stichting
mathematisch
centrum

AFDELING INFORMATICA IW 224/83 MAART
(DEPARTMENT OF COMPUTER SCIENCE)

J.A. BERGSTRA, A. CMIELIENSKA & J. TIURYN

ANOTHER INCOMPLETENESS THEOREM FOR HOARE'S LOGIC

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, Kruislaan 413, Amsterdam, The Netherlands.

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

1980 Mathematics subject classification 035D35, 03D75, 68B10

1982 CR Categories: D.2.4, F.3.1

Another incompleteness theorem for Hoare's Logic*)

by

J.A. Bergstra, A. Cmielienska**) & J. Tiuryn***)

ABSTRACT

It is known that if Hoare's rules are complete for a first-order structure A then the set of partial correctness assertions true over A is recursive in the first order theory of A. We show that the converse is not true. Namely, there is a first-order structure C such that the set of partial correctness assertions true over C is recursive in the theory of C but Hoare's Logic is not complete for C.

KEY WORDS & PHRASES: *Hoare's Logic*, while-*programs*, *soundness*, *completeness*

---

*) This report will be submitted for publication elsewhere.

**) Mathematical Institute, University of Torun, Poland.

***) Mathematical Institute, University of Warsaw, Poland. The third author was partially supported by the SNF Grant No. MCS8010707, and by a grant to the M.I.T. Laboratory for Computer Science by the I.B.M. Corporation.

INTRODUCTION

A first-order partial correctness assertion is a formula $\{P\}\alpha\{Q\}$, where P and Q are first-order formulae and α is a while-program. The assertion $\{P\}\alpha\{Q\}$ means that if P is true of some machine state, and if the program α halts when started this state, then the formula Q will be true in the halting state of α. Since the set of valid partial correctness assertions is Π^0_2, there is no finitary sound and complete axiom system for partial correctness (see HAREL, MEYER & PRATT [6]).

COOK [4] has shown that the axiom system composed of the rules of HOARE [7] together with the first-order theory of a structure is complete for a certain class of structures. More precisely, for any first-order structure A the system HL(A) consists of Hoare's inference rules together with the first-order theory of A. The structure A is expressive if, for any while-program α and first-order formula P, the strongest postcondition of α with respect to P, $sp(P,\alpha) = \{\vec{a} \in |A|^\omega$: there exists $\vec{b} \in |A|^\omega$ such that α with input $\vec{b}$ terminates in A with output $\vec{a}$ and $A \models P[\vec{b}]\}$ is first-order definable in A. Cook's theorem states that if A is expressive, then HL(A) is complete. In general, however, the set PC(A) of partial correctness assertions that are valid over A is Π^0_1, i.e. co-r.e., in the first-order theory Th(A) of A (cf. BERGSTRA & TUCKER [1]) whereas HL(A) is Σ^0, i.e. r.e. in Th(A). Thus HL(A) is not complete for arbitrary A.

Although expressiveness is sufficient to guarantee the completeness of HL(A), it is not a necessary condition. For example, any nonstandard model of the integers has a complete Hoare logic, but cannot be expressive (see BERSTRA & TUCKER [2]). Moreover, proving properties of programs over expressive structures may be considered a degenerate case. When expressiveness holds, partial correctness assertions reduce to first order formulae.

Since PC(A) is always Π^0_1 in Th(A) and HL(A) is Σ^0_1 in Th(A), then if HL(A) is complete the partial correctness theory PC(A) is recursive in Th(A). This paper studies the following question: is HL(A) complete for every structure A such that PC(A) is recursive in TH(A)? We show that it is not. We will present a general construction of counterexamples for this situation. A corollary of our construction is that the ability to code finite sequences cannot be removed from the hypothesis of Harel's completeness theorem for arithmetical universes (HAREL [5]).

As pointed out in HAREL [5], any structure A can be expanded to a structure with a complete Hoare logic by expanding to an arithmetical universe. This expansion may increase the degree of undecidability of the first-order theory of A. However, when HL(A) is incomplete but PC(A) is recursive in Th(A), the structure A may be expanded in a much simpler way to obtain a complete Hoare logic. We may consider proof systems HL(L,E) over a first order theory E in language L. HL(A) is then identified with HL(L,Th(A)), when A is an L-structure. It follows from BERGSTRA & TUCKER [3] that A can be expanded to an L^*-structure A^* with L^*-L being finite, such that for some decidable theory $T \subseteq Th(A^*)$, $PC(A) \subseteq HL(L^*,Th(A) \cup T)$. Thus $Th(A) \cup T$, where T is decidable but formulated in an extended language, contains enough information to derive all of PC(A).

