A CHARACTERIZATION OF PROGRAM EQUIVALENCE IN TERMS OF HOARE'S LOGIC

J.A. Bergstra
Institute of Applied Mathematics
and Computer Science
University of Leiden
Leiden, The Netherlands

J. Terlouw
Mathematical Institute
University of Utrecht,
Utrecht, The Netherlands

Abstract

We discuss the equivalence of simple while-programs S_1 and S_2 in all datatypes that implement a specification (Σ,E). A sufficient condition is that S_1 and S_2 are indistinguishable in all program logics $HL(\Sigma',E)$ for $\Sigma' \supseteq \Sigma$. A necessary condition is that for each refinement (Σ',E') of (Σ,E) another refinement (Σ^*,E^*) of (Σ',E') exists such that S_1 and S_2 *cannot be distinguished in* $HL(\Sigma^*,E^*)$.

TECHNICAL INTRODUCTION

1.1. *Program equivalence derives its interest both from a fundamental theoretical point* of view (MH[7]) and from its role in a theory of program transformation. Especially in the case of transformations that replace a complicated, high-level program, S, by a simpler, lower-level but equivalent one, the correctness proof of such a transformation T must establish some kind of equivalence between S and T(S). In the present paper we concentrate on one possible form of equivalence: behaving equivalent w.r.t. Hoare's logic.

1.2. *At this stage it is necessary to be precise about the main parameters that govern* this situation. These are, in order of importance:

a) The program language. We will use while-programs over various finite signatures Σ. $WP(\Sigma)$ is the smallest class of programs, working on registers (variables) $x_0,x_1,x_2,\ldots$ that contains assignments $x_i := \tau$ with τ an expression (term) in $L(\Sigma)$, and which is closed under composition, the if b then else fi and the while b do od constructs. Here b denotes a boolean expression (quantifier free formula) of $L(\Sigma)$, *the first oder logic of signature* Σ.

b) The data structures on which to run our programs. Here each Σ'-algebra with $\Sigma' \supseteq \Sigma$, is a conceivable datastructure on which $S \in WP(\Sigma)$ can run.
In general, however, we work relative to a specification E, a theory in $L(\Sigma)$. Semantic program equivalence is then expressed as follows: for $S_1,S_2 \in WP(\Sigma)$, E a Σ-specification we write $S_1 \equiv_{ALG(\Sigma,E)} S_2$ if S_1 and S_2 compute the same partial function $\mathfrak{A}^n \to \mathfrak{A}^n$ on each $\mathfrak{A} \in ALG(\Sigma,E)$, i.e. on the collection of structures that satisfies the requirements of specification (Σ,E). The partial function: $\mathfrak{A}^n \to \mathfrak{A}^n$ that S defines on $\mathfrak{A}$ is defined using a standard operational semantics. In this

case we must choose n as the number of variables occurring in S_1 or in S_2.
For technical reasons we must assume in proposition 2 and in theorem 1 that (Σ,E) has no finite models.

c) The precise version of Hoare's logic that is used. For each specification (Σ,E), a logic $HL(\Sigma,E)$ is introduced. This is exactly standard Hoare's logic for proving correctness assertions of the form $\{p\}\ S\ \{q\}$ with $p,q \in L(\Sigma)$ $S \in WP(\Sigma)$ where in applications of the rule of consequence only those implications $r \rightarrow s$ can be used that are derivable from E.
A detailed account of $HL(\Sigma,E)$ can be found in Apt [1], de Bakker [3] or B & T [5].
In [5] a deduction theorem for HL is proved, using a quite straightforward induction on proof length, which we will need. It reads as follows:
for each closed assertion t, for all p,q,S:

$$HL(\Sigma,E \cup \{t\}) \vdash \{p\}\ S\ \{q\} \Leftrightarrow$$
$$HL(\Sigma,E) \vdash \{p \wedge t\}\ S\ \{q\}$$

