LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TM-226

"HOARE'S LOGIC IS NOT COMPLETE WHEN IT COULD BE"

J. Bergstra
A. Chmielinska
J. Tiuryn

August 1982

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

MIT/LCS/TM-226

"HOARE'S LOGIC IS NOT COMPLETE WHEN IT COULD BE"

J. Bergstra
A. Chmielinska
J. Tiuryn

August 1982

Hoare's Logic is Not Complete When It Could Be

J. Bergstra [1]

A. Chmielinska [2]

J. Tiuryn [3]

5 August 1982

1. University of Leiden, Department of Computer Science, The Netherlands
2. University of Torun, Department of Mathematics, Poland
3. M.I.T., Laboratory for Computer Science, USA,
   Boston University, Department of Mathematics, USA and
   University of Warsaw, Department of Mathematics, Poland.

This work was supported in part by The National Science Foundation, Grant No. MCS 8010707, and by a grant to the M.I.T. Laboratory for Computer Science by the IBM Corporation.

Abstract

It is known (cf. [2]) that if the Hoare rules are complete for a first-order structure $\mathcal{A}$, then the set of partial correctness assertions true over $\mathcal{A}$ is recursive in the first-order theory of $\mathcal{A}$. We show that the converse is not true. Namely, there is a first-order structure $\mathcal{C}$ such that the set of partial correctness assertions true over $\mathcal{C}$ is recursive in the theory of $\mathcal{C}$, but the Hoare rules are not complete for $\mathcal{C}$.

KEY WORDS

Hoare Logic, Partial Correctness Assertions, Relative Completeness

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 $\mathcal{A}$, the system HL($\mathcal{A}$) consists of Hoare's inference rules together with the first-order theory of $\mathcal{A}$. The structure $\mathcal{A}$ is *expressive* if, for any program α and first-order formula Q, there is a first-order formula P such that for all $\vec{a}$

$$\mathcal{A} \models P(\vec{a}) \equiv (\{\vec{x} = \vec{a}\}\ \alpha\ \{Q\}).$$

Cook's theorem states that if $\mathcal{A}$ is expressive, then HL($\mathcal{A}$) is complete. In general, however, the set PC($\mathcal{A}$) of partial correctness assertions that are valid over $\mathcal{A}$ is Π_1^0 in the first-order theory Th($\mathcal{A}$) of $\mathcal{A}$ (cf. [1]) whereas HL($\mathcal{A}$) is Σ_1^0 in Th($\mathcal{A}$). Thus HL($\mathcal{A}$) is not complete for arbitrary $\mathcal{A}$.

Although expressiveness is sufficient to guarantee the completeness of HL($\mathcal{A}$), 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 PC($\mathcal{A}$) is always Π_1^0 in Th($\mathcal{A}$) and HL($\mathcal{A}$) is Σ_1^0 in Th($\mathcal{A}$), then if HL is complete the partial correctness theory PC($\mathcal{A}$) is recursive in Th($\mathcal{A}$). This paper is motivated by the following question: *is* HL($\mathcal{A}$) *complete for every structure* $\mathcal{A}$ *such that* PC($\mathcal{A}$) *is recursive in* Th($\mathcal{A}$)? 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 (cf. [5]).

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

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 **while**-programs.

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

(*Composition Rule*) $\quad \{P\}\ \alpha\ \{Q\},\ \{Q\}\ \beta\ \{R\} \vdash \{P\}\ \alpha; \beta\ \{R\}$

(*Conditional rule*) $\quad \{P \wedge B\}\ \alpha\ \{Q\},\ \{P \wedge \neg B\}\ \beta\ \{Q\}$
$\vdash \{P\}$ *if* B *then* α *else* β *fi* $\{Q\}$

(*Iteration rule*) $\quad P \supset R, \{R \wedge B\}\ \alpha\ \{R\},\ R \wedge \neg B \supset Q$
$\vdash \{P\}$ *while* B *do* α *od* $\{Q\}$