2. PRELIMINARIES

We begin by presenting a version of Hoare's inference rules that suits our purposes. In the following rules, P, Q and R denote first-order formulae, B denotes any quantifier-free first-order formula, t a term, and x a variable. We use Q[t/x] to denote the formula Q with t substituted for all free occurrences of x. Greek letters α and β denote arbitrary <u>while</u>-programs.

(Assignment Rule) $\quad P \supset Q[t/x] \vdash \{P\}x := t\{Q\}$

(Composition Rule) $$\frac{\{P\}\alpha\{Q\},\{Q\}\beta\{R\}}{\{P\}\alpha;\beta\{R\}}$$

(Conditional rule) $$\frac{\{P\wedge B\}\alpha\{Q\},\{P\wedge \neg B\}\beta\{Q\}}{\{P\}\ \underline{\text{if}}\ B\ \underline{\text{then}}\ \alpha\ \underline{\text{else}}\ \beta\ \text{fi}\ \{Q\}}$$

(Iteration rule) $$\frac{P\supset R,\{R\wedge B\}\alpha\{R\},R\wedge \neg B\supset Q}{\{P\}\ \underline{\text{while}}\ B\ \underline{\text{do}}\ \alpha\ \underline{\text{od}}\ \{Q\}}$$

(Oracle axioms) $\quad$ Every $P \in Th(A)$ is an axiom.

In the composition rule, the formula Q is called an intermediate assertion, and in the iteration rule, R is called the loop invariant. Formally, HL(A) denotes the set of all asserted programs $\{P\}\alpha\{Q\}$ provable from Th(A) using the above rules.

The reader may easily verify that the Rule of Consequence,

$$P \supset P_1,\ \ \{P_1\}\alpha\{Q_1\},\ \ Q_1 \supset Q \vdash \{P\}\alpha\{Q\}$$

is a derived rule of HL. Another rule that is easily derived is

$$\{P\}\alpha\{Q\} \vdash \{\exists xP\}\alpha\{\exists xQ\},$$

where x is a variable that does not occur in α. Together, these two rules imply that superfluous free variables may be eliminated from invariants and intermediate assertions of proofs.

LEMMA 1. *Let* X *be the set of all variables occurring free in* P, Q, *or* α. *If* HL(A) *proves* $\{P\}\alpha\{Q\}$, *then there exists a proof of* $\{P\}\alpha\{Q\}$ *in* HL(A) *using only invariants and intermediate assertions with free variables in* X.

The idea of the proof is as follows. Suppose we have a proof of $\{P\}\alpha\{Q\}$ in HL($\mathcal{A}$) and assume that x is free in P or Q but does not occur in α. This proof can be transformed into another proof by quantifying over x in each formula.

We define the disjoint union $\mathcal{A} \oplus \mathcal{B}$ of first-order structures $\mathcal{A}$ and $\mathcal{B}$ in order to state our theorems. Let L_1 and L_2 be two similarity types. Let A and B be unary predicate symbols and $\perp$ a constant symbol, none of which belong to $L_1 \cup L_2$. Let $\mathcal{A}$, $\mathcal{B}$ be L_1-, L_2-structures, respectively. For any integer i and set X let $X \times i$ denote $X \times \{i\}$. Let $L = L_1 \cup L_2 \cup \{A,B,\perp\}$. We define an L-structure $\mathcal{A} \oplus \mathcal{B}$ with carrier $|\mathcal{A}\oplus\mathcal{B}| = |\mathcal{A}|\times 0 \cup |\mathcal{B}|\times 1 \cup \{<2,2>\}$. We interpret A as the characteristic predicate of $|\mathcal{A}|\times 0$, B as the characteristic predicate of $|\mathcal{B}|\times 1$ and $\perp$ as $<2,2>$. We interpret the L_1 (respectively L_2) function symbols as in $\mathcal{A}$ (as in $\mathcal{B}$, respectively), provided all arguments are taken from $|\mathcal{A}|\times 0$ (from $|\mathcal{B}|\times 1$, resp). Otherwise, we take the value of a function to be $\perp$. We interpret L_1 (respectively L_2) predicate symbols as in $\mathcal{A}$ (as in $\mathcal{B}$, resp.), provided all arguments are taken from $|\mathcal{A}|\times 0$ (from $|\mathcal{B}|\times 1$, resp.), and set to be false elsewhere. In particular, if $R \in L_1 \cap L_2$, then either all arguments of R should be taken from $|\mathcal{A}|\times 0$ or all from $|\mathcal{B}|\times 1$.

Clearly a meaningful alternative to this definition would be to use two-sorted structure, but the disjoint union keeps us closer to the standard Hoare formalism.

We are now in position to formulate two general theorems which answer the question posed in the introduction.

