INFORMATION AND CONTROL 52, 159–171 (1982)

Another Incompleteness Result for Hoare's Logic

JAN A. BERGSTRA*

Department of Computer Science,
University of Leiden, Leiden, The Netherlands

ANNA CHMIELINSKA

Mathematical Institute, University of Torun, Poland

AND

JERZY TIURYN†

Mathematical Institute, University of Warsaw, Warszawa, Poland

It is known (Bergstra and Tucker (1982) *J. Comput. System Sci.* **25**, 217) that if the Hoare rules are complete for a first-order structure $\mathscr{A}$, then the set of partial correctness assertions true over $\mathscr{A}$ is recursive in the first-order theory of $\mathscr{A}$. We show that the converse is not true. Namely, there is a first-order structure $\mathscr{C}$ such that the set of partial correctness assertions true over $\mathscr{C}$ is recursive in the theory of $\mathscr{C}$, but the Hoare rules are not complete for $\mathscr{C}$.

1. 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 from this state, then the formula Q will be true in the halting state of α. Since the set of valid partial correctness assertion is Π_2^0-complete [6], there is no finitary sound and complete axiom system for partial correctness.

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 $\mathscr{A}$, the system HL($\mathscr{A}$) consists of Hoare's inference rules together with the first-order theory of $\mathscr{A}$. The structure $\mathscr{A}$ is *expressive* if, for any *while*-program α

* Jan A. Bergstra's present address is the Department of Computer Science, Mathematical Centre, Amsterdam, The Netherlands.

† Jerzy Tiuryn was partially supported by NSF Grant MCS 8010707, and by a grant to the M.I.T. Laboratory for Computer Science by the IBM Corporation.

0019-9958/82 $2.00

and first-order formula P, the strongest postcondition of α with respect to P, $\mathrm{sp}(P, \alpha) = \{\mathbf{a} \in |\mathscr{A}|^\omega$: there exists $\mathbf{b} \in |\mathscr{A}|^\omega$ such that α with input $\mathbf{b}$ terminates in $\mathscr{A}$ with output $\mathbf{a}$ and $\mathscr{A} \models P[\mathbf{b}]\}$ is first-order definable in $\mathscr{A}$. Cook's theorem states that if $\mathscr{A}$ is expressive, then $\mathrm{HL}(\mathscr{A})$ is complete. In general, however, the set $\mathrm{PC}(\mathscr{A})$ of partial correctness assertions that are valid over $\mathscr{A}$ is Π_1^0, i.e. co-r.e., in the first-order theory $\mathrm{Th}(\mathscr{A})$ of $\mathscr{A}$ (cf. [1]) whereas $\mathrm{HL}(\mathscr{A})$ is Σ^0, i.e., r.e. in $\mathrm{Th}(\mathscr{A})$. Thus $\mathrm{HL}(\mathscr{A})$ is not complete for arbitrary $\mathscr{A}$.

Although expressiveness is sufficient to guarantee the completeness of $\mathrm{HL}(\mathscr{A})$, it is not a necessary condition. For example, any nonstandard model of the integers has a complete Hoare logic, but cannot be expressive (cf. [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 $\mathrm{PC}(\mathscr{A})$ is always Π_1^0 in $\mathrm{Th}(\mathscr{A})$ and $\mathrm{HL}(\mathscr{A})$ is Σ_1^0 in $\mathrm{Th}(\mathscr{A})$, then if HL is complete the partial correctness theory $\mathrm{PC}(\mathscr{A})$ is recursive in $\mathrm{Th}(\mathscr{A})$. This paper studies the following question: *is* $\mathrm{HL}(\mathscr{A})$ *complete for every structure* $\mathscr{A}$ *such that* $\mathrm{PC}(\mathscr{A})$ *is recursive in* $\mathrm{Th}(\mathscr{A})$? We show that it is not. We will present a general construction of counterexamples for this situation. A corollary or our construction is that the ability to code finite sequence cannot be removed from the hypothesis of Harel's completeness theorem for arithmetical universes (cf. [5]).

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

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. Letters α and β denote arbitrary *while*-programs.

