Theoretical Computer Science 28 (1984) 215–221
North-Holland

NOTE

HOARE'S LOGIC FOR PROGRAMMING LANGUAGES WITH TWO DATA TYPES*

J.A. BERGSTRA**
Department of Computer Science, Mathematical Centre, 1098 SJ Amsterdam, The Netherlands

J.V. TUCKER
Department of Computer Studies, University of Leeds, Leeds LS2 9JT, United Kingdom

Communicated by E. Engeler
Received September 1982
Revised May 1983

Abstract. We consider the completeness of Hoare's logic with a first-order assertion language applied to **while**-programs containing variables of two (or more) distinct types. Whilst Cook's completeness theorem generalizes to many-sorted interpretations, certain fundamentally important structures turn out not to be expressive. We study the case of programs with distinguished counter variables and Boolean variables adjoined; for example, we show that adding counters to arithmetic destroys expressiveness.

Key words. Hoare's logic, partial correctness, **while**-programs, completeness, expressiveness, many-sorted programs, many-sorted first-order logic.

Introduction

Since the publication of [6] there has accumulated a large body of knowledge about proof systems for formally verifying the partial correctness of programs. Proof systems have been made which include a wide variety of programming features and, in particular, the soundness and completeness of these systems have been successfully analysed along the lines first set down in [5]. To obtain information about what has been achieved, at least for the sequential control aspects of programming languages, see [1].

In this note we consider a simple feature of most programming languages which has gone unnoticed to date, namely the property that *there may be two (or more) distinct types of variable or identifier in a single program.* We demonstrate that whilst

* Most of the work for this paper was performed in the course of two visits to the Mathematical Centre by the second author in February and July 1982.

Cook's account of completeness generalizes to include Boolean variables, it is, surprisingly, unable to cope with **while**-programs with counters.

In Section 1 we summarize prerequisites and observe that Cook's completeness theorem for Hoare's logic for **while**-programs applied to first-order expressive structures generalizes to the many-sorted case. However, in Section 2, we prove that adding arithmetic N to an expressive structure A can lead to a non-expressive two-sorted interpretation $[A, N]$. In particular, we prove that adding arithmetic N to arithmetic N leads to a non-expressive structure $[N, N]$ and, indeed, that Hoare's logic for $[N, N]$ is incomplete (Theorem 2.3). Thus, there is a general completeness theorem for the two-type situation, but it cannot be applied to a canonical example.

1. Assertions, programs and Hoare's logic

In addition to necessary prerequisites about two-sorted syntax and semantics, we outline the fate of Cook's study [6] of Hoare's logic when generalized to the two-sorted situation as this is the background of our main results.

Syntax

The first-order language $L(\Sigma)$ of some two-sorted signature Σ is based upon two sets of variables

$$x_1^1, x_2^1, \ldots \quad \text{and} \quad x_1^2, x_2^2, \ldots,$$

of sorts 1 and 2 respectively, and the constant, function and relation symbols of $L(\Sigma)$ are those of Σ together with equality symbols of sorts 1 and 2.

The usual inductive definition of term now yields two kinds of term giving values of sort 1 and sort 2. Atomic formulae have the form

$$t^i =_i s^i \quad \text{and} \quad R(y_1^{i_1}, y_2^{i_2}, \ldots, y_k^{i_k})$$

where t^i, s^i are terms (having values) of sort i, $=_i$ is the equality symbol for sort i, R is a relation symbol and the $y_j^{i_j}$ are variables of sort i_j, $j = 1, \ldots, k$ and $i, i_j \in \{1, 2\}$.

The well-formed formulae of $L(\Sigma)$ are made inductively by applying the logical connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$ and the quantifiers

$$\forall x_j^1 \quad \exists x_j^1 \quad \forall x_j^2 \quad \exists x_j^2 \qquad j \in \mathbb{N}$$

in the usual way.

Using the syntax of $L(\Sigma)$ the set $\mathrm{WP}(\Sigma)$ of all **while**-programs over Σ is defined in the obvious way. Note, in particular, that there are two kinds of assignment statement

$$x_j^1 := t^1 \quad \text{and} \quad x_j^2 := t^2$$

but that Boolean tests in control statements are simply quantifier-free formulae of $L(\Sigma)$ and may refer to both sorts.

By a *specified* or *asserted program* we mean a triple of the form $\{p\}S\{q\}$ where $p, q \in L(\Sigma)$ and $S \in \mathrm{WP}(\Sigma)$.

Semantics

The semantics of $L(\Sigma)$ is based on two-sorted structures A of signature Σ and is formally defined in the usual manner.