THEOREM 1. *For every* A *there is a structure* B *such that* $PC(A \oplus B)$ *is recursive in* $Th(A \oplus B)$.

THEOREM 2. *For every two structures* A *and* B *if* $HL(A)$ *is incomplete, then so is* $HL(A \oplus B)$.

The following corollary, stated as a claim in the introduction, follows immediately from these theorems.

COROLLARY. *There is a structure* C *such that* $PC(C)$ *is recursive in* $Th(C)$ *and* $HL(C)$ *is incomplete.*

PROOF. Take A bo be any structure for which $HL(A)$ is incomplete (cf. [1,8] for examples). Then, according to Theorem 1, exists B such that $PC(A \oplus B)$ is recursive in $Th(A \oplus B)$. Moreover, according to Theorem 2, $HL(A \oplus B)$ is incomplete. Thus we can put $C = A \oplus B$.

This corollary states a result about the actual formal system HL that aims at proving partial correctness facts true in all generality.

In the sense of BERGSTRA & TUCKER [1] (Thm.2.3) it certainly is conceivable that a special purpose logic of partial correctness can be devised for some given structure $\mathfrak{A}$ which is complete even if $HL(\mathfrak{A})$ is incomplete. Indeed that can be done as soon as $PC(\mathfrak{A})$ is recursive in $Th(\mathfrak{A})$. But the artificial logics thus obtained may well be quite unsatisfactory.

Theorem 1 also gives us some insight into Harel's theorem on arithmetical universes (cf. [5]). Let N stand for the standard model of arithmatic. By Theorem 2 we know that for any A with $HL(A)$ incomplete, $HL(A \oplus N)$ is incomplete. Harel's theorem says that if B is a structure which contains the standard model of arithmetic (as a first order definable part of B) and if B has the ability to code finite sequences of elements from $|B|$, then the first order language is expressive for while-programs over $\mathcal{B}$, and therefore HL($\mathcal{B}$) is complete. Since obviously N is first order definable part of $A \oplus N$, but HL($A \oplus N$) is not complete, Harel's encoding assumption is necessary to ensure the completeness of his axioms.

We prove Theorem 1 in Section 3 and Theorem 2 in Section 4.

3. ADDING AN EXPRESSIVE STRUCTURE

This section shows that for any structure A, there is a structure $\mathcal{B}$ such that PC($A \oplus \mathcal{B}$) is recursive in Th($A \oplus \mathcal{B}$). If the domain of A is finite, then $\mathcal{B}$ may be chosen to be any finite structure. Then $A \oplus \mathcal{B}$ is finite and PC($A \oplus \mathcal{B}$) is recursive in Th($A \oplus \mathcal{B}$). When A is infinite, we will define $\mathcal{B}$ to be a copy of A which also has the standard arithmetic operations defined on the elements of its domain. By construction, the first-order theorey of $\mathcal{B}$ will contain both the first-order theory of A and the first-order theory of arithmetic. As a consequence, PC($A \oplus \mathcal{B}$) will be recursive in Th($A \oplus \mathcal{B}$). However, the structure $A \oplus \mathcal{B}$ need not be expressive since there may not be any way to code pairs of elements of $|A|$ in Th($A \oplus \mathcal{B}$).

Assume now that A is finite and its similarity type is L_1. We construct $\mathcal{B}$ so that PC($A \oplus \mathcal{B}$) is recursive in Th($A \oplus \mathcal{B}$) as follows. First, we expand A to an arithmetic universe in the sense of [5]. To do this, we add a defining predicate for "non negative integers" N, arithmetic operations, constants 0 and 1, and in addition, we add a pairing function. The resulting structure, $\mathcal{B}$, has the same domain as A but has a richer similarity type which we denote by L_2. For technical reasons we assume that L_1 and L_2 are disjoint.

It is clear that Th($\mathcal{B}$) is recursive in Th($A \oplus \mathcal{B}$). This follows immediately from the definition of the $\oplus$-construction. Because $\mathcal{B}$ is expressive (being an arithmetical universe), HL($\mathcal{B}$) is complete. Therefore PC($\mathcal{B}$) is recursive in Th($\mathcal{B}$). Thus, it remains to be shown that PC($A \oplus \mathcal{B}$) is many-one reducible to PC($\mathcal{B}$).

We will outline the reduction of PC($A \oplus \mathcal{B}$) to PC($\mathcal{B}$) by showing an effective simulation of computations on $A \oplus \mathcal{B}$ by those on $\mathcal{B}$.

In order to describe a smooth translation of assertions and programs, we introduce an infinite family of new variables: $y_0, y_1, \ldots$. The translation will take a formula with variables $x_0, x_1, \ldots$ to a formula with variables $x_0, y_0, x_1, y_1, \ldots$ with double the number of quantifiers.