d) Important notational conventions are the following: let (Σ,E) be a specification, a *refinement of* (Σ,E) *is a specification* (Σ',E') *with* $\Sigma' \supseteq \Sigma$ *and* $E' \vdash E$. We will usually assume that E' is consistent whenever E is. A refinement (Σ',E') of (Σ,E) is conservative if for all $p \in L(\Sigma)$: $E' \vdash p$ implies $E \vdash p$. Here is our main technical definition:

$$S_1 \equiv_{HL(\Sigma,E)} S_2 \text{ if for all } p,q$$
$$HL(\Sigma,E) \vdash \{p\}\ S_1\ \{q\} \Leftrightarrow HL(\Sigma,E) \vdash \{p\}\ S_2\ \{q\}.$$

SURVEY OF OUR RESULTS

The following results will be established, all having to do with the relationship between $\equiv_{HL(\Sigma,E)}$ and $\equiv_{ALG(\Sigma,E)}$.

PROPOSITION 1. There are a signature and programs $S_1,S_2 \in WP(\Sigma)$ such that $S_1 \equiv_{ALG(\Sigma,\phi)} S_2$ but $s_1 \not\equiv_{HL(\Sigma,\phi)} S_2$.
This solves a problem posed by J. Tiuryn [8].

PROPOSITION 2. If $S_1 \equiv_{ALG(\Sigma,E)} S_2$ then there is a conservative refinement (Σ',E') of (Σ,E) such that $S_1 \equiv_{HL(\Sigma',E')} S_2$.

THEOREM 1. CHARACTERIZATION OF PROGRAM EQUIVALENCE.

For each specification (Σ,E) and all pairs of programs $S_1,S_2 \in WP(\Sigma)$ the following are equivalent:

i) $S_1 \equiv_{ALG(\Sigma,E)} S_2$

ii) for each refinement (Σ',E') of (Σ,E) there is a refinement (Σ^*,E^*) of (Σ',E') such that
$S_1 \equiv_{HL(\Sigma^*,E^*)} S_2$.

THEOREM 2. A sufficient condition for $S_1 \equiv_{ALG(\Sigma,E)} S_2$ is that for all Σ' extending Σ:

$$S_1 \equiv_{HL(\Sigma',E)} S_2$$

We have the impression that this criterion could well be the basis for a formal proof system for program equivalence. This will be a subject of further work by the first author and J.W. Klop (Mathematical Center, Amsterdam).

RELATIONS TO PREVIOUS LITERATURE

Let us introduce the notation $S_1 \equiv_{PC(\Sigma,E)} S_2$ to express that for all $p,q \in L(\Sigma)$:
$ALG(\Sigma,E) \models \{p\}\ S_1\ \{q\} \Leftrightarrow ALG(\Sigma,E) \models \{p\}\ S_2\ \{q\}$.
MH [7] and BTT [4] study the relationship between $\equiv_{ALG(\Sigma,E)}$ and $\equiv_{PC(\Sigma,E)}$.
Obviously $\equiv_{ALG(\Sigma,E)} \subseteq \equiv_{PC(\Sigma,E)}$ and, depending on (Σ,E) the inverse inclusion may or may not hold. If $HL(\Sigma,E)$ is complete, i.e. for all p,q,S $HL(\Sigma,E) \vdash \{p\}\ S\ \{q\} \Leftrightarrow ALG(\Sigma,E) \models \{p\}\ S\ \{q\}$ the relations $\equiv_{HL(\Sigma,E)}$ and $\equiv_{PC(\Sigma,E)}$ coincide.
In general, however, the relations are not so clear, because, according to Wand [9] $HL(\Sigma,E)$ may be incomplete. Semantic work on program equivalence occurs for instance in De Bakker [2].