Assignment Rule	$P \supset Q[t/x] \vdash \{P\}x := t\{Q\}$
Composition Rule	$\{P\}\,\alpha\,\{Q\}, \{Q\}\,\beta\,\{R\} \vdash \{P\}\,\alpha;\beta\,\{R\}$
Conditional rule	$\{P \wedge B\}\,\alpha\,\{Q\}, \{P \wedge \neg B\}\,\beta\,\{Q\}$ $\vdash \{P\}$ *if* B *then* α *else* β *fi* $\{Q\}$
Iteration rule	$P \supset R, \{R \wedge B\}\,\alpha\,\{R\}, R \wedge \neg B \supset Q$ $\vdash \{P\}$ *while* B *do* α *od* $\{Q\}$
Oracle axioms	Every $P \in \mathrm{Th}(\mathscr{A})$ 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, $\mathrm{HL}(\mathscr{A})$ denotes the set of all asserted programs $\{P\}\,\alpha\,\{Q\}$ provable from $\mathrm{Th}(\mathscr{A})$ using the above rules.

The reader may easily verify that the *rule of consequence*,

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

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

$$\{P\}\,\alpha\,\{Q\} \vdash \{\exists x P\}\,\alpha\,\{\exists x Q\},$$

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* $\mathrm{HL}(\mathscr{A})$ *proves* $\{P\}\,\alpha\,\{Q\}$, *then there exists a proof of* $\{P\}\,\alpha\,\{Q\}$ *in* $\mathrm{HL}(\mathscr{A})$ *using only invariants and intermediate assertions with free variables in X.*

The idea of the proof is as follows: Suppose we have proof of $\{P\}\,\alpha\,\{Q\}$ in $\mathrm{HL}(\mathscr{A})$ 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 $\mathscr{A} \oplus \mathscr{B}$ of first-order structures $\mathscr{A}$ and $\mathscr{B}$ in order to state our theorems. Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two similarity types. Let A and B be unary predicate symbols and $\perp$ a constant symbol, none of which belong to $\mathscr{L}_1 \cup \mathscr{L}_2$. Let $\mathscr{A}$, $\mathscr{B}$ be $\mathscr{L}_1$-, $\mathscr{L}_2$-structures, respectively. For any integer i and set X let $X \times i$ denote $X \times \{i\}$. Let $\mathscr{L} = \mathscr{L}_1 \cup \mathscr{L}_2 \cup \{A, B, \perp\}$. We define an $\mathscr{L}$-structure $\mathscr{A} \oplus \mathscr{B}$ with carrier $|\mathscr{A} \oplus \mathscr{B}| = |\mathscr{A}| \times 0 \cup |\mathscr{B}| \times 1 \cup \{\langle 2, 2\rangle\}$. We interpret A as the characteristic predicate of $|\mathscr{A}| \times 0$, B as the characteristic predicate of $|\mathscr{B}| \times 1$ and $\perp$ as $\langle 2, 2\rangle$. We interpret the $\mathscr{L}_1$ (resp. $\mathscr{L}_2$) function symbols as in $\mathscr{A}$ (as in $\mathscr{B}$, resp.), provided all arguments

are taken from $|\mathscr{A}| \times 0$ (from $|\mathscr{B}| \times 1$, resp.) Otherwise, we take the value of a function to be $\perp$. We interpret $\mathscr{L}_1$ (resp. $\mathscr{L}_2$) predicate symbols as in $\mathscr{A}$ (as in $\mathscr{B}$, resp.), provided all arguments are taken from $|\mathscr{A}| \times 0$ (from $|\mathscr{B}| \times 1$, resp.), and set to be false elsewhere. In particular, if $R \in \mathscr{L}_1 \cap \mathscr{L}_2$, then either all arguments of R should be taken from $|\mathscr{A}| \times 0$ or all from $|\mathscr{B}| \times 1$.

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 $\mathscr{A}$ there is a structure $\mathscr{B}$ such that* $\mathrm{PC}(\mathscr{A} \oplus \mathscr{B})$ *is recursive in* $\mathrm{Th}(\mathscr{A} \oplus \mathscr{B})$.