Because the structure $\mathcal{B}$ contains in its language names for 0,1 and 2 (since it contains the language of arithmetic) it is natural to identify the elements of $|\mathcal{B}|$ which correspond to these names with the actual numbers 0,1,2. By means of this identification, we can view

$$|\mathcal{A}\oplus\mathcal{B}| = |\mathcal{A}| \times \{0\} \cup |\mathcal{B}| \times \{1\} \cup \{<2,2>\}$$

as a subset of $|\mathcal{B}| \times |\mathcal{B}|$ (recall that $|\mathcal{A}|=|\mathcal{B}|$). In what follows we use the projection functions on the coordinates, π_1 and π_2 on elements of $|\mathcal{B}| \times |\mathcal{B}|$.

We show an effective translation Tr of first order formulae over the language of $\mathcal{A} \oplus \mathcal{B}$ to first order formulae over L_2. The translation will have the property that for every $P(x_1,\ldots,x_n)$ over the language of $\mathcal{A} \oplus \mathcal{B}$, and for all $c_1,\ldots,c_n \in |\mathcal{A}\oplus\mathcal{B}|$,

$$\mathcal{A} \oplus \mathcal{B} \models P(x_1,\ldots,x_n)[c_1,\ldots,c_n]$$

iff

$$\mathcal{B} \models \mathrm{Tr}(P)(x_1,y_1,\ldots,x_n,y_n)[\pi_1(c_1), \pi_2(c_1),\ldots, \pi_1(c_n), \pi_2(c_n)].$$

Because the formal definition of Tr is slightly cumbersome we present its details. We first introduce some notations. For a term t we define $L_1(t)$ to be <u>true</u> if t is a term over language L_1 and <u>false</u> otherwise.

We define Tr inductively. Suppose P is an equation $t = t'$, where t contains the variables $X = \{x_i : i \in J\}$, and t' contains the variables $X' = \{x_i : i \in J'\}$. Then we want Tr(P) to be true iff

(a) both t and t' have values in $|\mathcal{A}| \times 0$ and $t = t'$, or

(b) both t and t' have values in $|\mathcal{B}| \times 1$ and $t = t'$, or

(c) both t and t' yield $\perp$.

Formulae (a) - (c) can be written formally as follows:

(a') $\quad \bigwedge_{i\in J\cup J'} (y_i=0) \wedge L_1(t) \wedge L_1(t') \wedge t = t'$

(b') $\quad \bigwedge_{i\in J\cup J'} (y_i=1) \wedge t = t'$

(c') $[\bigvee_{i\in J}(y_i\neq 0) \vee \neg L_1(t)] \wedge [\bigvee_{i\in J}(y_i\neq 1)] \wedge$

$[\bigvee_{i\in J'}(y_i\neq 0) \vee \neg L_1(t')] \wedge [\bigvee_{i\in J'}(y_i\neq 1)]$.

Suppose P is an atomic formula $R(t_1,\ldots,t_n)$ with $R \in L_1$ and let $X = \{x_i : i \in J\}$ be the set of variables that occur in P. Then Tr(P) is:

$$\{[\bigwedge_{i\in J}(y_i=0) \wedge \bigwedge_{j=1}^{n} L_1(t_j)] \vee \bigwedge_{i\in J}(y_i=1)\} \wedge R(t_1,\ldots,t_n).$$

If P is A(t), then Tr(P) is:

$$\bigwedge_{i\in J}(y_i=0) \wedge L_1(t).$$

The cases for P of the form B(t) or $R(t_1,\ldots t_n)$ with $R \in L_2 - L_1$ are simpler and we omit them.

If P is $P_1 \vee P_2$, then Tr(P) is $Tr(P_1) \vee Tr(P_2)$.
If P has free variables $X = \{x_i : i \in J\}$, then $Tr(\neg P)$ is

$$\bigwedge_{i\in J}(y_i=0 \vee y_i=1 \vee y_i=2) \wedge \neg Tr(P).$$

Finally, $Tr(\exists x_i P)$ is

$$\exists x_i \exists y_i ((y_i=0 \vee y_i=1 \vee y_i=2) \wedge Tr(P)).$$

This concludes the inductive definition of Tr for formulae.

