Hoare's Logic Is Incomplete When It Does Not Have To Be

J. Bergstra[†]
A. Chmielinska[††]
J. Tiuryn[†††]

Abstract:

If Hoare's Logic, HL(A), is complete on a structure A, then the set PC(A) of all asserted programs true over A is recursive in the first order theory of A, Th(A). We show that this implication cannot be reversed.

Introduction

It is known (cf. [5]) that for a given first order language $\mathcal{L}$, the set of Floyd-Hoare partial correctness assertions $PC_{\mathcal{L}} = \{\{P\}\ \alpha\ \{Q\} \mid P,Q$ are first order formulae over $\mathcal{L}$, α is a *while*-program over $\mathcal{L}$, and $\models \{P\}\ \alpha\ \{Q\}\}$ is a Π^0_2-complete set. In particular $PC_{\mathcal{L}}$ is not r.e., and therefore it cannot possess a finitary sound and (absolutely) complete proof system.

Let HL denote the standard Hoare's proof system for proving partial correctness of *while*-programs. Cook's [4] idea to repair incompleteness of HL was to take the first order

† University of Leiden, Department of Computer Science, The Netherlands
†† University of Torun, Department of Mathematics, Poland
††† M.I.T., Laboratory for Computer Science, USA
††† Boston University, Department of Mathematics, USA
††† 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.

theory of an $\mathcal{L}$- structure $\underline{A}$, abbreviated Th($\underline{A}$), as additional axioms, obtaining in this way an enlarged proof system HL($\underline{A}$). In general for a given $\mathcal{L}$-structure $\underline{A}$, the set PC($\underline{A}$) = $\{\{P\}\ \alpha\ \{Q\} \mid A \models \{P\}\ \alpha\ \{Q\}\}$ is Π_1^0 in Th($\underline{A}$) (cf. [1]), and hence still is not r.e. even in Th($\underline{A}$). Thus HL($\underline{A}$) cannot be complete for all $\underline{A}$'s.

For some structures $\underline{A}$, however, HL($\underline{A}$) is complete. Cook's theorem [4] says that for every structure $\underline{A}$ for which $\mathcal{L}$ is expressive for *while*-programs, HL($\underline{A}$) is complete. However, expressiveness is not necessary for completeness of the HL system: any non standard model of arithmetic possess a complete HL, but the first order language of arithmetic is **never** expressive for *while*-programs over such model (cf. [2]). Moreover, expressiveness represents what amounts to a degenerate case: when expressiveness holds, programming logic (e.g. regular Dynamic Logic) collapses to first order logic, and in particular partial correctness assertions reduce to first order formulae.

From the above it follows that if for a given structure $\underline{A}$, HL($\underline{A}$) is complete, then PC($\underline{A}$) is recursive in Th($\underline{A}$). This paper is motivated by the following question: **is** HL($\underline{A}$) **complete for every structure** $\underline{A}$ **such that** PC($\underline{A}$) **is recursive in** Th($\underline{A}$)? It is our aim to show that the answer to the above question is negative. We will present a general construction of counterexamples for this situation.As a corollary of our results we obtain an example of a structure which shows that ability of coding finite sequences cannot be removed from assumptions of Harel's theorem on arithmetic universes (cf. [5]).

Suppose now for a while that HL($\underline{A}$) is incomplete but PC($\underline{A}$) is recursive in Th($\underline{A}$).What can be done to make HL complete? For this question it is meaningful to consider HL($\mathcal{L}$, E) as a proof system over a first order theory E in language $\mathcal{L}$.HL($\underline{A}$) may be then identified with HL($\mathcal{L}$, Th($\underline{A}$)), where $\underline{A}$ is an $\mathcal{L}$-structure. It follows from [3] that $\underline{A}$ can be expanded to an $\mathcal{L}^*$-structure $\underline{A}^*$ with $\mathcal{L}^*$-$\mathcal{L}$ being finite, such that for some decidable theory T $\subseteq$ Th($\underline{A}^*$), PC($\underline{A}$) $\subseteq$ HL($\mathcal{L}^*$, Th($\underline{A}$) $\cup$ T).Thus Th($\underline{A}$) $\cup$ T, where T is decidable but formulated in an extended language contains enough information to derive all of PC($\underline{A}$).

1. Preliminaries

1.1 For completeness sale we start with a formulation of HL that suits our purposes best. There are four rules of inference.