(*Oracle axioms*) $\quad$ Every $P \in Th(\mathcal{A})$ 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, $HL(\mathcal{A})$ denotes the set of all asserted programs $\{P\}\ \alpha\ \{Q\}$ provable from $Th(\mathcal{A})$ 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

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

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 $HL(\mathcal{A})$ proves $\{P\}\ \alpha\ \{Q\}$, then there exists a proof of $\{P\}\ \alpha\ \{Q\}$ in $HL(\mathcal{A})$ 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 $HL(\mathcal{A})$ 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 $\mathcal{L}_1$ and $\mathcal{L}_2$ be two similarity types. Let A and B be unary predicate symbols

and $\perp$ a constant symbol, none of which belong to $\mathcal{L}_1 \cup \mathcal{L}_2$. Let $\mathcal{A}$, $\mathcal{B}$ be $\mathcal{L}_1$-, $\mathcal{L}_2$-structures, respectively. For any integer i and set X let X×i denote X×{i}. Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \{A,B,\perp\}$. We define an $\mathcal{L}$-structure $\mathcal{A} \oplus \mathcal{B}$ with carrier $|\mathcal{A} \oplus \mathcal{B}| = |\mathcal{A}|\times 0 \cup |\mathcal{B}|\times 1 \cup \{\langle 2,2\rangle\}$. 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 $\langle 2,2\rangle$. We interpret the $\mathcal{L}_1$ (respectively $\mathcal{L}_2$) 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 $\mathcal{L}_1$ (respectively $\mathcal{L}_2$) 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 $R \in \mathcal{L}_1 \cap \mathcal{L}_2$, 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 $\mathcal{A}$ there is a structure $\mathcal{B}$ such that
PC($\mathcal{A}\oplus\mathcal{B}$) is recursive in Th($\mathcal{A}\oplus\mathcal{B}$), and

Theorem2: For every two structures $\mathcal{A}$ and $\mathcal{B}$
if HL($\mathcal{A}$) is incomplete, then so is HL($\mathcal{A}\oplus\mathcal{B}$).

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

Corollary : There is a structure $\mathcal{C}$ such that
PC($\mathcal{C}$) is recursive in Th($\mathcal{C}$) and HL($\mathcal{C}$) is incomplete.

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

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 $\mathcal{A}$ with HL($\mathcal{A}$) incomplete, HL($\mathcal{A}\oplus$N) is incomplete. Harel's theorem says that if $\mathfrak{B}$ is a structure which contains the standard model of arithmetic (as a first order definable part of $\mathfrak{B}$) and if $\mathfrak{B}$ has the ability to code finite sequences of elements from $|\mathfrak{B}|$, then the first order language is expressive for *while*-programs over $\mathfrak{B}$, and therefore HL($\mathfrak{B}$) is complete. Since obviously N is a first order definable part of $\mathcal{A}\oplus$N, but HL($\mathcal{A}\oplus$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 $\mathcal{A}$, there is a structure $\mathfrak{B}$ such that PC($\mathcal{A}\oplus\mathfrak{B}$) is recursive in Th($\mathcal{A}\oplus\mathfrak{B}$). If the domain of $\mathcal{A}$ is finite, then $\mathfrak{B}$ may be chosen to be any finite structure. Then $\mathcal{A}\oplus\mathfrak{B}$ is finite and PC($\mathcal{A}\oplus\mathfrak{B}$) is recursive in Th($\mathcal{A}\oplus\mathfrak{B}$). When $\mathcal{A}$ is infinite, we will define $\mathfrak{B}$ to be a copy of $\mathcal{A}$ which also has the standard arithmetic operations defined on the elements of its domain. By construction, the first-order theory of $\mathfrak{B}$ will contain both the first-order theory of $\mathcal{A}$ and the first-order theory of arithmetic. As a consequence, PC($\mathcal{A}\oplus\mathfrak{B}$) will be recursive in Th($\mathcal{A}\oplus\mathfrak{B}$). However, the structure $\mathcal{A}\oplus\mathfrak{B}$ need not be expressive since there may not be any way to code pairs of elements of $|\mathcal{A}|$ in Th($\mathcal{A}\oplus\mathfrak{B}$).