The next step is to extend Tr to programs α over the language of $\mathcal{A}\oplus\mathcal{B}$ so that for all first order formulae P, Q over the language of $\mathcal{A}\oplus\mathcal{B}$

$$(*)\ \mathcal{A}\oplus\mathcal{B} \models \{P\}\alpha\{Q\} \quad \text{iff} \quad \mathcal{B} \models \{Tr(P)\}\,Tr(\alpha)\{Tr(Q)\}.$$

Let α be a program and let $\{x_0,\ldots,x_{n-1}\}$ contain all variables occurring in α. The translation Tr(α) will use variables x_0, y_0, ..., x_{n-1}, y_{n-1} in such a way that the following diagram commutes:

$$\begin{array}{ccc} |\mathcal{A} \oplus \mathcal{B}|^n & \xrightarrow{\quad\alpha\quad} & |\mathcal{A} \oplus \mathcal{B}|^n \\ \big\downarrow {\scriptstyle \langle\pi_1,\pi_2\rangle^n} & & {\scriptstyle \langle\pi_1,\pi_2\rangle^n} \big\downarrow \\ |\mathcal{B}|^{2n} & \xrightarrow{\quad Tr(\alpha)\quad} & |\mathcal{B}|^{2n} \end{array}$$

We first show how to define Tr for assignment statements. Let $x_i := t$ be an assignment statement, where t is a term over the language of $\mathcal{A}\oplus\mathcal{B}$. Let $X = \{x_j : j \in J\}$ be the set of all variables which occur in t.

If t is a variable, say x_j, then $Tr(x_i := t)$ is $x_i := x_j;\ y_i := y_j$. If t is over L_1 and not a variable, then $Tr(x_i := t)$ is:

<u>if</u> $\bigwedge_{j\in J} y_i = 0$ <u>then</u> $x_i := t;\ y_i := 0$ <u>else</u> $x_i := 2;\ y_i := 2$ <u>fi</u>.

If t is over L_2 and not a variable, then $Tr(x_i := t)$ is

<u>if</u> $\bigwedge_{j\in J} y_i = 1$ <u>then</u> $x_i := t;\ y_i := 1$ <u>else</u> $x_i := 2;\ y_i := 2$ <u>fi</u>.

In all remaining cases $TR(x_i := t)$ is $x_i := 2;\ y_i := 2$.

$Tr(\alpha)$ is a program in which every assignment statement $x_i := t$ in α is replaced by $TR(x_i := t)$, and every test P in α is replaced by $Tr(P)$.

It follows from $(^*)$ that Tr is many-one reduction of $PC(\mathcal{A}\oplus\mathcal{B})$ to $PC(\mathcal{B})$. This completes the proof of Theorem 1.

4. HOARE'S LOGIC OVER DIRECT SUMS

In this section we will show that incompleteness of $HL(\mathcal{A})$ implies incompleteness of $HL(\mathcal{A}\oplus\mathcal{B})$.

Let P be a first order formula over the language of $\mathcal{A}\oplus\mathcal{B}$. We define P_A, a relativisation of P to $|\mathcal{A}|$, inductively as follows.

(i) if P is atomic, then P_A is P

(ii) $(\neg P)_A$ is $\neg(P_A)$

(iii) $(P \vee Q)_A$ is $P_A \vee Q_A$

(iv) $(\exists x P)_A$ is $\exists x(A(x) \wedge P_A)$.

If X is a finite set of variables, then A(X) denotes $\bigwedge_{x \in X} A(x)$. We define P_B and B(X) similarly.

Using relativised formulae, we can interpret PC($\mathcal{A}$) in PC($\mathcal{A}\oplus\mathcal{B}$).

<u>LEMMA 2</u>. *Let* P, Q *be first order formulae over* L_1, *and let* α *be a* <u>while</u>-*program over* L_1. *Let* X *be the set of all variables occurring free in* P *or* Q *or* α. *Then*

$$\mathcal{A} \models \{P\}\alpha\{Q\} \text{ iff } \mathcal{A}\oplus\mathcal{B} \models \{A(X) \wedge P_A\}\alpha\{A(X) \wedge Q_A\}.$$

Furthermore, if HL($\mathcal{A}\oplus\mathcal{B}$) *proves* $\{A(X) \wedge P_A\}\,\alpha\,\{A(x) \wedge Q_A\}$ *using only invariants and intermediate assertions with free variables in* X *and of the form* $A(X) \wedge R_A$, *then* HL($\mathcal{A}$) *proves* $\{P\}\,\alpha\,\{Q\}$.

The proof of this Lemma is straightforward and is omitted. To finish the proof of Theorem 2, we need the following position.

<u>PROPOSITION 3</u>. *Let* P *be a first order formula over the language of* $\mathcal{A}\oplus\mathcal{B}$ *and let* X *be the set of all variables occurring free in P. Then there exists a first order formula* Q *over* L_1 *such that*

$$\mathcal{A}\oplus\mathcal{B} \models (A(X) \wedge P) \equiv (A(X) \wedge Q_A).$$

Before we prove Proposition 3, we show how it yields the proof of Theorem 2.