1.1.1 **Assignment rule**

$$\frac{P \Rightarrow Q[t/x]}{\{P\}\ x:=t\ \{Q\}}$$

where P,Q are first order formulae, x is a variable, t is a term, and Q[t/x] denotes a formula obtained from Q by uniformly substituting t for all free occurences of x in Q.

1.1.2 **Composition rule**

$$\frac{\{P\}\ \alpha\ \{Q\},\ \{Q\}\ \beta\ \{R\}}{\{P\}\ \alpha;\beta\ \{R\}}$$

where P,Q,R are first order formulae, and α, β are *while*-programs.

1.1.3 **Conditional rule**

$$\frac{\{P \wedge B\}\ \alpha\ \{Q\},\ \{P \wedge \neg B\}\ \beta\ \{Q\}}{\{P\}\ \textit{if}\ B\ \textit{then}\ \alpha\ \textit{else}\ \beta\ \textit{fi}\ \{Q\}}$$

where P,Q,B are first order formulae, B is quantifier free, and α, β are *while*-programs.

1.1.4 **Iteration rule**

$$\frac{P \Rightarrow R,\ \{R \wedge B\}\ \alpha\ \{R\},\ R \wedge \neg B \Rightarrow Q}{\{P\}\ \textit{while}\ B\ \textit{do}\ \alpha\ \textit{od}\ \{Q\}}$$

where P,R,Q,B are first order formulae, B is quantifier free, and α is a *while*-program.

1.1.5 **Oracle axiom** (for a given structure $\underline{A}$). Every $P \in Th(\underline{A})$ is an axiom.

Given a structure $\underline{A}$, $HL(\underline{A})$ denotes the set of all asserted programs $\{P\}\ \alpha\ \{Q\}$ provable by using rules (1.1.1 - 1.1.4) and oracle axiom for $\underline{A}$.

The reader can easily check that the following rule

Consequence rule

$$\frac{P \Rightarrow P_1,\ \{P_1\}\ \alpha\ \{Q_1\},\ Q_1 \Rightarrow Q}{\{P\}\ \alpha\ \{Q\}}$$

where P, P_1, Q, Q_1 are first order formulae, and α is a *while*-program; is derivable in HL.

Another rule easily derivable in HL is the following:

$$\frac{\{P\}\ \alpha\ \{Q\}}{\{\exists x P\}\ \alpha\ \{\exists x Q\}}$$

where P,Q are first order formulae, α is a *while*-program and x does not occur in α.

We will need a slightly stronger statement. Let a proof of $\{P\}\ \alpha\ \{Q\}$ in HL($\underline{A}$) be given and let x does not occur in α. This proof can be transforwarded into another one by putting $\{\exists x P\}\ \alpha\ \{\exists x Q\}$. Repeated application of this procedure yields the following lemma.

1.1.6 **Lemma**

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

1.2 As the next step in this technical introduction we will explain a construction of a structure $\underline{A} \oplus \underline{B}$ on which our paper is technically based.

Let $\mathcal{L}_1$, and $\mathcal{L}_2$ be two disjoint similarity types. Let $A, B, \perp$ be new symbols not belong to $\mathcal{L}_1 \cup \mathcal{L}_2$. A and B are unavy predicate symbols, and $\perp$ is a constant symbol. Let $\underline{A}$, $\underline{B}$ be $\mathcal{L}_1$-, $\mathcal{L}_2$-structures, respectively. Assume that carriers of $\underline{A}$ and $\underline{B}$ (denoted by $|\underline{A}|$ and $|\underline{B}|$, respectively) are disjoint. Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \{A, B, \perp\}$. $\underline{A} \oplus B$ is an $\mathcal{L}$-structure with carrier $|\underline{A} \oplus \underline{B}| = |\underline{A}| \cup |\underline{B}| \cup \{\perp\}$, where $\perp$ is a new element, not in $|\underline{A}| \cup |\underline{B}|$. A is a characteristic predicate of $|\underline{A}|$, and B is a characteristic predicate of $|\underline{B}|$. $\mathcal{L}_1$ (respectively $\mathcal{L}_2$) function symbols are defined the same as in $\underline{A}$ (as in $\underline{B}$, respectively), provided all arguments are taken from $|\underline{A}|$ (from $|\underline{B}|$, resp.) getting value $\perp$ otherwise. $\mathcal{L}_1$ (respectively $\mathcal{L}_2$) predicate

symbols are defined the same as in $\underline{A}$ (as in $\underline{B}$, resp.), provided all arguments are taken from $|\underline{A}|$ (from $|\underline{B}|$, resp.), and set to be false elsewhere.

