stichting
mathematisch
centrum

AFDELING INFORMATICA IW 207/82 SEPTEMBER
(DEPARTMENT OF COMPUTER SCIENCE)

J.A. BERGSTRA & J.V. TUCKER

# HOARE'S LOGIC FOR PROGRAMMING LANGUAGES WITH TWO DATA TYPES

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: D.2.4, F.3.1

Hoare's logic for programming languages with two data types *)

by

J.A. Bergstra** & J.V. Tucker***

ABSTRACT

We consider the completeness of Hoare's logic with a first-order assertion language applied to while-programs containing variables of two (or more) distinct types. Whilst Cook's completeness theorem generalises to many-sorted interpretations certain fundamentally important structures turn out not to be expressive. We study the case of programs with distinguished counter variables and boolean variables adjoined; for example, we show that adding counters to arithmetic destroys expressiveness.

KEY WORDS & PHRASES : *Hoare's logic, partial correctness, while-programs, completeness, expressiveness, many-sorted programs, many-sorted first-order logic*

---

*) This paper is not for review as it will be published elsewhere. Most of the work for this paper was performed in the course of two visits to the Mathematical Centre by the second author in February and July 1982.

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

***) Department of Computer Studies, University of Leeds, LEEDS LS2 9JT, Britain

INTRODUCTION

Since the publication of HOARE [5] there has accumulated a large body of knowledge about proof systems for formally verifying the partial correctness of programs. Proof systems have been made which include a wide variety of programming features and, in particular, the soundness and completeness of these systems have been successfully analysed along the lines first set down in COOK [4]. To obtain information about what has been achieved, at least for the sequential control aspects of programming languages, see APT [1].

In this note we consider a simple feature of most programming languages which has gone unnoticed to date, namely the property that *there may be two (or more) distinct types of variable or identifier in a single program*. Specifically, we will concentrate on the completeness of Hoare's logic for while-programs having variables of an arbitrary, unspecified type together with *boolean variables*, and with *natural number variables* or *counters*. We prove theorems which demonstrate that whilst Cook's account of completeness generalises to include boolean variables it is, surprisingly, unable to cope with while-programs with counters.

In Section 1 we summarise prerequisites and observe that Cook's completeness theorem for Hoare's logic for while-programs applied to first-order expressive structures generalises to the many-sorted case. However, in Section 2, we prove that adding arithmetic N to an expressive structure A can lead to a non-expressive two-sorted interpretation [A,N]. In particular, we prove that adding arithmetic N to arithmetic N leads to a non-expressive structure [N,N] and, indeed, that Hoare's logic for [N,N] is incomplete (Theorem 2.3). Thus, there is a general completeness theorem for the two-type situation, but it cannot be applied to a canonical example. In Section 3 we show that adding booleans does not give rise to a similar problem (Theorem 3.1); the same is true for finite counters, by generalising the argument (Theorem 3.7).

We wish to thank E.R. Olderog for useful discussions on the subject matter of this paper.

## 1. ASSERTIONS, PROGRAMS AND HOARE'S LOGIC

In addition to necessary prerequisites about two-sorted syntax and semantics, we outline the fate of Cook's study [4] of Hoare's logic when generalised to the two-sorted situation as this is the background of our main results. We assume the reader is familiar with the one-sorted case : as well as [4] we recommend APT [1] and DE BAKKER [2] for clear accounts of the subject. For a thorough discussion of the completeness problem for Hoare's logic see [3]. For a thorough discussion of program correctness in the many-sorted situation see [6].

SYNTAX. The first-order language $L(\Sigma)$ of some two-sorted signature $\Sigma$ is based upon two sets of variables

$$x^1_1, x^1_2, \ldots \quad \text{and } x^2_1, x^2_2, \ldots,$$

of sorts 1 and 2 respectively, and the constant, function and relation symbols of $L(\Sigma)$ are those of $\Sigma$ together with equality symbols of sorts 1 and 2.