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

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

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

Proof. Take $\mathscr{A}$ to be any structure for which $\mathrm{HL}(\mathscr{A})$ is incomplete (cf. [1, 8], for examples). Then, according to Theorem 1, exists $\mathscr{B}$ such that $\mathrm{PC}(\mathscr{A} \oplus \mathscr{B})$ is recursive in $\mathrm{Th}(\mathscr{A} \oplus \mathscr{B})$. Moreover, according to Theorem 2, $\mathrm{HL}(\mathscr{A} \oplus \mathscr{B})$ is incomplete. Thus we can put $\mathscr{C} = \mathscr{A} \oplus \mathscr{B}$. 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 [1, Theorem 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 $\mathrm{HL}(\mathfrak{A})$ is incomplete. Indeed that can be done as soon as $\mathrm{PC}(\mathfrak{A})$ is recursive in $\mathrm{Th}(\mathfrak{A})$. 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 arithmetic. By Theorem 2 we know that for any $\mathscr{A}$ with $\mathrm{HL}(\mathscr{A})$ incomplete, $\mathrm{HL}(\mathscr{A} \oplus N)$ is incomplete. Harel's theorem says that if $\mathscr{B}$ is a structure which contains the standard model of arithmetic (as a first-order definable part of $\mathscr{B}$) and if $\mathscr{B}$ has the ability to code finite sequences of elements from $|\mathscr{B}|$, then the first-order language is expressive for *while*-programs over $\mathscr{B}$, and therefore $\mathrm{HL}(\mathscr{B})$ is complete. Since obviously N is a first-order definable part of

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

Assume now that $\mathscr{A}$ is infinite and its similarity type is $\mathscr{L}_1$. We construct $\mathscr{B}$ so that $\mathrm{PC}(\mathscr{A} \oplus \mathscr{B})$ is recursive in $\mathrm{Th}(\mathscr{A} \oplus \mathscr{B})$ as follows. First, we expand $\mathscr{A}$ to an arithmetic universe in the sense of [5]. To do this, we add a defining predicate for "nonnegative integers" N, arithmetic operations, constants 0 and 1, and in addition, we add a pairing function. The resulting structure $\mathscr{B}$ has the same domain as $\mathscr{A}$ but has a richer similarity type which we denote by $\mathscr{L}_2$. For technical reasons we assume that $\mathscr{L}_1$ and $\mathscr{L}_2$ are disjoint.

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

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

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

$$|\mathscr{A} \oplus \mathscr{B}| = |\mathscr{A}| \times \{0\} \cup |\mathscr{B}| \times \{1\} \cup \{\langle 2, 2\rangle\}$$

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

We show an effective translation Tr of first-order formulae over the language of $\mathscr{A} \oplus \mathscr{B}$ to first-order formulae over $\mathscr{L}_2$. The translation will have the property that for every $P(x_1,..., x_n)$ over the language of $\mathscr{A} \oplus \mathscr{B}$, and for all $c_1,..., c_n \in |\mathscr{A} \oplus \mathscr{B}|$,

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

iff

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

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 *true* if t is a term over language $\mathscr{L}_1$ and *false*, 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 $\mathrm{Tr}(P)$ to be true iff

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

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

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

Formulas (a)–(c) can be written formally as

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

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

(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 $R(t_1,..., t_n)$ with $R \in \mathscr{L}_1$ and let $X = \{x_i : i \in J\}$ be the set of variables that occur in P. Then $\mathrm{Tr}(P)$ is

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

If P is $A(t)$, then $\mathrm{Tr}(P)$ is

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

The cases for P of the form $B(t)$ or $R(t_1,..., t_n)$ with $R \in \mathscr{L}_2 - \mathscr{L}_1$ are simpler and we omit them.

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

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

Finally, $\mathrm{Tr}(\exists x_i P)$ is

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

This concludes the inductive definition of Tr for formulae.