$\underline{A} \oplus \underline{B}$ is a kind of disjoint union of $\underline{A}$ and $\underline{B}$. Clearly a meaningful alternative to this definition would be a two-sorted structure. We will not do so mainly with the purpose in mind to keep as close as possible to the standard Hoare formalism.

We are now in position to formulate a general theorem which has the claim in the introduction as a corollary.

1.2.1 **Theorem**

(i) For every $\underline{A}$ there is a structure $\underline{B}$ such that $PC(\underline{A}\oplus\underline{B})$ is recursive in $Th(\underline{A}\oplus\underline{B})$.

(ii) For arbitrary $\underline{A}$ and $\underline{B}$, if $HL(\underline{A})$ is incomplete, then so is $HL(\underline{A}\oplus\underline{B})$.

It follows from the above result that if we take $\underline{A}$ such that $HL(\underline{A})$ is incomplete (e.g. $\underline{A}$ can be chosen to be $\langle\omega,s,o\rangle$, where s is a successor function on nonnegative integers, or $\underline{A}$ can be the field of complex numbers, cf. [1] for proofs of incompleteness of $HL(\underline{A})$ for these $\underline{A}$'s), and choose $\underline{B}$ such that $PC(\underline{A}\oplus\underline{B})$ is recursive in $Th(\underline{A}\oplus\underline{B})$, then $HL(\underline{A}\oplus\underline{B})$ is also incomplete, and $\underline{A} \oplus \underline{B}$ is the kind of structure we looked for.

Let $\underline{N}$ stand for the standard model of arithmetic. By 1.2.1(ii) we have that for any $\underline{A}$ with incomplete $HL(\underline{A})$, $HL(\underline{A}\oplus\underline{N})$ is incomplete. Harel's theorem on arithmetical universes (cf. [5]) says that if $\underline{B}$ is a structure which contains the standard model of arithmetic (as a first order definable part of $\underline{B}$) and if $\underline{B}$ has "ability" to code finite sequences of elements from $|\underline{B}|$, then the first order language is expressive for *while*-programs over $\underline{B}$, and therefore $HL(\underline{B})$ is complete. Since obviously $\underline{N}$ is a first order definable part of $\underline{A} \oplus \underline{N}$, this shows that quite technical assumption of being able to code finite sequences cannot be removed from Harel's theorem.

The proofs of (i) and (ii) are quite independent and are subjects of sections 2 and 3.

2. Adding an expressive structure (proof of (i))

Let $\underline{A}$ be a structure of signature $\mathcal{L}_1$. If $\underline{A}$ is finite, then choose any finite $\underline{B}$. In that case $\underline{A} \oplus \underline{B}$ is finite as well and $PC(\underline{A} \oplus \underline{B})$ is recursive in $Th(\underline{A} \oplus \underline{B})$. This rerifies theorem 1.2.1(i) in that case.

Assume now that $\underline{A}$ is infinite. $\underline{B}$ is constructed as follows. Let $\underline{A}^*$ be an isomorphic copy of $\underline{A}$ such that $|\underline{A}| \cap |\underline{A}^*| = \varnothing$. Let $\mathcal{L}_1^*$ be a similarity type of $\underline{A}^*$, such that $\mathcal{L}_1^*$ is an isomorphic copy of $\mathcal{L}_1$ and disjoint with $\mathcal{L}_1$. Then we expand $\underline{A}^*$ to an arithmetic universe in the sense of [5] (i.e. we add a definning predicate for "non negative integers" N, arithmetic operations, a relation for coding finite sequences, and in addition, for technical reasons, we add pairing and unpairing functions). We also expand the language of $\underline{A}^*$ by three new constants $\underline{a}$, $\underline{b}$, $\underline{\perp}$ denoting different elements. The resulting structure is $\underline{B}$. It has the same domain as $\underline{A}^*$, but it has got a richer similarity type which we will denote by $\mathcal{L}_2$.

It is clear that

2.1 $Th(\underline{B})$ is recursive in $Th(\underline{A} \oplus \underline{B})$.

It follows immediately from the definition of $\oplus$-construction.

Because $\underline{B}$ is expressive (being an arithmetical universe), $HL(\underline{B})$ is complete. Therefore

2.2 $PC(\underline{B})$ is recursive in $Th(\underline{B})$. By (2.1) and (2.2) it is enough to prove that

2.3 $PC(\underline{A} \oplus \underline{B})$ is many-one reducible to $PC(\underline{B})$.