<u>PROOF OF THEOREM 2</u>. Assume that HL($\mathcal{A}$) is incomplete and HL($\mathcal{A}\oplus\mathcal{B}$) complete. Choose {P}α{Q} true in $\mathcal{A}$ but not derivable in HL($\mathcal{A}$). Let X be the set of all variables occurring free in P, Q, or α. By Lemma 2, $\{A(X) \wedge P_A\}\,\alpha\,\{A(X) \wedge Q_A\}$ is true in $\mathcal{A}\oplus\mathcal{B}$, and therefore HL($\mathcal{A}\oplus\mathcal{B}$) $\vdash \{A(X) \wedge P_A\}\,\alpha\,\{A(X) \wedge Q_A\}$.

We derive a contradiction by constructing a proof of {P}α{Q} in HL($\mathcal{A}$). By Lemma 1, there is a proof of $\{A(X) \wedge P\}\,\alpha\,\{A(X) \wedge Q\}$ in which all intermediate assertions and invariants have their free variables in X. In addition, each {R}α{S} in the proof may be replaced by $\{A(X) \wedge R\}\,\alpha\,\{A(X) \wedge S\}$ to yield another valid proof. Then, according to Proposition 3, all invariants and intermediate assertions can be written in the form

$\{A(X) \wedge R'_A\}\ \alpha\ \{A(X) \wedge S'_A\}$ with R' and S' are first order formulae over L_1. By Lemma 2, HL(A) proves $\{P\}\alpha\{Q\}$ in contrast to our assumptions. □

The proof of Proposition 3 uses Lemmas 4-6 stated below.

LEMMA 4. *($\bot$-elimination) For every first order formula* P *over* $L_1 \cup \{A,B,\bot\}$ *there is a formula* $P^\bot$ *over* $L_1 \cup \{A,B\}$ *such that*

(i) $A+B \models P \equiv P^\bot$

(ii) $A+B \models (P^\bot)_A \equiv (P_A)^\bot$.

PROOF. It suffices to notice that we can define the constant $\bot$ using the unary relations A and B: $x = \bot$ iff $\neg A(x) \wedge \neg B(x)$. □

We say that a formula P of the language of $A \oplus B$ is normalised iff there is a number n, formulae $F^1,\ldots,F^n$ over $L_1 \cup \{A,B\}$ and formulae $G^1,\ldots,G^n$ over $L_2 \cup \{A,B\}$ such that P is of the form $\bigwedge_{i=1}^{1}(F^i_A \vee G^i_B)$.

LEMMA 5. *Let* P *be a formula over* $L_2 \cup \{A,B\}$. *There exists a normalised formula* Q *of the form* $\bigwedge_{i=1}^{n}(F^i_A \vee G^i_B)$ *such that* $A \oplus B \models A(x) \supset (P_B \equiv Q)$ *and* x *is not free in* G^i_B, i = 1,...n. *Moreover all variables free in* Q *are free in* P.

PROOF. Let us consider first the case when formula P_B is atomic. If P_B is over $L_1 \cup \{A,B\}$ then Q can be $P_B \vee$ false. If x does not occur in P_B as free variable, then Q can be false $\vee P_B$. If P_B is not over $L_1 \cup \{A,B\}$, contains x as a free variable and is of the form $R(t_1,\ldots)$ then A(x) implies $P_B \equiv$ false. The remaining subcase is a formula of the form $t_1 = t_2$, not over $L_1 \cup \{A,B\}$, and containing x as a free variable. Then A(x) implies $P_B \equiv t_1 = t_2 = \bot$, which means that P_B is equivalent to a propositional combination of clauses of the form A(y) and B(y).

If P_B is not atomic, then we transform it to the desired form in four steps. Steps 2 and 3 should be skipped in case when P_B is quantifier free.

STEP 1. Replace all atomic subformulae of P_B containing x as a free variable and not over $L_1 \cup \{A,B\}$ by false or by a combination of clauses of the form A(x) and B(x), according to the previous reasoning.

STEP 2. Replace each atomic subformula containing both x as a free variable and at least one occurrence of a bound variable. Since every bound variable y of P_B is assumed to fulfill B(y), we again can replace such subformula by false if it is a relation, and by a combination of A and B clauses if it is term equality.

STEP 3. Transform P_B in such a way, that no subformula containing x as a free variable is in the range of any quantifier, and the set of all subformulae is unchanged (we can do it, because due to step 2 no such atomic subformula of contains any bound variable).

STEP 4. Use the laws of distributivity and the de Morgan's rule to transform P_B to the form $\bigwedge_{i=1}^{n}(F^i \vee G_B^i)$ such that F's are created from exactly these atomic subformulae in which x occurs as a free variable.