SKETCHES OF PROOFS

5.1. Let Σ be the signature $\{0,S,P\}$. Then we can construct the following programs S_0,S',S'',S_1,S_2 :
S_0 = y := 0; <u>while</u> y ≠ x <u>do</u> <u>if</u> y = PS(y) <u>then</u> y := S(y) <u>else</u> DIV <u>fi</u> <u>od</u>.
S' = y := 0; <u>while</u> x ≠ 0 <u>do</u> y := S(y); x := P(x) <u>od</u>
S'' = y := x; x := 0
$S_1 = S_0;\ S'$ $S_2 = S_0;\ S''$.
Here DIV = <u>while</u> x = x <u>do</u> x := x <u>od</u>.
First observe that $S_1 \equiv_{ALG(\Sigma,\phi)} S_2$. To see this note that $S_0(x,y)\downarrow$ if for some n $x = S^n(0)$ and for all $m < n$ $PS^{m+1}(0) = S^m(0)$. Then, however, S_1 just like S_2 results giving y the value of x and putting x := 0. Then we consider $HL(\Sigma,\phi)$. It is obvious

that

$$HL(\Sigma,\phi) \vdash \{x=z\}\ S_2\ \{x=0 \wedge y=z\}$$

On the other hand $HL(\Sigma,\phi) \nvdash \{x=z\}\ S_1\ \{x=0 \wedge y=z\}$.
To prove this consider the structure $A = (\mathbb{Z},S,P,0)$. If
$HL(\Sigma,\phi) \vdash \{x=z\}\ S_1\ \{x=0 \wedge y=z\}$ then also $HL(\Sigma,Th(A))$ proves this fact.
Suppose that $r(x,y,z)$ is the invariant of the while loop in S' (in this second proof), then one can show that, in A,r defines the predicate $x+y = z$.
This, however, cannot be done by a first order formula of $L(\Sigma)$.

This proves proposition 1. In this situation we can already observe a version of the phenomenon indicated in proposition 2. Let $\Sigma' = \Sigma \cup \{+\}$ and
$E' = \{x+0 = x,\ 0+x = x,\ x \neq 0 \quad P(x) + S(y) = x+y\}$ plus all induction axioms. Then with some work one can show $S_1 \equiv_{HL(\Sigma',E')} S_2$.

5.2. For further proofs we need a lemma that is immediate from B & T [6].
Assume that (Σ,E) is a specification without finite models.
Let Γ_ω be the *signature of* $\mathbb{N} = (\omega,+,\cdot,0,1)$. *With* Σ_ω *we denote a disjoint union of* Σ and Γ_ω. We denote with E_ω the theory E plus Peano arithmetic, over Γ_ω, plus all induction axioms of $L(\Sigma_x)$. First of all we observe that (Σ_ω,E_ω) is conservative over (Σ,E)
To prove this assume that $\phi \in L(\Sigma)$, ϕ closed, and $E \nvdash \phi$; according to the completeness theorem there is a countable model A of $E \cup \{\neg\phi\}$. Expanding A to a Σ_ω-structure A' such that $A'|\Gamma_\omega$ is a standard model of arithmetic, A' satisfies E_ω and $\neg\phi$, thus $E_\omega \nvdash \phi$.

5.3. LEMMA. For all p,S there is a formula SP(p,S) in $L(\Sigma_\omega)$ such that the following properties hold:

- i) $HL(\Sigma_\omega,E_\omega) \vdash \{p\}\ S\ \{SP(p,S)\}$
- ii) for all q: $HL(\Sigma_\omega,E_\omega) \vdash \{p\}\ S\ \{q\}$ if and only if $E_\omega \vdash SP(P,S) \rightarrow q$.
- iii) if $A \in ALG(\Sigma_\omega,E_\omega)$ and $A|\Gamma_\omega \cong \mathbb{N}$ then SP(p,S) defines in A the strongest postconditon w.r.t. p of S.
- iv) $E_\omega \vdash \forall(p \leftrightarrow q) \rightarrow (SP(p,S) \leftrightarrow SP(q,S))$. Here $\forall(p \leftrightarrow q)$ is the universal closure of $p \leftrightarrow q$.
- v) if $x \notin VAR(S)$ then $E_\omega \vdash SP(\exists xp,S) \leftrightarrow \exists xSP(p,S)$
- vi) if $FV(q) \cap VAR(S) = \emptyset$ then

  $E_\omega \vdash SP(p \wedge q,S) \leftrightarrow q \wedge SP(p,S)$.