We will outline the proof of (2.3) by showing an effective simulation of computations on $\underline{A} \oplus \underline{B}$ by those on $\underline{B}$. The details of this simulation are slightly messy but completely harmless and we take the liberty to skip some of them.

Let $M: |\underline{A}| \Rightarrow |\underline{B}|$ be a bijective mapping which corresponds to the isomorphism $\underline{A} \simeq \underline{A}^*$. First we code $\underline{A} \oplus \underline{B}$ in $\underline{B} \times \underline{B}$ as follows:

$$\mathbf{Code(x)} = \begin{cases} (\underline{a}, M(x)), & \text{if } x \in |\underline{A}| \\ (\underline{b}, x), & \text{if } x \in |\underline{B}| \\ (\underline{\perp}, \underline{\perp}), & \text{if } x = \perp \end{cases}$$

In order to describe a smooth translation of assertions and programs we introduce two infinite families of new variables: $y_0, y_1, \ldots, z_0, z_1, \ldots$

As a next step we show an effective translation **Tr** of first order formulae over the language of $\underline{A} \oplus \underline{B}$ which use individual variables: $x_0, x_1, \ldots$ to first order formulae over $\mathcal{L}_2$ with variables $y_0, y_1, \ldots, z_0, z_1, \ldots$. The translation will have the following property:
for every $P(x_0, \ldots, x_{n-1})$ over the language of $\underline{A} \oplus \underline{B}$, and for all $c_0, \ldots, c_{n-1} \in |\underline{A} \oplus \underline{B}|$,
$\underline{A} \oplus \underline{B} \models P(x_0, \ldots, x_{n-1})[c_0, \ldots, C_{n-1}]$ iff $\underline{B} \models \mathbf{tr}(P)(y_0, z_0, \ldots, y_{n-1}, z_{n-1})[\mathbf{code}(c_0), \ldots, \mathbf{code}(c_{n-1})]$.

Because the formal definition of **Tr** is slightly cumbersome we present its details. We first introduce some notations. For a term t and for $i \in \{1,2\}$, $\mathcal{L}_i(t)$ is a truth value equal to **true** iff t is a term over language $\mathcal{L}_i$.

Moreover, if t is a term over $\mathcal{L}_1$, then $\tilde{t}$ denotes the corresponding term over $\mathcal{L}_1{}^* \subseteq \mathcal{L}_2$.

We define **Tr** inductively. Let P be $t = t'$, where t contains the variables $X = \{x_{i_1}, \ldots, x_{i_k}\}$, and t' contains the variables $X' = \{x_{j_1}, \ldots, x_{j_m}\}$. Then **Tr**(P) is

2.4 "t in $\underline{A}$, and t' in $\underline{A}$, and $t = t'$" $\vee$

2.5 "t in $\underline{B}$, and t' in $\underline{B}$, and $t = t'$" $\vee$

2.6 "t is $\perp$, and t' is $\perp$"

Fomulae (2.4) - (2.6) can formally be written as follows:

(2.4') $$\bigwedge_{x_i \in X \cup X'} (y_i = \underline{a}) \wedge \mathcal{L}_1(t) \wedge \mathcal{L}_2(t') \quad \wedge$$
$$\tilde{t}[z_{i_1}/x_{i_1}, \ldots, z_{i_k}/x_{i_k}] = \tilde{t}'[z_{j_1}/x_{j_1}, \ldots, z_{j_m}/x_{j_m}].$$

(2.5') $$\bigwedge_{x_i \in X \cup X'} (y_i = \underline{b}) \wedge \mathcal{L}_2(t) \wedge \mathcal{L}_2(t') \quad \wedge$$
$$t[z_{i_1}/x_{i_1}, \ldots, z_{i_k}/x_{i_k}] = t'[z_{j_1}/x_{j_1}, \ldots, z_{j_m}/x_{j_m}]$$