The set of all sentences of $L(\Sigma)$ which are true in structure A is called the first-order theory of A and is denoted $\mathrm{Th}(A)$. For $\phi \in L(\Sigma)$ the set defined in A by ϕ we denote $\phi[A]$.

For the semantics of $\mathrm{WP}(\Sigma)$ on an interpretation A we leave the reader free to choose any sensible account of **while**-program computation in one-sorted structures and then to generalize it. Certainly, the operational and denotational semantics given in [2] have natural many-sorted generalizations (see [8]).

We suppose that the meaning of $S \in \mathrm{WP}(\Sigma)$ on interpretation A is defined as a state transformation

$$M_A(S): \text{STATES}(A) \to \text{STATES}(A).$$

Also if S has n variables of sort 1 and m variables of sort 2, then $\text{STATES}(A) \cong A_1^n \times A_2^m$, where A_1, A_2 are the domains of sorts 1, 2 in A, and we suppose that $M_A(S)$ is represented by a mapping

$$\hat{M}_A(S): A_1^n \times A_2^m \to A_1^n \times A_2^m.$$

Putting together the semantics of $L(\Sigma)$ and $\mathrm{WP}(\Sigma)$ we consider the partial correctness semantics of the specified programs: $\{p\}S\{q\}$ is valid on A, written $A \models \{p\}S\{q\}$, if when p is true, then either S diverges or S converges to a state at which q is true. The set of all specified programs valid on A is called the partial correctness theory of A and we write

$$\mathrm{PC}(A) = \{\{p\}S\{q\}: A \models \{p\}S\{q\}\}.$$

Hoare's logic

Hoare's logic for the two-sorted $\mathrm{WP}(\Sigma)$ has exactly the same axiom scheme for assignment statements and the same rules for composition, conditionals and iteration. In addition, any first-order theory T may be employed to prove a specification for the underlying data types and T affects program correctness proofs via the Rule of Consequence (see [5, 6]). The set of all specified programs provable from T is denoted $\mathrm{HL}(T)$.

In this note we are interested in proving correctness with respect to a given two-sorted structure A. Cook's work on the single-sorted version of this case generalizes to provide us with the following account.

1.1. Soundness Theorem. *If* $A \models T$, *then* $\mathrm{HL}(T) \subset \mathrm{PC}(A)$.

The assertion language $L(\Sigma)$ is said to be expressive for $\mathrm{WP}(\Sigma)$ over A if for any $p \in L(\Sigma)$ and $S \in \mathrm{WP}(\Sigma)$ there is a formula $\mathrm{SP}(p, S) \in L(\Sigma)$ that defines the strongest postcondition $\mathrm{SP}_A(p, S)$ of S with respect to p over A,

$$\mathrm{SP}_A(p, S) = \{\sigma \in \text{STATES}(A)\colon \exists\tau\, [M_A(S)(\tau)\downarrow\sigma \,\&\, p(\tau)]\}.$$

Notice that expressiveness is actually a property of the interpretation A rather than $L(\Sigma)$. We call HL complete for A if $\mathrm{HL}(\mathrm{Th}(A)) = \mathrm{PC}(A)$.

1.2. Cook's Completeness Theorem. *Suppose $L(\Sigma)$ is expressive for $\mathrm{WP}(\Sigma)$ over A and let $T = \mathrm{Th}(A)$. Then $\mathrm{HL}(T) = \mathrm{PC}(A)$.*

In view of Theorem 1.2 we define $\mathrm{HL}(A) = \mathrm{HL}(\mathrm{Th}(A))$, and observe that $\mathrm{HL}(A)$ represents the strongest Hoare logic for analyzing correctness on A because it is equipped with all first-order true facts about A.

1.3. Theorem. *If A is finite, then A is expressive and $\mathrm{HL}(A)$ is complete.*

2. Adding arithmetic

Semantically, adding counters to **while**-programs is effected by interpreting the two-sorted programming language $\mathrm{WP}(\Sigma)$ on certain two-sorted structures of the following form.

Let A and B be single-sorted structures with disjoint signatures Σ_A and Σ_B respectively. Then we define the *join* $[A, B]$ of A and B to be the two-sorted structure of signature $\Sigma_{A,B} = \Sigma_A \cup \Sigma_B$ whose disjoint domains and operations are simply those of A and B.

What is noteworthy in this operation on structures is that algebraically A and B remain independent data types. Adding arithmetic means computing on structures $[A, \mathbb{N}]$ where $\mathbb{N}$ is the standard model of arithmetic. Adding Booleans means computing on structures $[A, \mathbb{B}]$ where $\mathbb{B} = \{\mathrm{tt}, \mathrm{ff}\}$ equipped with $\wedge$, $\neg$.

We prove that Hoare's logic is incomplete when applied to structures $[A, \mathbb{N}]$.

