Theoretical Computer Science 17 (1982) 303-315
North-Holland Publishing Company

SOME NATURAL STRUCTURES WHICH FAIL TO POSSESS A SOUND AND DECIDABLE HOARE-LIKE LOGIC FOR THEIR WHILE-PROGRAMS

J.A. BERGSTRA
Department of Computer Science, University of Leiden, 2300RA Leiden, The Netherlands

J.V. TUCKER
Department of Computer Science, Mathematical Centre, 1098SJ Amsterdam, The Netherlands

Communicated by J. de Bakker
Received May 1980
Revised October 1980

Introduction

With the term *Hoare-like logic* we have in mind some proof system designed for the formal manipulation of assertions about the partial correctness of program texts with respect to a fixed interpretation A for the programming language. Stated simply, and informally, our aim in this paper is to exhibit some familiar algebraic structures A over which *any* sound Hoare-like logic for the partial correctness of **while**-program computations in A will possess some unfamiliar structural properties. From this exercise follows somewhat stronger incompleteness results than those first reported in Cook [10] and in Wand [25] for Hoare's original system about **while**-programs. And, as we shall make clear in a moment, these results in turn address some sharply defined issues in the theoretical literature to do with the complexity of the programming language in the design of a Hoare logic.

Our point of departure is Hoare's proof system as it is formally constituted for **while**-programs in [10]. We take it for granted that the reader is familiar with the papers Hoare [13], Cook [10] and Wand [25]: with these prerequisites, or the invaluable survey paper Apt [1], we can discuss our examples in more technical terms.

Let A be any relational structure and let $\mathscr{WP}$ be the class of all **while**-programs destined to compute functions on A. On choosing the first-order logical language L as assertion language, and applying a definition of the semantics $\mathscr{S}$ of $\mathscr{WP}$ to interpretation A, one may identify the study of partial correctness for $\mathscr{WP}$ computations over A as the study of a set PC(A), the *partial correctness theory* for $\mathscr{WP}$ on A.

PC(A) is defined to be the set

$$\{\{p\}S\{q\}: p, q \in L, S \in \mathcal{WP} \;\&\; A \models \{p\}S\{q\}\}$$

wherein $A \models \{p\}S\{q\}$ means *whenever p is true of an initial state for S then either S terminates in a state for which q is true or S diverges.*

With the same level of generality, one can define the *standard Hoare logic* $\mathrm{HL}_0(A)$ for $\mathcal{WP}$ on A as the set of all triples $\{p\}S\{q\}$ generated by Hoare's proof rules for $\mathcal{WP}$ and including the usually undecidable first-order theory Th(A) of A as axioms. For any sensible program semantics $\mathcal{S}$, $\mathrm{HL}_0(A)$ is *sound* in the sense that $\mathrm{HL}_0(A) \subset \mathrm{PC}(A)$. In [10], Cook showed that if L is *expressive* for $\mathcal{WP}$ over A then $\mathrm{HL}_0(A)$ is *complete* in the sense that $\mathrm{HL}_0(A) = \mathrm{PC}(A)$. Of more interest to us, on this occasion, is another theorem of [10]: if A is Presburger arithmetic then $\mathrm{HL}_0(A)$ is *not* complete; and also Wand's more detailed analysis of incompleteness [25] wherein he gave a simple, although artificial, structure A and an asserted program $\{p\}S\{q\}$ such that $\{p\}S\{q\} \in \mathrm{PC}(A)$ but $\{p\}S\{q\} \notin \mathrm{HL}_0(A)$. After settling on a weak criterion for a set of asserted programs to qualify as a Hoare logic, we will build up some general theory from which one can read off more extreme examples of incompleteness:

Theorem. *Let A be Presburger arithmetic, the field of real algebraic numbers, or the field of algebraic numbers. Then A is a computable algebraic structure with decidable first-order theory* Th(A) *such that*

(1) *each sound Hoare logic* $\mathrm{HL}(A) \supset \mathrm{HL}_0(A)$ *is r.e. but not recursive*;

(2) PC(A) *is co-r.e. but not recursive*; *in fact,* PC(A) *is a complete co-r.e. set.*

Therefore, A has no sound and complete Hoare logic for its **while**-*programs*; *and, in particular, A fails to possess even a sound, if incomplete, Hoare logic which is recursive.*

First, let us compare the theorem with the well understood intermediate situation of the standard model of arithmetic N. By Cook's Completeness Theorem, $\mathrm{HL}_0(N)$ is sound and complete, of course. The three components Th(N), $\mathrm{HL}_0(N)$ and PC(N) are highly non-constructive for they are not arithmetical sets, but they are of the same complexity, each having Turing degree O^{ω}, see Rogers [23]. For the A of the theorem the situation is quite the reverse: no completeness possible and, whatever Hoare logic HL(A) is chosen, Th(A), HL(A) and PC(A) are arithmetical and effective, but in three disparate ways (up to Turing equivalence). In view of the fact that for any finite structure A, $\mathrm{HL}_0(A) = \mathrm{PC}(A)$, it is presumably the case that Presburger Arithmetic is the canonical example of a structure for which no useful Hoare-like logic is available to reason about partial correctness for such a simple program language as $\mathcal{WP}$. This is certainly supported by the theorem that there is indeed a nice Hoare logic, which is sound and complete, for certain **loop**-programs over Presburger Arithmetic: see Cherniavsky and Kamin [8].