(2.6') $$[\bigvee_{x_i \in X} (y_i = \perp) \quad \vee \quad (\neg \mathcal{L}_1(t) \wedge \neg \mathcal{L}_2(t)) \quad \vee$$
$$(\mathcal{L}_1(t) \wedge \bigvee_{x_i \in X} (y_i = \underline{b})) \quad \vee \quad (\mathcal{L}_2(t) \wedge \bigvee_{x_i \in X'} (y_i = \underline{a}))] \quad \wedge$$

$$[\bigvee_{x_i \in X'} y_1 = \underline{\perp} \quad \vee \quad (\neg \mathcal{L}_1(t') \wedge \neg \mathcal{L}_2(t')) \quad \vee$$

$$(\mathcal{L}_1(t') \wedge \bigvee_{x_i \in X'} (y_i = \underline{b})) \quad \vee \quad (\mathcal{L}_2(t') \wedge \bigvee_{x_i \in X'} (y_i = \underline{a}))].$$

If P is $R(t_0,...,t_{n-1})$ with $R \in \mathcal{L}_1$ and $X = \{x_{i_1},...,x_{i_k}\}$ are all variables occuring in P then **Tr**(P) is:

$$\bigwedge_{x_i \in X} (y_i = \underline{a}) \quad \wedge \quad \bigwedge_{j=0}^{n-1} \mathcal{L}_1(t_j) \quad \wedge$$

$\tilde{R}(\tilde{t}_0(z_{i_1}/x_{i_1}, ..., z_{i_k}/x_{i_k}), ..., \tilde{t}_{n-1}(z_{i_k}/x_{i_k}, ..., z_{i_k}/x_{i_k}))$., where $\tilde{R} \in \mathcal{L}_1{}^*$ is a symbol which corresponds to R.

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

$$\bigwedge_{x_i \in X} (y_i = \underline{a}) \quad \wedge \quad \mathcal{L}_1(t).$$

The cases P is B(t) or P is $R(t_0, ..., t_{n-1})$ with $R \in \mathcal{L}_2$ are dealt similarly and are left for the reader.

If P is $P_1 \vee P_2$, then **Tr**(P) is **Tr**(P_1) $\vee$ **Tr**(P_2).

If P has free variables $X = \{x_{i_1}, ..., x_{i_m}\}$, then **Tr**(¬P) is

$$\bigwedge_{x_i \in X} (y_i = \underline{a} \quad \vee \quad y_i = \underline{b} \quad \vee \quad y_i = \underline{\perp}) \quad \wedge \quad \neg \mathbf{Tr}(P).$$

And finally **Tr**($\exists x_i$ P) is

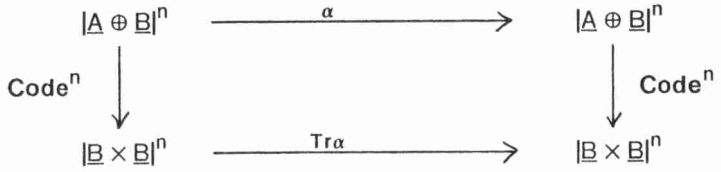
$$\exists y_i \, \exists z_i \, ((y_i = \underline{a} \quad \vee \quad y_i = \underline{b} \quad \vee \quad y_i = \underline{\perp}) \quad \wedge \quad \mathbf{Tr}(P)).$$

This describes **Tr** in detail.

The next step is to extend **Tr** to programs α over the language of $\underline{A} \oplus \underline{B}$ in order to have this property for all first order formulae P, Q over the language of $\underline{A} \oplus \underline{B}$:

2.7 $\quad \underline{A} \oplus \underline{B} \models \{P\}\alpha\{Q\}$ iff $\underline{B} \models \{\mathbf{Tr}(P)\}\, \mathbf{Tr}(\alpha)\{\mathbf{Tr}(Q)\}$.

Given a program α and let $\{x_0, ..., x_{n-1}\}$ be all variables occuring in α. **Tr**(α) will be using variables $y_0, z_0, ..., y_{n-1}, z_{n-1}$ in such a way that the following diagram commutes



To achieve this every assignment statement $x_i := t$ in α is replaced by an assignment statement to both y_i and z_i, depending on $\ell_1(t)$, $\ell_2(t)$, and the values of y_j's corresponding to x_j's occuring in t. Every test P in α is replaced by $\mathbf{Tr}(P)$. The details of this construction can safely be left for the reader.

By (2.7), PC($\underline{A} \oplus \underline{B}$) is many-one reducible to PC($\underline{B}$). This completes the proof of part (i) of Theorem 1.2.1.