The usual inductive definition of term now yields two kinds of term giving values of sort 1 and sort 2. Atomic formulae have the forms

$$t^i =_i s^i \quad \text{and} \quad R(y_1^{i_1}, y_2^{i_2}, \ldots, y_k^{i_k})$$

where $t^i, s^i$ are terms (having values) of sort i, $=_i$ is the equality symbol for sort i, R is a relation symbol and the $y_j^{i_j}$ are variables of sort $i_j$ $j=1,\ldots,k$ and $i, i_j \in \{1,2\}$.

The well-formed formulae of $L(\Sigma)$ are made inductively by applying the logical connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$ and the quantifiers

$$\forall x^1_j \quad \exists x^1_j \quad \forall x^2_j \quad \exists x^2_j \qquad j \in \mathbb{N}$$

in the usual way.

Using the syntax of $L(\Sigma)$ the set $WP(\Sigma)$ of all <u>while</u> - programs over $\Sigma$ is defined in the obvious way. Note, in particular, that there are two kinds of assignment statement

$$x^1_j := t^1 \quad \text{and} \quad x^2_j := t^2$$

but that boolean tests in control statements are simply quantifier-free formulae of $L(\Sigma)$ and may refer to both sorts.

By a *specified* or *asserted program* we mean a triple of the form $\{p\}S\{q\}$ where $p,q \in L(\Sigma)$ and $S \in WP(\Sigma)$.

SEMANTICS. The semantics of $L(\Sigma)$ is based on two-sorted structures $A$ of signature $\Sigma$ and is formally defined in the usual manner as established by Tarski. The set of all sentences of $L(\Sigma)$ which are true in structure $A$ is called the first-order theory of $A$ and is denoted $Th(A)$. If $\phi \in L(\Sigma)$ the set defined in $A$ by $\phi$ we denote $\phi[A]$.

For the semantics of $WP(\Sigma)$ on an interpretation $A$ we leave the reader free to choose any sensible account of while-program computation in one-sorted structures and then to generalise it. Certainly, the operational and denotational semantics given in DE BAKKER [2] have natural many-sorted generalisations : see [6].

We suppose that the meaning of $S \in WP(\Sigma)$ on interpretation $A$ is defined as a state transformation

$$M_A(S) : STATES(A) \to STATES(A)$$

Also if $S$ has $n$ variables of sort 1 and $m$ variables of sort 2 then $STATES(A) \cong A_1^{\ n} \times A_2^{\ n}$, where $A_1$, $A_2$ are the domains of sorts 1,2 in $A$, and we suppose that $M_A(A)$ is represented by a mapping

$$\hat{M}_A(S) : A_1^{\ n} \times A_2^{\ m} \to A_1^{\ n} \times A_2^{\ m}$$

Putting together the semantics of $L(\Sigma)$ and $WP(\Sigma)$ we consider the partial correctness semantics of the specified programs : $\{p\}S\{q\}$ is valid on $A$, written $A \models \{p\}S\{q\}$, if when $p$ is true then either $S$ diverges or $S$ converges to a state at which $q$ is true. The set of all specified programs valid on $A$ is called the partial correctness theory of $A$ and we write

$$PC(A) = \{\{p\}S\{q\} : A \models \{p\}S\{q\}\}.$$

HOARE'S LOGIC Hoare's logic for the two-sorted $WP(\Sigma)$ has exactly the same axiom scheme for assignment statements and the same rules for composition, conditionals and iteration. In addition, any first-order theory $T$ may be employed to provide a specification for the underlying data types and T affects program correctness proofs via the Rule of Consequence (see [5,4]). The set of all specified programs provable from T is denoted HL(T).

In this paper we are interested in proving correctness with respect to a given two-sorted structure A. Cook's work on the single-sorted version of this case generalises to provide us with the following account:

1.1 SOUNDNESS THEOREM. *If* $A \models T$ *then* $HL(T) \subset PC(A)$.

The assertion language $L(\Sigma)$ is said to be expressive for $WP(\Sigma)$ over A if for any $p \in L(\Sigma)$ and $S \in WP(\Sigma)$ there is a formula $SP(p,S) \in L(\Sigma)$ that defines the strongest postcondition $SP_A(p,S)$ of S with respect to p over A,

$$SP_A(p,S) = \{\sigma \in STATES(A) : \exists \tau [M_A(S)(\tau)\downarrow \sigma \ \& \ p(\tau)]\}.$$

Notice that expressiveness is actually a property of the interpretation A rather than $L(\Sigma)$.