Assume now that $\mathcal{A}$ is infinite and its similarity type is $\mathcal{L}_1$. We construct $\mathfrak{B}$ so that PC($\mathcal{A}\oplus\mathfrak{B}$) is recursive in Th($\mathcal{A}\oplus\mathfrak{B}$) as follows. First, we expand $\mathcal{A}$ 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, $\mathfrak{B}$, has the same domain as $\mathcal{A}$ but has a richer similarity type which we denote by $\mathcal{L}_2$.

It is clear that Th($\mathfrak{B}$) is recursive in Th($\mathcal{A}\oplus\mathfrak{B}$). This follows immediately from the

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

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

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 $\mathfrak{B}$ 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 $|\mathfrak{B}|$ which correspond to these names with the actual numbers 0,1,2. By means of this identification, we can view

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

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

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

$$\mathcal{A} \oplus \mathfrak{B} \models P(x_1,\ldots, x_n)[c_1,\ldots,c_n]$$
$$\text{iff}$$
$$\mathfrak{B} \models \mathbf{Tr}\,(P)(x_1,y_1,\ldots,x_n,y_n)[\pi_1(c_1),\pi_2(c_1),\ldots,\pi_1(c_n),\pi_2(c_n)].$$

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 *true* if t is a term over language $\mathcal{L}_1$ 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 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 $|\mathfrak{B}| \times 1$ and $t = t'$, or

(c) both t and t' are $\perp$

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

(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 $R \in \mathcal{L}_1$ and let $X = \{x_i: i \in J\}$ contain variables that occur in P. Then Tr(P) is:

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

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

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

The cases for P of the form B(t) or $R(t_1, ..., t_n)$ with $R \in \mathcal{L}_2\text{-}\mathcal{L}_1$ 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

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

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

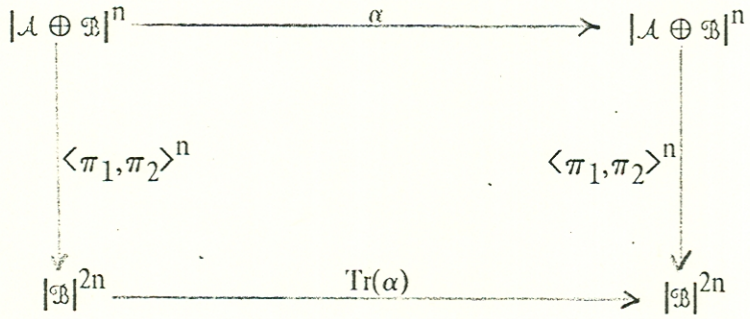
$$\exists x_i\, \exists y_i\, ((y_i=0 \vee y_i=1 \vee y_i=2) \wedge \mathbf{Tr}(P)).$$

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 $\mathcal{A}\oplus\mathcal{B}$

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

Let α be a program and let $\{x_1, ..., x_n\}$ contain all variables occurring in α. The translation $\mathbf{Tr}(\alpha)$ will use variables $x_1, y_1, ..., x_n, y_n$ in such a way that the following diagram commutes



To achieve this, every assignment statement $x_i := t$ in α is replaced by a pair of assignment statements for x_i and y_i, depending on $L_1(t)$ and the values of y_j's corresponding to x_j's occurring in t. Every test P in α is replaced by $\mathbf{Tr}(P)$. The details of this construction are left for the reader.

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 Sum

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

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

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

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 proposition.

Proposition 3: 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 $\mathcal{L}_1$ 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.

Proof of Theorem 2: Assume that $HL(\mathcal{A})$ is incomplete and $HL(\mathcal{A}\oplus\mathcal{B})$ complete. Choose $\{P\}\ \alpha\ \{Q\}$ 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 if $HL(\mathcal{A}\oplus\mathcal{B}) \vdash \{A(X) \wedge P_A\}\ \alpha\ \{A(X) \wedge Q_A\}$.