2.1. Proposition. *If $[A, B]$ is expressive, then A and B are expressive.*

Proof. We begin by stating a basic fact about first-order definability on $[A, B]$.

Let H be the smallest set of $\Sigma_{A,B} = \Sigma_A \cup \Sigma_B$ formulae that contains $L(\Sigma_A)$ and $L(\Sigma_B)$ and is closed under $\neg$, $\wedge$, $\vee$. Thus, H does *not* contain formulae with quantifiers ranging over different sorts such as

$$\forall x^A\, (\phi^A \wedge \phi^B).$$

2.2. Separation of Variables Lemma. *Each formula $\phi \in L(\Sigma_{A,B})$ is equivalent to a formula of H.*

Proof. The proof follows by induction on the structure of ϕ (see [3]). $\square$

Proof of Proposition 2.1 (*continued*). To prove the proposition we assume $[A, B]$ is expressive and prove that A is expressive (the case for B follows mutatis nomine).

Let $\phi \in L(\Sigma_A)$ and $S \in \mathrm{WP}(\Sigma_A)$. Let $\mathrm{SP}(\phi, S)$ define the strongest postcondition $\mathrm{SP}_{[A,B]}(\phi, S)$ on $[A, B]$. By the Separation of Variables Lemma 2.2,

$$\mathrm{SP}(\phi, S) \equiv \bigvee_{i=1}^{s} (\psi_i^A \wedge \psi_i^B).$$

Because ϕ and S involve variables of type A only, the components ψ_i^B for $1 \leq i \leq s$ are closed and can be replaced by their propositional values **true** and **false**. This being done we obtain a formula $\psi \in L(\Sigma_{A,B})$, equivalent to $\mathrm{SP}(\phi, S)$, that is first-order over Σ_A and, indeed, ψ defines $\mathrm{SP}_A(\phi, S)$ on A. $\square$

Our main result implies that the converse of Proposition 2.1 is false. Let $\mathbb{N}$ denote standard model of arithmetic; to be precise let

$$\mathbb{N} = (\{0, 1, \ldots\}, 0, 1, x+1, x \dot{-} 1, x+y, x \cdot y).$$

Consider the structure $[N_1, N_2]$ of signature $\Sigma_{1,2}$ wherein $N_1 = \mathbb{N}$ has signature Σ_1 and $N_2 = \mathbb{N}$ has signature Σ_2, i.e., $[N_1, N_2]$ is a pair of algebraically independent copies of $\mathbb{N}$. We are looking at the case of adding arithmetic to arithmetic, so to say.

2.3. Theorem. *The two-sorted structure* $[N_1, N_2]$ *is not expressive and* $\mathrm{HL}([N_1, N_2])$ *is not complete.*

Proof. Consider the following program:

$$\begin{aligned} S ::= \quad & x := 0;\ z := 0; \\ & \textbf{while } x \neq y \textbf{ do } x := x+1;\ z := z+1 \textbf{ od} \end{aligned}$$

with x, y variables of sort 1 and z a variable of sort 2. The strongest post-condition of S with respect to **true** is

$$\mathrm{SP}(\mathbf{true}, S) = \{(a, b, c) \in N_1 \times N_1 \times N_2 \colon a = b = c = n \in \mathbb{N}\}.$$

Suppose $\mathrm{SP}(\mathbf{true}, S)$ is first-order definable over $[N_1, N_2]$; then clearly the 'diagonal' $\Delta = \{(a, b) \in N_1 \times N_2 \colon a = b = n \in \mathbb{N}\}$ is first-order definable: to this latter statement we derive a contradiction.

By the Separation of Variables Lemma 2.2, it is sufficient to show that Δ is not definable by a formula of $H(\Sigma_{1,2})$.

Suppose as a contradiction that Δ is definable by $\phi \in H(\Sigma_{1,2})$ with free variables x, y of sorts 1, 2; thus,

$$\Delta = \{(a, b) \in N_1 \times N_2 \colon [N_1, N_2] \models \phi(a, b)\}.$$

Now ϕ can be written in disjunctive normal form:

$$\phi \equiv \bigvee_{i=1}^{s} \bigwedge_{j=1}^{t} \phi_{i,j},$$

where $\phi_{i,j} \in L(\Sigma_1) \cup L(\Sigma_2)$ for $1 \leq i \leq s$ and $1 \leq j \leq t$. This can be compressed to

$$\phi \equiv \bigvee_{i=1}^{s} (\Phi_i^1 \wedge \Phi_i^2),$$

where $\Phi_i^1 \in L(\Sigma_1)$ and $\Phi_i^2 \in L(\Sigma_2)$ with free variables x and y respectively. For $1 \leq i \leq s$, set

$$\Delta_i = \{(a, b) \in N_1 \times N_2 : [N_1, N_2] \models \Phi_i^1(a) \wedge \Phi_i^2(b)\},$$

so that $\Delta = \bigcup_{i=1}^{s} \Delta_i$. At least one Δ_i is infinite, say Δ_0. We choose two points (a, a), $(b, b) \in \Delta_1$ with $a \neq b$. Now

$$[N_1, N_2] \models \Phi_0^1(a) \wedge \Phi_0^2(a) \quad \text{and} \quad [N_1, N_2] \models \Phi_0^1(b) \wedge \Phi_0^2(b).$$