Due to the steps 1, 2 and 3 formulae F's are over $L_1 \cup \{A\}$, moreover they are quantifier free (this is what assures that F is equal to F_A). Since all atomic subformulae introduced in the transformation are of the form A(y) or B(x), the new P_B is still over $L_2 \cup \{A,B\}$. Moreover, no new variable has been introduced. Thus the new P_B is of the desired form. □

We observe that due to the symmetry of the construction of $\mathcal{A}\oplus\mathcal{B}$, Lemmas 4 and 5 are true when L_1 is interchanged with L_2 and A with B.

LEMMA 6. *For every formula* P *of the language of* $\mathcal{A}\oplus\mathcal{B}$ *there is a normalised formula* Q *such that* $\mathcal{A}\oplus\mathcal{B} \models P \equiv Q$.

PROOF. The proof is by induction on P. In the basis case, if P is over $L_1 \cup \{A,B\}$ (resp. $L_2 \cup \{A,B\}$) then Q can be P ∨ false (resp. false ∨ P). In the remaining case if P is of the form $R(t_1,\ldots)$ then it is equivalent in $\mathcal{A}\oplus\mathcal{B}$ to false, and if it is of the form $t_1 = t_2$ then it is equivalent to $t_1 = t_2 = \perp$. The latter is equivalent in $\mathcal{A}\oplus\mathcal{B}$ to a formula over {A,B}.

The only nontrivial case in the inductive step is for P of the form ∀x Q. We assume inductively, that over $\mathcal{A}\oplus\mathcal{B}$ the formula Q is equivalent to a normalised Q', where Q' is of the form $\bigwedge_{i=1}^{n}(F_A^i \vee G_B^i)$.

Since

$$\mathcal{A}\oplus\mathcal{B} \models P \equiv \bigwedge_{i=1}^{n} \forall x(F_A^i \vee G_B^i)$$

it is enough to show a transformation of every formula $\forall x(F^i_A \vee G^i_B)$ for $i = 1,\ldots, n$ into a formula of the desired form. First we observe that such a formula is equivalent over $\mathcal{A}\oplus\mathcal{B}$ to the conjunction of the formulae

(aa) $\quad F^i_A(\bot/x) \vee G^i_B)(\bot/x)$

(bb) $\quad \forall x[A(x) \supset (F^i_A \vee G^i_B)]$

(cc) $\quad \forall x[B(x) \supset (F^i_A \vee G^i_B)]$.

Using Lemma 4 we convert the (aa) into an equivalent formula of the desired form. The transformations of (bb) and (cc) are similar and we present here only a transformation of (bb).

Using Lemma 5, we can replace G^i_B in (bb):

(bb') $\quad \forall x[A(x) \supset (F^i_A \vee \bigwedge_{j=1}^{m}(H^j_A \vee J^j_B))]$.

Since x does not occur free in J^j_B, $j = 1,\ldots,n$, (bb') is equivalent over $\mathcal{A}\oplus\mathcal{B}$ to

(bb') $\quad (\bigwedge_{j=1}^{m}[(\forall x(F^i \vee H^j))_A \vee J^j_B]$.

Because we assumed that F's and H's are over $L_1 \cup \{A,B\}$ and J's are over $L_2 \cup \{A,B\}$, that last formula is normalised. This completes the proof of the lemma. □

We can now prove Proposition 3.

PROOF OF PROPOSITION 3. Let P be a first order formula over the language of $\mathcal{A}\oplus\mathcal{B}$. By Lemma 6 it is equivalent over $\mathcal{A}\oplus\mathcal{B}$ to $\bigwedge_{i=1}^{n}(F^i_A \vee G^i_B)$, where F_i's are over $L_1 \cup \{A,B\}$ and G_i's are over $L_2 \cup \{A,B\}$.

Let X be the set of all variables which occur free in P. Using Lemma 5 repeatedly for every variable from X we can get a normalised formula Q" of the form $\bigwedge_{i=1}^{m}(K^i_A \vee L^i_B)$ such that

$$\mathcal{A}\oplus\mathcal{B} \models A(X) \wedge P \equiv A(X) \wedge Q''$$

and in L^i_B no free occur. Let ϵ_i be <u>true</u> if $\mathcal{A}\oplus\mathcal{B} \models A(X) \wedge L^i_B$, and <u>false</u> otherwise.