3. Hoare's Logic over $\underline{A} \oplus \underline{B}$ (Proof of (ii))

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

First we introduce some notations. Let P be a first order formula over the language of $\underline{A} \oplus \underline{B}$. $(P)_A$, a relativization of P to $|\underline{A}|$ is defined inductively as follows:

if P is atomic, then $(P)_A$ is P

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

$(P_1 \vee P_2)_A$ is $(P_1)_A \vee (P_2)_A$

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

If X is a finite set of variables, then A(X) will always denote $\bigwedge_{x \in X} A(x)$. $(P)_B$ and B(X) are defined similarily.

We can now formulate on important lemma connecting PC(A) and PC($\underline{A} \oplus \underline{B}$).

3.1 Lemma

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 occuring free in P or Q or α.

(i) $\underline{A} \models \{P\}\ \alpha\ \{Q\}$ iff $\underline{A} \oplus \underline{B} \models \{A(X) \wedge (P)_A\}\ \alpha\ \{A(X) \wedge (Q)_A\}$.

(ii) If HL($\underline{A} \oplus \underline{B}$) proves $\{A(X) \wedge (P)_A\}\ \alpha\ \{A(x) \wedge (Q)_A\}$ using only invariants and inter-mediate assertions with free variables in X and of the form $A(X) \wedge (R)_A$, then HL($\underline{A}$) proves $\{P\}\ \alpha\ \{Q\}$.

Both parts of the above lemma are obvious and the proof is omitted. The result at this stage can be proved using the following proposition.

3.2 Proposition

Let P be a first order formula over the language of $\underline{A} \oplus \underline{B}$. Let X be the set of all variables occuring free in P. there exists a first order formula Q over $\mathcal{L}_1$ such that

$\underline{A} \oplus \underline{B} \models (A(X) \wedge P) \leftrightarrow (A(X) \wedge (Q)_A)$.

Before we start proving Proposition 3.2 we show how to complete the proof of Theorem 1.2.1. Assume that HL($\underline{A}$) is incomplete. Choose $\{P\}\ \alpha\ \{Q\}$ true in $\underline{A}$ but not derivable in HL($\underline{A}$). Let X be the set of all variables occuring free in P, Q, or α. By Lemma 3.1.(i), $\{A(X) \wedge (P)_A\}\ \alpha\ \{A(X) \wedge (Q)_A\}$ is true in $\underline{A} \oplus \underline{B}$, and therefore if HL($\underline{A} \oplus \underline{B}$) $\vdash \{A(X) \wedge (P)_A\}\ \alpha\ \{A(X) \wedge (Q)_A\}$.

We derive a contradiction from this fact. Choose a proof of this asserted program in HL($\underline{A} \oplus \underline{B}$) and use Lemma 1.1.1 to transform it into a proof in which all intermediate assertions and invariants have their free variables in X. Conjuncting A(X) to all of these will not disburb the proof. Then, using Proposition 3.2, all invariants and intermediate assertions can be written in the form $A(X) \wedge (R)_A$ with R a first order formula over $\mathcal{L}_1$. By Lemma 3.1(ii), HL($\underline{A}$) proves $\{P\}\ \alpha\ \{Q\}$ in contrast to our assumptions.

The proof of Proposition 3.2 follows from a series of auxiliary lemmas stated below.

3.3 **Lemma** ($\perp$ - 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) $\underline{A} \oplus \underline{B} \models (P) \leftrightarrow P^{\perp}$

(ii) $\underline{A} \oplus \underline{B} \models (P^{\perp})_A \leftrightarrow ((P)_A)^{\perp}$.

Proof: The proof is by induction on structure of P. The basis step is long due to a large number of cases to be considered. We show only one - the other cases for the basis step are dealt similarly.

Let P be $t_1 = t_2$ where t_1, t_2 are terms, and assume that $\perp$ occurs in t_1. It is easy to see that over $\underline{A} \oplus \underline{B}$ t_1 assumes constantly value $\perp$. Consider the following three sub-cases for t_2:

3.3.1 $\perp$ occurs in t_2

3.3.2 t_2 is a variable, say x

3.3.3 remaining cases.

For (3.3.1) we define $P^{\perp}$ as *true*. For (3.3.2) $P^{\perp}$ is $\neg A(x) \wedge \neg B(x)$. Finally for (3.3.3), if X is the set of all variables occuring in t_2, then we set $P^{\perp}$ to be $\neg A(X)$, i.e. $\bigvee_{x_i \in X} \neg A(x_i)$. Observe that if $X = \varnothing$ then $P^{\perp}$ becomes *false*. It is easy to check that $P^{\perp}$ defined above satisfies (i) and (ii). The inductive step is straightforward.

$$(P \wedge Q)^{\perp} \quad \text{is} \quad P^{\perp} \wedge Q^{\perp}$$

$$(\neg P)^{\perp} \quad \text{is} \quad \neg(P^{\perp})$$

$$(\forall x\, P)^{\perp} \quad \text{is} \quad \forall x\, (P^{\perp}).$$

Again it is very easy to check that $P^{\perp}$ has the desired properties.