1.2 COOK'S COMPLETENESS THEOREM. *Suppose* $L(\Sigma)$ *is expressive for* $WP(\Sigma)$ *over* A *and let* $T = Th(A)$. *Then* $HL(T) = PC(A)$.

In view of Theorem 1.2 we define $HL(A) = HL(Th(A))$, and observe that HL(A) represents the strongest Hoare logic for analysing correctness on A because it is equipped with all first-order true facts about A.

1.3 THEOREM *If* A *is finite then* A *is expressive and* HL(A) *is complete.*

## 2. ADDING ARITHMETIC

Semantically, adding counters or booleans to while-programs is effected by interpreting the two-sorted programming language $WP(\Sigma)$ on certain two-sorted structures of the following form.

Let A and B be single-sorted structures with disjoint signatures $\Sigma_A$ and $\Sigma_B$ respectively. Then we define the *join* [A,B] of A and B to be the two-sorted structure of signature $\Sigma_{A,B} = \Sigma_A \cup \Sigma_B$ whose domains and operations are simply those of A and B.

What is noteworthy in this operation on structures is that algebraically A and B remain independent data types. Adding arithmetic means computing on structures $[A, \mathbb{N}]$ where $\mathbb{N}$ is the standard model of arithmetic. Adding booleans means computing on structures $[A, \mathbb{B}]$ where $\mathbb{B} = \{tt, ff\}$ equipped with $\wedge, \neg$ (say).

The main result in this section is that Hoare's logic is incomplete when applied to structures $[A, \mathbb{N}]$. Before proving this we will first study the join in general.

2.1 PROPOSITION. *If* [A,B] *is expressive then* A *and* B *are expressive.*

PROOF. We begin by proving a basic fact about first-order definability on [A,B].

Let H be the smallest set of $\Sigma_{A,B} = \Sigma_A \cup \Sigma_B$ formulae that contains $L(\Sigma_A)$ and $L(\Sigma_B)$ and is closed under $\neg, \wedge, \vee$. Thus, H does *not* contain formulae with quantifiers ranging over different sorts such as

$$\forall x^A \ (\phi^A \wedge \phi^B)$$

2.2 SEPARATION OF VARIABLES LEMMA. *Each formula* $\phi \in L(\Sigma_{A,B})$ *equivalent to a formula of* H.

PROOF. It is sufficient to show that, up to provable equivalence, $H \subset L(\Sigma_{A,B})$ is closed under quantification $\exists x^A$ and $\exists x^B$.

Let $\phi \in H$. Then $\phi$ can be rewritten as a formula in H in disjunctive normal form, and by simple transformations we proceed :

$$\phi \equiv \bigvee_{i=1}^{s} \bigwedge_{j=1}^{t} \phi_{i,j}$$

$$\equiv \bigvee_{i=1}^{s} (\bigwedge_{j=1}^{t} \phi^A_{i,j}) \wedge (\bigwedge_{j=1}^{t} \phi^B_{i,j})$$

$$\equiv \bigvee_{i=1}^{s} (\Phi_i^A \wedge \Phi_i^B)$$

Then we have $$\exists x^A \phi \equiv \exists x^A (\bigvee_{i=1}^{s} (\Phi_i^A \wedge \Phi_i^B))$$

$$\equiv \bigvee_{i=1}^{s} \exists x^A (\Phi_i^A \wedge \Phi_i^B)$$

$$\equiv \bigvee_{i=1}^{s} (\exists x^A \Phi_i^A) \wedge \Phi_i^B$$

This last formula belongs to H; the proof of the Lemma 2.2 is complete.

To prove the proposition we assume [A,B] is expressive and prove that A is expressive (the case for B follows mutatis nomine).