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

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

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

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

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 $\mathscr{A} \oplus \mathscr{B}$. 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 $\mathrm{Tr}(x_i := t)$ is $x_i := x_j;\ y_i := y_j$. If t is over $\mathscr{L}_1$ and not a variable, then $\mathrm{Tr}(x_i := t)$ is

$$\textit{if} \bigwedge_{j \in J} y_i = 0 \textit{ then } x_i := t;\ y_i := 0, \textit{ else } x_i := 2;\ y_i := 2 \textit{ fi}.$$

If t is over $\mathscr{L}_2$ and not a variable, then $\mathrm{Tr}(x_i := t)$ is

$$\textit{if} \bigwedge_{j \in J} y_i = 1, \textit{ then } x_i := t;\ y_i := 1, \textit{ else } x_i := 2;\ y_i := 2 \textit{ fi}.$$

In all remaining cases $\mathrm{Tr}(x_i := t)$ is $x_i := 2;\ y_i := 2$.

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

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

4. Hoare's Logic over Direct Sums

In this section we will show that incompleteness of HL($\mathscr{A}$) implies incompleteness of HL($\mathscr{A} \oplus \mathscr{B}$). Let P be a first-order formula over the language of $\mathscr{A} \oplus \mathscr{B}$. We define P_A, a relativisation of P to $|\mathscr{A}|$, 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($\mathscr{A}$) in PC($\mathscr{A} \oplus \mathscr{B}$).

Lemma 2. *Let P, Q be first-order formulae over $\mathscr{L}_1$, and let α be a* while*-program over $\mathscr{L}_1$. Let X be the set of all variables occurring free in P or Q or α. Then*

$$\mathscr{A} \models \{P\}\,\alpha\,\{Q\} \qquad \textit{iff} \quad \mathscr{A} \oplus \mathscr{B} \models \{A(X) \wedge P_A\}\,\alpha\,\{A(X) \wedge Q_A\}.$$

Furthermore, if HL($\mathscr{A} \oplus \mathscr{B}$) *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($\mathscr{A}$) *proves* $\{P\}\,\alpha\,\{Q\}$.

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

Proposition 3. *Let P be a first-order formula over the language of $\mathscr{A} \oplus \mathscr{B}$ and let X be the set of all variables occurring free in P. Then there exists a first-order formula Q over $\mathscr{L}_1$ such that*

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

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

Proof of Theorem 2. Assume that HL($\mathscr{A}$) is incomplete and HL($\mathscr{A} \oplus \mathscr{B}$) complete. Choose $\{P\}\,\alpha\,\{Q\}$ true in $\mathscr{A}$ but not derivable in HL($\mathscr{A}$). 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 $\mathscr{A} \oplus \mathscr{B}$, and therefore HL($\mathscr{A} \oplus \mathscr{B}$) $\vdash \{A(X) \wedge P_A\}\,\alpha\,\{A(X) \wedge Q_A\}$.

We derive a contradiction by constructing a proof of $\{P\}\,\alpha\,\{Q\}$ in HL($\mathscr{A}$). 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 $\{R\}\,\alpha\,\{S\}$ in the proof may be replaced by $\{A(X) \wedge R\}\,\alpha\,\{(X) \wedge S\}$ 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 formulas over $\mathscr{L}_1$. By Lemma 2, HL($\mathscr{A}$) proves $\{P\}\,\alpha\,\{Q\}$ in contrast to our assumptions. ∎

The proof of Proposition 3 uses Lemmas 4–6.

LEMMA 4 ($\perp$-elimination). *For every first-order formula P over $\mathscr{L}_1 \cup \{A, B, \perp\}$ there is a formula $P^{\perp}$ over $\mathscr{L}_1 \cup \{A, B\}$ such that*

(i) $\mathscr{A} \oplus \mathscr{B} \models P \equiv P^{\perp}$

(ii) $\mathscr{A} \oplus \mathscr{B} \models (P^{\perp})_A \equiv (P_A)^{\perp}$.

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

We say that a formula P of the language of $\mathscr{A} \oplus \mathscr{B}$ is normalised iff there is a number n, formulas $F^1,\ldots, F^n$ over $\mathscr{L}_1 \cup \{A, B\}$ and formulas $G^1,\ldots, G^n$ over $\mathscr{L}_2 \cup \{A, B\}$ such that P is of the form $\bigwedge_{i=1}^{n}(F^i_A \vee G^i_B)$.