3.4 **Lemma**

Let P be a first order formula over $\mathcal{L}_1 \cup \{A,B\}$, and assume that x occurs free in P. There exists a formula $P^{(x)}$ over $\mathcal{L}_1 \cup \{A,B\}$ such that x is not free in $P^{(x)}$, and

(i) $\underline{A} \oplus \underline{B} \models B(x) \Rightarrow (P \leftrightarrow P^{(x)})$

(ii) $\underline{A} \oplus \underline{B} \models B(x) \Rightarrow \{(P^{(x)})_A \leftrightarrow ((P)_A)^{(x)}\}$.

Proof. The proof is by induction on P. Again, the basis step is quite long due to a large number of cases to be considered, and as in the previous proof we will discuss here only one case in detail leaving for the reader the remaining one.

Suppose P is $t_1 = t_2$ and consider the following possibilities.

3.4.1 Both t_1 and t_2 are x

3.4.2 x occurs in t_1 and in t_2 and neither t_1 nor t_2 is x

3.4.3 t_1 is x, t_2 is not x and x occurs in t_2

3.4.4 t_1 is x, and x does not occur in t_2

3.4.5 t_1 is not x, x occurs in t_1, and it does not occur in t_2.

Symmetrical cases to these from (3.4.1) - (3.4.5) are omitted in the above list.

It is easy to check that in case (3.4.1), (3.4.2) P is constantly true when x gets any value from $|\underline{B}|$. Therefore we set in these cases $P^{(x)}$ to be *true*.

In case (3.4.3) - (3.4.5) P is constantly false for every value of x in $|\underline{B}|$, and we set $P^{(x)}$ to be *false*.

The reader can easily check that (i) and (ii) are satisfied. Inductive step being obvious is omitted.

Before proceeding to the last auxiliary result let us observe that due to symmetry of construction of $\underline{A} \oplus \underline{B}$ Lemmas 3.3 and 3.4 remain true when $\mathcal{L}_1$ is interchanged with $\mathcal{L}_2$ and A with B.

3.5 **Lemma**

For every first order formula P over the language of $\underline{A} \oplus \underline{B}$ there exist a nonnegative integer n, formulae $F_1, \ldots, F_n$ over $\mathcal{L}_1 \cup \{A,B\}$, and formulae $G_1, \ldots, G_n$ over $\mathcal{L}_2 \cup \{A,B\}$ such that

$$\underline{A} \oplus \underline{B} \models \bigwedge_{i=1}^{n} ((F_i)_A \vee (G_i)_B)$$

Proof: We prove the statement of Lemma 3.5 by induction on P.

Let P be of the form $t_1 = t_2$. Consider the following cases.

3.5.1 t_1 is a variable, say x. In t_2 there occur symbols from $\mathcal{L}_1$, and $\mathcal{L}_2$.

3.5.2 t_1 is over $\mathcal{L}_1$, t_2 is over $\mathcal{L}_2$, and both t_1 and t_2 are not variables.

3.5.3 t_1 is over $\mathcal{L}_1$, in t_2 there are symbols from $\mathcal{L}_1$ and $\mathcal{L}_2$.

3.5.4 t_1 and t_2 are over $\mathcal{L}_i \cup \{A,B,\perp\}$, where i = 1 or i = 2.

Cases symmetric to these listed above are omitted. Observe that if t_1 and t_2 satisfy (3.5.1), then $\underline{A} \oplus \underline{B} \models x = \perp \leftrightarrow P$. For (3.5.2) we have $\underline{A} \oplus \underline{B} \models P$

Suppose P satisfies (3.5.3), let X_i for i = 1.2 be the set of all variables which occur in t_i. If $X_1 = \varnothing$ or $X_2 = \varnothing$, then obviously $\underline{A} \oplus \underline{B} \models \neg P$, otherwise it is easy to see that $\underline{A} \oplus \underline{B} \models P \leftrightarrow [(\bigvee_{x_i \in X_1} x_i = \perp) \wedge (\bigvee_{x_i \in X_2} x_i = \perp)]$.