Clearly

$$\mathcal{A}\oplus\mathcal{B} \models A(X) \wedge P \equiv A(X) \wedge \bigwedge_{i=1}^{m} (K^i_A \vee \epsilon_i).$$

Let the formula $\bigwedge_{i=1}^{m} (K^i_A \vee \epsilon_i)$ be called Q' and let Q be obtained from Q' by replacing subformulae of the form A(t) by <u>true</u>, and subformulae of the form B(t) by <u>false</u>. It is easy to check that such a Q fulfills our requirements. □

<u>CONCLUDING REMARKS</u>.

Because HL is a very natural system, classifying the structures for which HL is complete and for which it is incomplete is an interesting issue. Previous examples of structures with an incomplete HL such as in BERGSTRA & TUCKER [1] and WAND [8] share the property that $PC(\mathfrak{A})$ is not recursive in $Th(\mathfrak{A})$ (from which incompleteness of $HL(\mathfrak{A})$ immediately follows). Our corollary yields a different example. Further it is worthwhile to find generalized logics of partial correctness even for the simple case of <u>while</u>-programs.

We would like to state some question based on the following simple definitions.

<u>DEFINITION 1</u>. $\mathfrak{A}$ is PC-compact if for each asserted program {p}S{q} true in $\mathfrak{A}$ there is a sentence $\phi \in L(\mathfrak{A})$ such that $Mod(\phi) \models \{p\}S\{q\}$.
2. A logic of partial correctness $LPC(\mathfrak{A})$ for $\mathfrak{A}$ is an r.e. set of pairs $< \phi_i, \{p_i\}S_i\{q_i\} >$ wiht $\phi_i \in L(\mathfrak{A})$ and $\{p_i\}S_i\{q_i\}$ an asserted program over $L(\mathfrak{A})$. $LPC(\mathfrak{A})$ is *sound* if for all $< \phi, \{p\}S\{q\} > \in LPC(\mathfrak{A})$, $Mod(\phi) \models \{p\}S\{q\}$. $LPC(\mathfrak{A})$ is *complete* if whenever $\mathfrak{A} \models \{p\}S\{q\}$ there is $\phi \in Th(\mathfrak{A})$ such that $< \phi, \{p\}S\{q\} > \in LPC(\mathfrak{A})$.
3. $\mathfrak{A}$ is PC-complete if there exists a sound and complete logic $LPC(\mathfrak{A})$ for $\mathfrak{A}$.

It is easily seen that if $HL(\mathfrak{A})$ is complete $\mathfrak{A}$ is PC-complete, and that PC-completeness implies PC-compactness. Moreover, if $\mathfrak{A}$ is PC-complete $PC(\mathfrak{A})$ is recursive in $Th(\mathfrak{A})$.

QUESTIONS

(i) If $\mathfrak{A}$ is computable and HL($\mathfrak{A}$) is complete must $\mathfrak{A}$ be expressive?

(ii) If PC($\mathfrak{A}$) is recursive in $\mathfrak{A}$ and $\mathfrak{A}$ is PC-compact must HL($\mathfrak{A}$) be complete?

(iii) If the answer in (ii) is negative , must $\mathfrak{A}$ be PC-complete?

(iv) Is there a sound LPC which is complete for all PC-complete structures of a given signature?

ACKNOWLEDGEMENT

We thank Piotr Berman and an anonymous referee for criticism and suggestions concerning this paper.

REFERENCES

[1] BERGSTRA, J.A. & J.V. TUCKER, *Some Natural Structures Which Fail To Possess a Sound and Decidable Hoare-like Logic for their While-Programs.* Theoret. Comp. Sci. 17,3 (March 1982), pp. 235-350.

[2] BERGSTRA, J.A. & J.V. TUCKER, *Expressiveness and the completeness of Hoare's logic:* To appear in J.C.S.S.

[3] BERGSTRA, J.A. & J.V. TUCKER, *Two Theorems on the Completeness of Hoare's Logic:* To appear in Inf. Proc. Letters.

[4] COOK, S.A., *Soundness and Completeness of an Axiom System for Program verification:* SIAM J. Computing 7 (1978), pp. 129-147.

[5] HAREL, D., *First-Order Dynamic Logic.* Springer LNCS 68, 1979.

[6] HAREL, D., A.R. MEYER & V. PRATT, *Computability and Completeness in Logics of Programs: Preliminary Report.* 9-th ACM Symposium on Theory of Computing, Boulder, Colorado, May, 1977, pp. 261-268. Revised version, M.I.T. Lab. for Computer Science TM-97, (Feb. 1978) 16 pp.

[7] HOARE, C.A.R., *An Axiomatic Basis for Computer Programming,* CACM 12, 10 (1969), pp. 576-580.

[8] WAND, M., *A new Incompleteness Result for Hoare's System,* J. ACM 25, 1 (Jan. 1978), pp. 168-175.

69 D 24
69 F 31