Thus,

$$[N_1, N_2] \models \Phi_0^1(a) \wedge \Phi_0^2(b).$$

This means that $(a, b) \in \Delta_0 \subset \Delta$ which is not the case. Therefore, $[N_1, N_2]$ is not expressive.

In order to see that HL($[N_1, N_2]$) is not complete, consider the program

$$S_2 ::= \textbf{while } x \neq 0 \wedge y \neq 0 \wedge z \neq 0$$
$$\textbf{do } x := x \dot{-} 1;\ y := y \dot{-} 1;\ z := z \dot{-} 1 \textbf{ od}.$$

Clearly,

$$[N_1, N_2] \models \{\textbf{true}\}\ S_1\ ;\ S_2\ \{x = 0 \wedge y = 0 \wedge z = 0\}.$$

In order to prove this valid asserted program using Hoare's logic, an intermediate assertion θ must be found, i.e., a formula such that

$$[N_1, N_2] \models \{\textbf{true}\}\ S_1 \{\theta\}, \qquad [N_1, N_2] \models \{\theta\}\ S_2 \{x = 0 \wedge y = 0 \wedge z = 0\}.$$

Thus,

$$\mathrm{SP}_{[N_1, N_2]}(\textbf{true}, S_1) \subset \theta[N_1, N_2] \subset \mathrm{WP}_{[N_1, N_2]}(S_2, x = 0 \wedge y = 0 \wedge z = 0).$$

But

$$\mathrm{WP}_{[N_1, N_2]}(S_2, x = 0 \wedge y = 0 \wedge z = 0) = \mathrm{SP}_{[N_1, N_2]}(\textbf{true}, S_1)$$

and hence $\theta[N_1, N_2] = \mathrm{SP}(\textbf{true}, S_1)$. This contradicts the fact that $\mathrm{SP}(\textbf{true}, S_1)$ is not definable. $\square$

3. Concluding remarks

Quite clearly no useful account of the correctness of many-typed programs can be founded on a first-order assertion language. Fortunately, it is possible to give a very thorough theory of the partial and total correctness of the basic sequential constructs in a many-sorted abstract setting if one allows the extension to a weak second-order assertion language (see [8]). Moreover, allowing hidden functions to enhance expressiveness is certainly an acceptable step; for initial algebra specification it is required.

In contrast to Theorem 2.3 one can show the following theorem.

Theorem. *If A is expressive and F is finite, then $[A, F]$ is expressive and consequently* $\mathrm{HL}([A, F])$ *is complete.*

Finally it should be pointed out that, in logic, preservation theorems are known for products (cf. the theorem of Feferman and Vaught as in [7]); such properties still have to be established for program verification logics.

Acknowledgment

We wish to thank E.R. Olderog for useful discussions on the subject matter of this note.

References

[1] K.R. Apt, Ten years of Hoare's Logic: A survey—Part 1, *ACM Trans. Programming Languages and Systems* **3** (1981) 431–483.
[2] J.W. de Bakker, *Mathematical Theory of Program Correctness* (Prentice-Hall, Englewood Cliffs, NJ, 1980).
[3] J.A. Bergstra, A. Chmielienska and J. Tiuryn, Another incompleteness theorem for Hoare's logic, *Inform. Control* **52** (2) (1982) 159–171.
[4] J.A. Bergstra and J.V. Tucker, Expressiveness and the completeness of Hoare's logic, *J. Comput. System Sci.* **25** (1983) 267–284.
[5] S.A. Cook, Soundness and completeness of an axiom system for program verification, *SIAM J. Comput.* **7** (1978) 70–90; Corrigendum, *SIAM J. Comput.* **10** (1981) 612.
[6] C.A.R. Hoare, An axiomatic basis for computer programming, *Comm. ACM* **12** (1969) 576–580.
[7] J.D. Monk, *Mathematical Logic* (Springer, Berlin, 1976).
[8] J.V. Tucker and J.I. Zucker, Program correctness over abstract data types, with error-state semantics, Monograph, in preparation.