In case (3.5.4) obviously $\underline{A} \oplus \underline{B} \models P \leftrightarrow t_1 = \bot$ holds.

Observe now that in cases (3.5.1) - (3.5.5) P is equivalent over $\underline{A} \oplus \underline{B}$ to a formula over $\mathcal{L}_i \cup \{A,B,\bot\}$, where i = 1 or i = 2. Using Lemma 3.3 we obtain a desired decomposition of P.

Other cases of P being an atomic formula are dealt similarly and we omit them.

The only nontrivial case in the inductive step is P being of the form $\forall x\, Q$, where $\underline{A} \oplus \underline{B} \models Q \leftrightarrow \bigvee_{i=1}^{n} ((F_i)_A \vee (G_i)_B)$ for certain F_i's over $\mathcal{L}_1 \cup \{A,B\}$ and G_i's over $\mathcal{L}_2 \cup \{A,B\}$.

Since $\underline{A} \oplus \underline{B} \models (\forall x\, P) \leftrightarrow \bigvee_{i=1}^{n} \forall x((F_i)_A \vee (G_i)_B)$,

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 desired form. First we observe that such a formula is equivalent over $\underline{A} \oplus \underline{B}$ to the disjunction of these formulae

3.5.5 $((F_i)_A (\bot/x) \vee (G_i)_A (\bot/x)) \quad \wedge$

3.5.6 $\forall x\, [A(x) \Rightarrow ((F_i)_A \vee (G_i)_B)] \quad \wedge$

3.5.7 $\forall x\, [B(x) \Rightarrow ((F_i)_A \vee (G_i)_B)]$

Using Lemma 3.3 we transform formula (3.5.6) into an equivalent one which has a desired form. transformations of (3.5.7) and (3.5.8) are similar and we present here only a transformation of (3.5.7).

By Lemma 3.4 (3.5.7) is equivalent over $\underline{A} \oplus \underline{B}$ to

3.5.8 $\forall x\, [A(x) \Rightarrow ((F_i)_A \vee (G_i^{(x)})_B)]$

Since in $(G_i^{(x)})_B$ x does not occur free, (3.5.9) is equivalent over $\underline{A} \oplus \underline{B}$ to

$(\forall x\, F_i)_A \vee (G_i^{(x)})_B$.

This completes the proof of Lemma 3.5.

3.6 Proof of Proposition 3.2

Let P be a first order formula over the language of $\underline{A} \oplus \underline{B}$. By Lemma 3.5 it is equivalent over $\underline{A} \oplus \underline{B}$ to $\bigvee_{i=1}^{n} [(F_i)_A \vee (G_i)_B]$, 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. By Lemma 3.4 for every $1 \leq i \leq n$, $A(X) \wedge (G_i)_B$ has a constant true value over $\underline{A} \oplus \underline{B}$ (i.e. it is equivalent over $\underline{A} \oplus \underline{B}$ to a sentence. Let ϵ_i be *true* if $\underline{A} \oplus \underline{B} \models A(X) \wedge (G_i)_B$, and *false* otherwise obviously.

3.6.1 $\quad \underline{A} \oplus \underline{B} \models (A(X) \wedge P) \quad \leftrightarrow \quad A(X) \wedge \bigwedge_{i=1}^{n} ((F_i)_A \vee \epsilon_i)$.

Let F_i^* be obtained from F_i by replacing in F_i every subformula of the form A(t) by *true*, and every subformula of the form B(t) by *false*. It follows easily from (3.6.1) that we can take as the sought Q: $\bigwedge_{i=1}^{n} (F_i^* \vee \epsilon_i)$. This completes the proof of proposition 3.2.

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" (to appear in TCS. An earlier edition of this paper is registered at the Mathematical Centre as report IW 136/80).

2. Bergstra, J.A. & J.F. Tucker, "Expressiveness and the completeness of Hoare's logic", Mathematical Centre Report IW 143/80.

3. Bergstra, J.A., & J.V. Tucker, "Two theorems on the completeness of Hoare's Logic", Mathematical Centre Report IW?/81.

4. Cook, S.A., "Soundness and completeness of an axiom system for program verification", SIAM J. Computing ? (1978) 70-90.

5. Harel, D., "First order dynamic logic", Lecture notes in Computer Science 68, Springer 1978.

6. Wand, M., "A new incompleteness result for Hoare's system", JACM 25(1978) 168-175.