LEMMA 5. *Let P be a formula over $\mathscr{L}_2 \cup \{A, B\}$. There exists a normalised formula Q of the form $\bigwedge_{i=1}^{n}(F^i_A \vee G^i_B)$ such that $\mathscr{A} \oplus \mathscr{B} \models A(x) \supset (P_B \equiv Q)$ and x is not free in G^i_B, $i = 1,\ldots, n$. 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 $\mathscr{L}_1 \cup \{A, B\}$, then Q can be $P_B \vee$ *false*. If x does not occur in P_B as a free variable, then Q can be *false* $\vee P_B$. If P_B is not over $\mathscr{L}_1 \cup \{A, B\}$, 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 $\mathscr{L}_1 \cup \{A, B\}$, and containing x as a free variable. Then $A(x)$ implies $P_B \equiv t_1 = t_2 = \perp$, 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 the case where P_B is quantifier free.

Step 1. Replace all atomic subformulas of P_B containing x as a free variable and not over $\mathscr{L}_1 \cup \{A, B\}$ 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 subformulas is unchanged (we can do it, because due to step 2 no such atomic subformula 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 subformulas in which x occurs as a free variable.

Due to steps 1–3 formulas F's are over $\mathscr{L}_1 \cup \{A\}$, moreover they are quantifier free (this is what assures that F is equal to F_A). Since all atomic subformulas introduced in the transformation are of the form $A(y)$ or $B(x)$, the new P_B is still over $\mathscr{L}_2 \cup \{A, B\}$. 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 $\mathscr{A} \oplus \mathscr{B}$, Lemmas 4 and 5 are true when $\mathscr{L}_1$ is interchanged with $\mathscr{L}_2$ and A with B.

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

Proof. The proof is by induction on P. In the basis case, if P is over $\mathscr{L}_1 \cup \{A, B\}$ (resp. $\mathscr{L}_2 \cup \{A, B\}$), then Q can be $P \vee$ *false* (resp. *false* $\vee P$). In the remaining case, if P is of the form $R(t_1,...)$, then it is equivalent in $\mathscr{A} \oplus \mathscr{B}$ 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 $\mathscr{A} \oplus \mathscr{B}$ to a formula over $\{A, B\}$.

The only nontrivial case in the inductive step is for P of the form $\forall x\, Q$. We assume inductively, that over $\mathscr{A} \oplus \mathscr{B}$ 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

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

it is enough to show a transformation of every formula $\forall x\ (F_A^i \vee G_B^i)$ for $i = 1,..., n$ into a formula of the desired form. First we observe that such a formula is equivalent over $\mathscr{A} \oplus \mathscr{B}$ to the conjunction of the formulas

(aa) $F_A^i(\perp/x) \vee G_B^i)(\perp/x)$,

(bb) $\forall x\ [A(x) \supset (F_A^i \vee G_B^i)]$,

(cc) $\forall x\ [B(x) \supset (F_A^i \vee G_B^i)]$.

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_B^i in (bb):

$$(\mathrm{bb}') \quad \forall x \quad [A(x) \supset (F_A^i \vee \bigwedge_{j=1}^m (H_A^j \vee J_B^j))].$$

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

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

Because we assumed that F's and H's are over $\mathscr{L}_1 \cup \{A, B\}$ and J's are over $\mathscr{L}_2 \cup \{A, B\}$, the last formula is normalised. This completes the proof of the lemma. ∎

We can now prove Proposition 3.

Proof. Let P be a first-order formula over the language of $\mathscr{A} \oplus \mathscr{B}$. By Lemma 6 it is equivalent over $\mathscr{A} \oplus \mathscr{B}$ to $\bigwedge_{i=1}^n (F_A^i \vee G_B^i)$, where F_i's are over $\mathscr{L}_1 \cup \{A, B\}$ and G_i's are over $\mathscr{L}_2 \cup \{A, B\}$. 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 $\bigwedge_{i=1}^m (K_A^i \vee L_B^i)$ such that

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

and in L_B^i no free variables occur. Let ε_i be *true* if $\mathscr{A} \oplus \mathscr{B} \models A(X) \wedge L_B^i$, and *false*, otherwise.