Let $\phi \in L(\Sigma_A)$ and $S \in WP(\Sigma_A)$. Let $SP(\phi,S)$ define the strongest postcondition $SP_{[A,B]}(\phi,S)$ on [A,B]. By the Separation of Variables Lemma 2.2,

$$SP(\phi,S) \equiv \bigvee_{i=1}^{s} (\psi_i^A \wedge \psi_i^B)$$

Because $\phi$ and S involve variables of type A only, the components $\psi_i^B$ for $1 \le i \le s$ are closed and can be replaced by their propositional values true and false. This done we obtain a formula $\psi \in L(\Sigma_{A,B})$, equivalent to $SP(\phi,S)$, that is first-order over $\Sigma_A$ and, indeed, $\psi$ defines $SP_A(\phi,S)$ on A. □

Our main result implies that the converse of Proposition 2.1 is false. Let $\mathbb{N}$ denote standard model of arithmetic; to be precise let

$$\mathbb{N} = (\{0,1,\ldots\}\ 0,1,x+1,x\dot{-}1,x+y,\ x\cdot y).$$

Consider the structure $[N_1,N_2]$ of signature $\Sigma_{1,2}$ wherein $N_1 = \mathbb{N}$ has signature $\Sigma_1$ and $N_2 = \mathbb{N}$ has signature $\Sigma_2$, i.e. $[N_1,N_2]$ is a pair of algebraically independent copies of $\mathbb{N}$. We are looking at the case of adding arithmetic to arithmetic, so to say.

2.3 THEOREM. *The two sorted structure $[N_1,N_2]$ is not expressive and $HL([N_1,N_2])$ is not complete.*

PROOF. Consider the following program

$$S ::= \; x:=0; \; z:=0;$$
$$\underline{\text{while}} \; x \neq y \; \underline{\text{do}} \; x:=x+1; \; z:=z+1 \; \underline{\text{od}}$$

with x,y variables of sort 1 and z a variable of sort 2. The strongest post-condition of S with respect to true is

$$sp(\underline{true},S) = \{(a,b,c) \in N_1 \times N_1 \times N_2 \; : \; a=b=c=n \in \mathbb{N}\}.$$

Suppose $sp(\underline{true},S)$ is first-order definable over $[N_1,N_2]$ then clearly "the diagonal" $\Delta = \{(a,b) \in N_1 \times N_2 \; : \; a=b=n \in \mathbb{N}\}$ is first-order definable : to this latter statement we derive a contradiction.

By the Separation of Variables Lemma 2.2, it is sufficient to show that $\Delta$ is not definable by a formula of $H(\Sigma_{1,2})$.

Suppose for a contradiction that $\Delta$ is definable by $\phi \in H(\Sigma_{1,2})$ with free variables x,y of sorts 1 and 2; thus,

$$\Delta = \{(a,b) \in N_1 \times N_2 \; : \; [N_1,N_2] \models \phi(a,b)\} .$$

Now $\phi$ can be rewritten in disjunctive normal form

$$\phi \equiv \bigvee_{i=1}^{s} \bigwedge_{j=1}^{t} \phi_{i,j}$$

where $\phi_{i,j} \in L(\Sigma_1) \cup L(\Sigma_2)$ for $1 \le i \le s$ and $1 \le j \le t$. This can be compressed to

$$\phi \equiv \bigvee_{i=1}^{s} (\Phi_i^1 \wedge \Phi_i^2)$$

where $\Phi_i^1 \in L(\Sigma_1)$ and $\Phi_i^2 \in L(\Sigma_2)$ with free variables x and y respectively. For $1 \le i \le s$, set

$$\Delta_i = \{(a,b) \in N_1 \times N_2 \; : \; [N_1,N_2] \models \Phi_i^1(a) \wedge \Phi_i^2(b)\},$$

so that $\Delta = \bigcup_{i=1}^{s} \Delta_i$. At least one $\Delta_i$ is infinite, say $\Delta_0$. We choose two points $(a,a)$, $(b,b) \in \Delta_1$ with $a \neq b$. Now

$$[N_1,N_2] \models \Phi_0^1(a) \wedge \Phi_0^2(a) \text{ and } [N_1,N_2] \models \Phi_0^1(b) \wedge \Phi_0^2(b).$$

Thus,

$$[N_1,N_2] \models \Phi_0^1(a) \wedge \Phi_0^2(b) \; .$$

This means that $(a,b) \in \Delta_0 \subset \Delta$ which is not the case. Therefore, $[N_1,N_2]$ is not expressive.