We derive a contradiction by constructing a proof of $\{P\}\ \alpha\ \{Q\}$ 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 $\{R\}\ \alpha\ \{S\}$ 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 $\mathcal{L}_1$. By Lemma 2, $HL(\mathcal{A})$ proves $\{P\}\ \alpha\ \{Q\}$ in contrast to our assumptions. ∎

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

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

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

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 $\mathcal{A} \oplus \mathcal{B}$ is normalized iff there is a number n, formulae $F^1,...,F^n$ over $\mathcal{L}_1 \cup \{A,B\}$ and formulae $G^1,...,G^n$ over $\mathcal{L}_2 \cup \{A,B\}$ such that P of the form $\bigwedge_{i=1}^{n} (F^i_A \vee G^i_B)$.

Lemma 5: Let P be a formula over $\mathcal{L}_2 \cup \{A,B\}$. There exists a normalized formula Q of the form $\bigwedge_{i=1}^{n} (F^i_A \vee G^i_B)$ such that $\mathcal{A} \oplus \mathcal{B} \models A(x) \supset (P_B \equiv Q)$ 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 $\mathcal{L}_1 \cup \{A,B\}$ 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 $\mathcal{L}_1 \cup \{A,B\}$, contains x as a free variable and is of the form $R(t_1,...)$ then A(x) implies $P_B \equiv$ *false*. The remaining subcase is a formula of the form $t_1 = t_2$, not over $\mathcal{L}_1 \cup \{A,B\}$, 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 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 $\mathcal{L}_1 \cup \{A,B\}$ by *false* or by a combinations 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 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 occur as a free variable.

Due to the steps 1, 2 and 3 formulae F's are over $\mathcal{L}_1 \cup \{A\}$, 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 $\mathcal{L}_2 \cup \{A,B\}$. 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 $\mathcal{L}_1$ is interchanged with $\mathcal{L}_2$ and A with B.

Lemma 6: For every formula P of the language of $\mathcal{A} \oplus \mathcal{B}$ there is a normalized 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 $\mathcal{L}_1 \cup \{A,B\}$ (resp. $\mathcal{L}_2 \cup \{A,B\}$) 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 $\mathcal{A} \oplus \mathcal{B}$ to *false*, and if it is of the form $t_1 = t_2$ then it is equivalent to $t_1 = t_2 = \bot$. 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 $\forall x$ Q. We assume inductively, that over $\mathcal{A} \oplus \mathcal{B}$ the formula Q is equivalent to normalized Q', where Q' is of the form $\bigwedge_{i=1}^{n} (F_A^i \vee G_B^i)$.

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

it is enough to show a transformation of every formula $\forall x(F^i_A \vee G^i_B)$ for $i=1,...,n$ 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) $F^i_A(\perp/x) \vee G^i_B(\perp/x)$

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

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

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

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

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

Because we assumed that F's and H's are over $\mathcal{L}_1 \cup \{A,B\}$ and J's are over $\mathcal{L}_2 \cup \{A,B\}$, the last formula is normalized . 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 $\mathcal{L}_1 \cup \{A,B\}$ and G_i's are over $\mathcal{L}_2 \cup \{A,B\}$.

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 normalized 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 variable occur. Let ϵ_i be *true* if $\mathcal{A}\oplus\mathcal{B} \models A(X) \wedge L^i_B$, and *false* otherwise.

Clearly

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

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

5. References

1. Bergstra, J.A. and J.F. 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. and J.F. Tucker. Expressiveness and the completeness of Hoare's logic. Tech. Rep. IW 149/80, Amsterdam University Mathematical Centre Report, 1980. To appear in JCSS.

3. Bergstra, J.A. and J.F. Tucker. Two Theorems on the Completeness of Hoare's Logic. Tech. Rep. IW 81, Amsterdam University Mathematical Centre Report, 1981, To appear in Inf. Proc. Let.

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. *Lecture Notes in Computer Science. Vol. 68: First-Order Dynamic Logic.* Springer-Verlag, 1979.

6. Harel, D., A.R. Meyer and 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.

Table of Contents