \wedge Px+Sy=x_0 \wedge x\neq 0]\}$

y:=Sy

$\{\exists x_0[r_3(0,x_0) \wedge Px+y=x_0 \wedge x\neq 0]\}$

x:=Px

$\{\exists x_0[r_3(0,x_0) \wedge x+y=x_0]\}$

od

$\{\exists x_0[r_3(0,x_0) \wedge x+y=x_0] \wedge x=0\}$

$\{\exists x_0[r_3(0,x_0) \wedge y=x_0 \wedge x=0]\}$

$\{r_3(x,y)\}$.

REFERENCES.

[1] Back, R.J., Correctness preserving program refinements: proof theory and applications, Mathematical Centre Tracts 131, Mathematical Centre, Amsterdam, 1980.

[2] De Bakker, J.W., Recursive procedures, Mathematical Centre Tracts 24, Mathematical Centre, Amsterdam, 1973.

[3] De Bakker, J.W., Mathematical theory of program correctness, Prentice-Hall International, London, 1980.

[4] Bergstra, J.A. & J.W. Klop, Proving program inclusion using Hoare's logic, Mathematical Centre, Department of Computer Science, Research Report IW 176, Amsterdam 1981.

[5] Bergstra, J.A. & J. Terlouw, A characterization of program equivalence in terms of Hoare's logic, Proceedings of the G.I. Jahrestagung München 1981, Springer LNCS 123.

[6] Bergstra, J.A. & J.V. Tucker, Expressiveness and the completeness of Hoare's logic, Mathematical Centre, Department of Computer Science Research Report IW 149, Amsterdam, 1980. To appear in JCSS.

[7] Bergstra, J.A. & J.V. Tucker, Two theorems about the completeness of Hoare's logic, Mathematical Centre, Department of Computer Science Research Report IW 165, Amsterdam, 1981.

[8] Boolos, G.S. & R.C. Jeffrey, Computability and Logic, Cambridge University Press (1974, 1980).

[9] Cousineau, G. & P. Enjalbert, Program equivalence and provability, Mathematical Foundations of Computer Science 1979, Proc. 8th Symp., Olomouc (Czechoslovakia), Springer Lecture Notes in Computer Science 74, p.237-245.

[10] Guessarian, I., Algebraic Semantics, Springer Lecture Notes in Computer Science 99, 1981.

[11] Harel, D., A. Pnueli & J. Stavi, A complete axiom system for proving deductions about recursive programs, in Proc. 9th ACM Symp. Theory of Computing, Boulder, 1977.

[12] Hemerik, C., Relaties tussen taaldefinitie en taalimplementaie, in Colloquium Capita Implementatie van Programmeertalen, J.C. van Vliet (red.), MC Syllabus 42, Mathematical Centre, Amsterdam 1980.

[13] Manna, Z., Mathematical theory of computation, McGraw-Hill, New York, 1974.

[14] Meyer, A.R. & J.Y. Halpern, Axiomatic definitions of programming languages. A theoretical assessment, Proceedings 7th ACM Symp. on Principles of Programming Languages, ACM, New York, 1980, p.203-212.

[15] Monk, J.D., Mathematical Logic, Springer-Verlag (1976).

[16] Russell, B., Correctness of the compiling process based on axiomatic semantics, Acta Informatica 14, p.1-20, 1980.

[17] Shoenfield, J., Mathematical Logic, Reading, Addison-Wesley (1967).