In order to see that $HL([N_1,N_2])$ is not complete consider the program

$$S_2 ::= \underline{\text{while}}\ x\neq 0 \wedge y\neq 0 \wedge z\neq 0$$
$$\underline{\text{do}}\ x:=x\dot{-}1;\ y:=y\dot{-}1;\ z:=z\dot{-}1\ \underline{\text{od}}.$$

Clearly,

$$[N_1,N_2] \models \{\underline{\text{true}}\}\ S_1;S_2\{x=0\wedge y=0\wedge z=0\}\ .$$

In order to prove this valid asserted program using Hoare's logic, an intermediate assertion $\theta$ must be found i.e. a formula such that

$$[N_1,N_2] \models \{\underline{\text{true}}\}\ S_1\ \{\theta\}$$
$$[N_1,N_2] \models \{\theta\}\ S_2\ \{x=0\wedge y=0\wedge z=0\}\ .$$

Thus,

$$sp_{[N_1,N_2]}(\underline{\text{true}},\ S_1) \subset \theta[N_1,N_2]$$

$$\theta[N_1,N_2] \subset wp_{[N_1,N_2]}(S_2,x=0\wedge y=0\wedge z=0)\ .$$

But

$$wp_{[N_1,N_2]}(S_2,x=0\wedge y=0\wedge z=0) = sp_{[N_1,N_2]}(\underline{\text{true}},S_1)$$

and hence $\theta[N_1,N_2] = sp(\underline{\text{true}},S_1)$. This contradicts the fact that $sp(\underline{\text{true}},S_1)$ is not definable. □

## 3. ADDING BOOLEANS

Let A be a single-sorted structure of signature $\Sigma_A$ and let $\mathbb{B} = (\{tt,ff\}, \wedge, \neg)$.

3.1 <u>THEOREM</u> *If* A *is expressive then* $[A,\mathbb{B}]$ *is expressive and consequently* $HL([A,\mathbb{B}])$ *is complete.*

PROOF. We distinguish two cases : A is finite and A is infinite. When A is finite, $[A,\mathbb{B}]$ is finite and expressive by Theorem 1.3. Suppose that A is infinite.

We add two new constants $c=(c_1,c_2)$ to $L(\Sigma_A)$ to make $L(\Sigma_A^c)$. Adjoining $a=(a_1,a_2)$ to A to make $A^a$, an interpretation for $\Sigma_A^c$, we observe

3.2 LEMMA *If* A *is expressive with respect to* $L(\Sigma_A)$ *then* $A^a$ *is expressive with respect to* $L(\Sigma_A^c)$. *Moreover, given* $\phi \in L(\Sigma_A^c)$ *and* $S \in WP(\Sigma_A^c)$ *there is* $SP(\phi,S) \in L(\Sigma_A^c)$ *that defines* $sp_A a(\phi,S)$ *uniformly for any interpretation* **a** *for* c.

3.3 LEMMA *If* A *is expressive and* $a=(a_1,a_2)$ *with* $a_1 \neq a_2$ *then the structure* $[A^a,\mathbb{B}]$ *is expressive.*

PROOF. Let $\phi \in L(\Sigma^c_{A,B})$ and $S \in WP(\Sigma^c_{A,B})$ have variables among

$$x = x_1, \ldots, x_n \quad \text{and} \quad y = y_1, \ldots, y_m$$

of types $A^a$ and $\mathbb{B}$ respectively. We will construct a formula $SP(\phi,S) \in L(\Sigma^c_{A,B})$ which defines $sp_{[A^a,\mathbb{B}]}(\phi,S)$ .

IDEA OF CONSTRUCTION We use $a_1, a_2$ to represent in $A^a$ the elements tt, ff from $\mathbb{B}$, and we simulate $\phi$ and S on $A^a$ by a formula $\phi^*$ and program $S^*$ over $\Sigma_A^c$. By the expressiveness of A and $A^a$ (Lemma 3.2), there is a formula $SP(\phi^*, S^*)$ in $L(\Sigma_A^c)$ to define $sp_{A^a}(\phi^*,S^*)$ and from this we will obtain a formula $SP(\phi,S)$ in $L(\Sigma^c_{A,B})$ for $sp_{[A^a,\mathbb{B}]}(\phi,S)$.