Clearly

$$\mathscr{A} \oplus \mathscr{B} \models A(X) \wedge P \equiv A(X) \wedge \bigwedge_{i=1}^m (K_A^i \vee \varepsilon_i).$$

Let the formula $\bigwedge_{i=1}^m (K_A^i \vee \varepsilon_i)$ be called Q' and let Q be obtained from Q' by replacing subformulas of the form $A(t)$ by *true*, and subformulas of the form $B(t)$ by *false*. It is easy to check that such a Q fulfills our requirements. ∎

5. Concluding Remarks

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 [1, 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 *while*-programs.

We would like to state some questions based on the following simple definitions:

DEFINITION 1. $\mathfrak{A}$ is PC-compact if for each asserted program $\{p\}\ S\ \{q\}$ true in $\mathfrak{A}$ there is a sentence $\varphi \in \mathscr{L}(\mathfrak{A})$ such that $\mathrm{mod}(\varphi) \models \{p\}\ S\ \{q\}$.

DEFINITION 2. A logic of partial correctness $\mathrm{LPC}(\mathfrak{A})$ for $\mathfrak{A}$ is an r.e. set of pairs $\langle \phi_i, \{p_i\}\ S_i\ \{q_i\}\rangle$ with $\varphi_i \in \mathscr{L}(\mathfrak{A})$ and $\{p_i\}\ S_i\ \{q_i\}$ an asserted program over $\mathscr{L}(\mathfrak{A})$. $\mathrm{LPC}(\mathfrak{A})$ is *sound* if for all $\langle \varphi, \{p\}\ S\ \{q\}\rangle \in \mathrm{LPC}(\mathfrak{A})$, $\mathrm{mod}(\phi) \models \{p\}\ S\ \{q\}$. $\mathrm{LPC}(\mathfrak{A})$ is *complete* if whenever $\mathfrak{A} \models \{p\}\ S\ \{q\}$ there is $\varphi \in \mathrm{Th}(\mathfrak{A})$ such that $\langle \varphi, \{p\}\ S\ \{q\}\rangle \in \mathrm{LPC}(\mathfrak{A})$.

DEFINITION 3. $\mathfrak{A}$ is PC-complete if there exists a sound and complete logic $\mathrm{LPC}(\mathfrak{A})$ for $\mathfrak{A}$.

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

6. QUESTIONS

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

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

ACKNOWLEDGMENTS

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

REFERENCES

1. BERGSTRA, J. A., & TUCKER, J. V. (1982), Some natural structures which fail to possess a sound and decidable Hoare-like logic for their *while*-programs, *Theor. Comput. Sci.* **17** (3), 235.
2. BERGSTRA, J. A., & TUCKER, J. V. (1982), Expressiveness and the completeness of Hoare's logic, *J. Comput. System Sci.* **25**, 217.
3. BERGSTRA, J. A., & TUCKER, J. V. (1982), Two theorems on the completeness of Hoare's logic, *Inf. Process. Lett.* **15** (4), 143.

4. COOK, S. A. (1978), Soundness and completeness of an axiom system for program verification, *SIAM J. Comput.* **7**, 129.
5. HAREL, D. (1979), "First-Order Dynamic Logic," Lecture Notes in Computer Science, Vol. 68, Springer-Verlag, Berlin/New York.
6. HAREL, D., MEYER, A. R., & PRATT, V. (1977), Computability and completeness in logics of programs: Preliminary report, *in* "9th ACM Symposium on Theory of Computing, Boulder, Colorado, May, 1977," pp. 261–268; revised version, M.I.T. Lab. for Computer Science TM-97, 16 pp., February 1978.
7. HOARE, C. A. R. (1969), An axiomatic basis for computer programming, *Commun. ACM* **12** (10), 576.
8. WAND, M. (1978), A new incompleteness result for Hoare's system. *J. Assoc. Comput. Math.* **25**(1), 168.