5.4. Suppose $S_1 \equiv_{ALG(\Sigma,E)} S_2$. With θ we denote the universal closure of the formula $SP(X = Z,S_1) \leftrightarrow SP(X = Z,S_2)$, where X is a list containing $VAR(S_1) \cup VAR(S_2)$ and Z is a list of variables disjoint from X. Now $E_\omega \cup \{\theta\}$ is conservative over E as well because the model A' constructed in 5.2 satisfies θ in view of property iii) of the

SP construct and the operational equivalence of S_1 and S_2 in A.

At this stage proposition 2 is proved by taking $\Sigma' = \Sigma_\omega$ and $E' = E_\omega \cup \{\theta\}$. We find that for all p,q:

$$HL(\Sigma',E') \vdash \{p\}\ S_1\ \{q\} \Leftrightarrow \text{(by the deduction lemma)}$$
$$HL(\Sigma_\omega,E_\omega) \vdash \{\theta \wedge p\}\ S_1\ \{q\} \Leftrightarrow \text{(by property II))}$$
$$E_\omega \vdash SP(\theta \wedge p,S_1) \rightarrow q \quad \Leftrightarrow \text{(by vi))}$$
$$E_\omega \vdash \theta \wedge SP(p,S_1) \quad \Leftrightarrow \text{(by iv))}$$
$$E_\omega \vdash \theta \wedge SP(\exists Z(X=Z \wedge p[X/Z]),S_1) \Leftrightarrow \text{(by v))}$$
$$E_\omega \vdash \theta \wedge \exists Z\ SP(X=Z \wedge p[X/Z]),S_1) \Leftrightarrow \text{(by vi))}$$
$$E_\omega \vdash \theta \wedge \exists Z(p[X/Z] \wedge SP(X=Z,S_1)) \Leftrightarrow \text{(by assumption)}$$
$$E_\omega \vdash \theta \wedge \exists Z(p[X/Z] \wedge SP(X=Z,S_2)) \Leftrightarrow$$

$HL(\Sigma',E') \vdash \{p\}\ S_2\ \{q\}$ by arguing backwards in a symmetrical way.

5.5. The i) $\Rightarrow$ ii) part of 1 now follows by observing that $S_1 \equiv_{ALG(\Sigma,E)} S_2$ implies $S_1 \equiv_{ALG(\Sigma',E')} S_2$ and then taking $\Sigma^* = \Sigma'_\omega$ and $E^* = E'_\omega \cup \{\theta\}$, repeating the above argument for (Σ^*,E^*).

5.6. At last we arrive at proving the negative parts, i.e. those working from the assumption $S_1 \not\equiv_{ALG(\Sigma,E)} S_2$. For theorem 2 we have to find an extension signature Σ' such that $S_1 \not\equiv_{HL(\Sigma',E)} S_2$; and for the ii) $\Rightarrow$ i) part of theorem 1 we must find a refinement (Σ',E') such that for each refinement (Σ^*,E^*) of (Σ',E'), $S_1 \not\equiv_{HL(\Sigma^*,E^*)} S_2$.