First, we consider semantically an encoding $h:(A^a)^n \times \mathbb{B}^m \to (A^a)^n \times (A^a)^m$ defined by $h(d,b) = (d,e)$ where for $i=1,\ldots,m$

$$e_i = \begin{cases} a_1 & \text{if } b_i = tt \\ a_2 & \text{if } b_i = ff \end{cases} .$$

We transform $\phi$ to $\phi^*$ in such a way that

3.4 REQUIREMENT *For all* $\sigma \in (A^a)^n \times \mathbb{B}^m$

$$[A^a,\mathbb{B}] \models \phi(\sigma) \text{ if, and only if, } A^a \models \phi^*(h(\sigma)).$$

$\phi^*$ is defined as follows. First we use the following syntactic rewrite rules to eliminate the operators *or, not* of type $\mathbb{B}$ in favour of the $L(\Sigma^c_{A,B})$ constructs $\vee, \wedge, \neg$.

$$\mathit{or}(t_1,t_2) = t_3 \quad \geq (t_1=c_1 \wedge t_3=c_1) \vee (t_2=c_1 \wedge t_3=c_1)$$

$$(t_1=c_2 \wedge t_2=c_2 \wedge t_3=c_1)$$

$$\mathit{not}(t_1) = t_2 \qquad \geq \neg(t_1=t_2)$$

The resulting formula we denote $\phi_1$ .

Next choose new variables $z = z_1,\ldots,z_m$ of sort A and replace each occurrence of $y_i$ of type $\mathbb{B}$ in $\phi_1$ by $z_i$. In addition, replace each instance of <u>true</u>, <u>false</u> in $\phi_1$ by $c_1,c_2$ respectively. The actions result in a formula $\phi_2$.

Define $\phi^* = \phi_2 \wedge \bigwedge_{i=1}^{m} (z_i=c_1 \vee z_i=c_2)$ .

The proof of 3.4 is by induction on the complexity of $\phi$ and is ommitted.

Now we must transform S to S* such that executing S* on $A^a$ simulates S on $[A^a,\mathbb{B}]$ via h; more formally :

3.5 <u>REQUIREMENT</u> *The following diagram commutes :*

$$\begin{tikzcd}
(A^a)^n \times (A^a)^m \arrow[r, "\hat{M}(S^*)"] & (A^a)^n \times (A^a)^m \\
(A^a)^n \times \mathbb{B}^m \arrow[u, "h"] \arrow[r, "\hat{M}(S)"'] & (A^a)^n \times \mathbb{B}^m \arrow[u, "h"']
\end{tikzcd}$$

S* is obtained from S by rewriting the latter's $\mathbb{B}$-terms. With the same variables z as chosen earlier, boolean conditions in control statements are rewritten without their $\mathbb{B}$-operations *or, not* just as above. To remove the operators from assignments the following five syntactic rewrite rules are applied wherein the * operation on formulae is that already defined :

$$
\begin{array}{lll}
y_i := \underline{true} & \geq & z_i := c_1 \\
y_i := \underline{false} & \geq & z_i := c_2 \\
y_i := y_j & \geq & z_i := z_j \\
y_i := \underline{or}(t_1,t_2) & \geq & \underline{if}(t_1=\underline{true})^* \ \underline{then}\ y_i := c_1 \\
 & & \qquad \underline{else}\ \underline{if}(t_2=\underline{true})^* \ \underline{then}\ y_i:=c_1 \\
 & & \qquad\qquad \underline{else}\ y_i:=c_2 \\
 & & \qquad \underline{fi} \\
 & & \underline{fi} \\
y_i := \underline{not}(t) & \geq & \underline{if}\ (t=\underline{true})^* \ \underline{then}\ z_i:=c_2 \\
 & & \qquad \underline{else}\ z_i:=c_1 \\
 & & \underline{fi}
\end{array}
$$

The proof of 3.5 is by induction on the complexity of S; it requires 3.4.