Let us assume that for some $A \in ALG(\Sigma,E)$ and some vectors a and b of elements of A, $A \models S_1[a] = b$ and $A \not\models S_2\ [a] = b$. (The symmetric other case is dealt with similarly.) Extend Σ to Σ^0 by adding constant names for the a's and b's, $\Sigma = \Sigma^0 \cup \{\underline{a},\underline{b}\}$. A^0 is the corresponding Σ^0-structure. Then $A^0 \models S_1\ [\underline{a}] = \underline{b}$ but $A^0 \not\models S_2[\underline{a}] = \underline{b}$. Now choose a closed assertion t of $L(\Sigma^0)$, true in A^0 such that $t \models S_1[\underline{a}] = \underline{b}$ (the construction of such t is quite straightforward, see BTT[4] for instance). Observe that $A \models \{X = \underline{a}\}S_2\{\neg(X = \underline{b})\}$, with X a vector of variables.

Take $\Sigma' = \Sigma^0$ and

$$E' = E_\omega \cup \{t\} \cup SP(X = \underline{a},\ S_2) \rightarrow \neg(X = \underline{b}).$$

Then (Σ',E') is a (consistent) refinement of (Σ,E).
Suppose now that (Σ^*,E^*) refines (Σ',E'), then clearly $HL(\Sigma^*,E^*) \vdash \{X = \underline{A}\}\ S_2\{\ (X = \underline{b})\}$. But because $E^* \vdash t$, $ALG(\Sigma^*,E^*) \not\models \{X = \underline{a}\}\ S_1\ \{\neg(X = \underline{b})\}$. This proves $S_1 \not\equiv_{HL(\Sigma^*,E^*)} S_2$ and theorem 1.

5.7. To obtain theorem 2 choose a closed assertion r such that $E' \vdash r$ and

$HL(\Sigma',E \cup \{r\}) \vdash \{X = \underline{a}\}\ S_2\ \{\neg(X = \underline{b})\}$. This r exists due to the (obvious) finiteness lemma from BT [5]. By the deduction lemma mentioned in the introduction this gives $HL(\Sigma',E) \vdash \{X = \underline{a} \wedge r\}\ S_2\ \{\neg(X = \underline{b})\}$; but if $HL(\Sigma',E) \vdash \{X = \underline{a} \wedge r\} S_1\ \{\neg(X = \underline{b})\}$ then arguing the other way around $HL(\Sigma',E') \vdash \{X = \underline{a}\}\ S_1\ \{\neg(X = \underline{b})\}$ *which is not true, because of soundness.*

This ends our proofs. It will not be so easy to extend these results to more complicated languages, including recursion and concurrency, because Lemma 1 depends heavily on the 'algebraic' structure of $HL(\Sigma,E)$ for $WP(\Sigma)$. But it is in fact quite likely that similar results can be obtained for each sound and relatively complete HL proof system.

REFERENCES

[1] K.R. Apt, Ten years of Hoare's Logic, a survey, F.V. Jensen, B.H. Mayoh & K.K. Møller (eds.). Proceedings from 5th Scandinavian Logic Symposium, Aalborg University Press, Aalborg 1979.

[2] J.W. de Bakker, Recursive procedures, Mathematical Centre Tracts 24, Math. Centre Amsterdam 1973.

[3] J.W. de Bakker, Mathematical Theorey of Program Correctness, Prentice Hall International, London 1980.

[4] J.A. Bergstra, J. Tiuryn & J.V. Tucker, Floyd's principle, correctnes theories and program equivalence, Math. Centre report IW 145/80, to appear in T.C.S.

[5] J.A. Bergstra & J.V. Tucker, On the refinement of specifications and Hoare's logic, Math. Centre report IW 155, Amsterdam 1980.

[6] J.A. Bergstra & J.V. Tucker, Peano's arithmetic and Hoare's logic, Math. Centre report IW 160, Amsterdam 1981.

[7] A.R. Meyer & J.Y. Halpern, Axiomatic definitions of programming languages, a theoretical assesment. Proceedings 6th POPL 1979.

[8] J. Tiuryn, private communication, november 1980.

[9] M. Wand, A new incompleteness result for Hoare's system, J.A.C.M. 25 (1) 1978.