From 3.4 and 3.5 we can conclude that

$$h(sp_{[A^a,\mathbb{B}]}(\phi,S)) = sp_{A^a}(\phi^*,S^*) .$$

Because A is expressive $A^a$ is expressive and we can take a formula $\theta^*$ in $L(\Sigma_A^c)$ which defines $sp_{A^a}(\phi^*,S^*)$ in $A^a$ irrespectively of the values $a_1,a_2$ for $c_1,c_2$. Our task now is to find $\theta \epsilon L(\Sigma_{A,B}^c)$ such that

3.6 <u>REQUIREMENT</u> $h(\theta[A^a,\mathbb{B}]) = \theta^*[A^a]$

In consequence of the fact that h is injective, $\theta[A^a,B]=sp_{[A^a,B]}(\phi,S)$.

$\theta$ is found as follows : we rewrite $\theta^*$ as a formula $\theta_1$ in which the variables in z occur only in the forms

$$z_i = c_1 \quad \text{or} \quad z_i = c_2 .$$

This is accomplished by applying the rewrite rules

$$\alpha(z_i) \geq (z_i=c_1 \wedge \alpha[c_1/z_i]) \vee (z_i=c_2 \wedge \alpha[c_2/z_i])$$

where $\alpha$ is any subformula of $\theta^*$ in which $z_i$ appears in an inappropriate form.

$\theta$ is now obtained from $\theta_1$ by replacing the equations $z_i=c_1$ and $z_i=c_2$ by $y_i=\underline{true}$ and $y_i=\underline{false}$. To prove 3.6 one proves that

$$h(\theta[A^a,\mathbb{B}]) = \theta_1[A^c].$$

By construction, $(\theta)^* = \theta_1$ and so we are done. $\square$

Notice that the formula $\theta \in L(\Sigma^c_{A,\mathbb{B}})$ above defines $sp_{[A^a,\mathbb{B}]}(\phi,S)$ in any $A^a$ uniformly for any choice of $a_1,a_2$ with $a_1 \neq a_2$.

To conclude the proof of Theorem 3.1 we must deduce that $[A,\mathbb{B}]$ is expressive. Let $\phi \in L(\Sigma_{A,B})$ and $S \in WP(\Sigma_{A,B})$. Construct the formula $\theta \in L(\Sigma^c_{A,B})$ of Lemma 3.3. Choose variables $z_1$, $z_2$ not in $\phi$, S and $\theta$. By the uniformity property just observed, the formula

$$\exists\, z_1\; z_2[z_1 \neq z_2 \wedge \theta[z_1/c_1,\; z_2/c_2]$$

lies in $L(\Sigma_{A,B})$ and defines $sp_{[A,\mathbb{B}]}(\phi,S)$. $\square$

This method of adding constants can be used to prove that adding finite arithmetics such as modulo n arithmetic (and others discussed in HOARE [5]) preserves expressiveness. Most generally :

3.7 THEOREM *If* A *is expressive and* F *is finite then* [A,F] *is expressive and consequently* HL([A,F]) *is complete.*

## CONCLUDING REMARKS

Quite clearly no useful account of the correctness of many-typed programs can be founded on a first-order assertion language. Fortunately, it is possible to give a very thorough theory of the partial and total correctness of the basic sequential constructs in a many-sorted abstract setting if one allows the extension to a weak second-order assertion language : see [6].

## REFERENCES

[1] APT, K.R., *Ten years of Hoare's Logic : a survey - Part 1*, ACM Transactions Programming Languages and Systems 3 (1981) 431-483.

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

[3] BERGSTRA, J.A. & J.V. TUCKER, *Expressiveness and the completeness of Hoare's Logic*, J. Computer and System Sciences, to appear.

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

[5] HOARE, C.A.R., *An axiomatic basis for computer programming*, Communications ACM 12 (1969) 576-580.

[6] TUCKER, J.V. & J.I. ZUCKER, *Program correctness over abstract data types, with error-state semantics*. Monograph, in preparation.

MC NR

35229

03 D45
03 D80
03 D35
03 D75
69 D24
69 F31