In this way one is led to reflect on the rôle of the complexity of program languages in seeking sound and complete Hoare logics. Although our examples are familiar (and simpler, at least in the case of Presburger Arithmetic), Wand's structure is by no means redundant as it makes the point that the *computational power* of a program language is not necessarily a factor in its possession of a complete Hoare logic: on Wand's structure, the **while**-programs compute rather trivial functions whereas on ours they compute all recursive functions. On the other hand, there is a particularly striking incompleteness theorem in Clarke [9] which says that for very rich program languages there can be no Hoare-like logic for the partial correctness of their computations on *finite structures*. Of course, for **while**-programs, augmented by many programming constructs, explicit Hoare logics which are complete for finite structures are known, see [9] and the survey paper [1].

Our principal motive for preparing this paper was to complete a technical picture of the incompleteness properties which trouble Hoare's logic, a picture largely outlined in the articles already cited. After a brief resumé of background material, we give precise definitions for the concepts we use and develop their basic properties. Several of these primary definitions are natural generalisations of ideas implicitly used in papers such as Clarke [9] and Lipton [17]. In Section 3 we establish a general sufficient condition for the phenomena just described (Theorem 3.2) and this, too, echoes particular observations repeatedly made by earlier writers on the subject; we include a complete list of relevant references at the end of the section. In Section 4 we work out the applications of the machinery announced above.

This paper is a companion to our [7] which reconsiders the relationship between the expressiveness of the assertion language L for $\mathcal{WP}$ over a structure A and the completeness of Hoare's logic $\mathrm{HL}_0(A)$. Although, strictly speaking, expressiveness has no technical rôle in the present article, some discussion of the property is necessary in order to appreciate the extent to which Hoare's logic is complete and we include notes on this subject in Section 5. Both the present paper and [7] are sequels to [5], written with J. Tiuryn, which deals with technical issues in a theoretical analysis of the thesis that a programming language semantics can be uniquely defined by a system of proof rules for its constructs; knowledge of [5] is not required, however.

We would like to thank K.R. Apt for his criticisms of an earlier edition of this paper.

1. Preliminaries on structures, assertions and programs

In this preparatory section we shall map out the technical prerequisites for the paper. In addition to the three important sources Hoare [13], Cook [10] and Wand [25], the reader would do well to consult the survey article [1].

By an *algebraic system*, *algebraic structure* or, simply, an *algebra* we shall mean a relational structure A of recursively enumerable signature Σ with constants c_i, operations σ_j and relations R_k.

The first-order language L of some signature Σ is based upon sets of variables $x_1, x_2, \ldots$ for *algebraic values* and $\beta_1, \beta_2, \ldots$ for *boolean values*. The algebraic constant, function and relational symbols of L are exactly those of Σ; its boolean constant symbols are **true**, **false** and its boolean operation symbols are $\wedge$, $\neg$. In addition, we assume L has equality symbols for its algebraic and boolean sorts as well as the usual logical connectives and quantifiers. The set of all algebraic terms of L we denote $T(\Sigma)$.

Using the syntax of L, the class $\mathcal{WP}$ of all **while**-programs (with boolean variables) over Σ is defined in the customary way.

Now for any algebra A of signature Σ, the semantics of the first-order language L over Σ determined by A has its standard definition in model theory and this we assume to be understood. The set of all assertions of L which are true in A is called the *first-order theory* of A and is denoted Th(A). For the semantics $\mathcal{S}$ of $\mathcal{WP}$ over Σ determined by A we leave the reader free to choose any sensible account of **while**-program computations: [10]; the graph-theoretic semantics in Greibach [12]; the denotational semantics described in de Bakker [4]. What constraints must be placed on this choice are the necessities of formulating and proving certain lemmas, such as Lemmas 1.1 and 1.2 below, and of verifying soundness for the standard Hoare Logic (Theorem 2.1). These conditions will be evident from the text and, for such a simple programming formula as $\mathcal{WP}$, can hardly be problematical. For definiteness, we have in mind a naïve operational semantics based upon appropriate A-register machines which yield straightforward definitions of a *state* in a $\mathcal{WP}$ computation and of the *length* of a $\mathcal{WP}$ computation [24]; and a straightforward proof of this first fact:

1.1. Lemma. *Let $S \in \mathcal{WP}$ involve variables $x = (x_1, \ldots, x_n)$. Then for each $l \in \omega$ there is a formula* $\mathrm{COMP}_{S,l}(x, y)$ *of L, wherein $y = (y_1, \ldots, y_n)$ are new variables, such that for any A and any $a, b \in A^n$, $A \models \mathrm{COMP}_{S,l}(a, b)$ if, and only if, the computation $S(a)$ terminates in l or less steps leaving the variables with values $b = (b_1, \ldots, b_n)$.*

The reader is also responsible for verifying for his or her semantics the following Normal Form Theorem for $\mathcal{WP}$ taken from Mirkowska [21].

1.2. Lemma. *There is an effective procedure which given any* **while**-*program S over signature Σ constructs a new* **while**-*program S_M over Σ of the form*

$$S_m \equiv S_1;\ \textbf{while}\ b\ \textbf{do}\ S_2\ \textbf{od},$$

where S_1 and S_2 are straight-line programs over Σ containing the variables of S, such that for any Σ-algebra A and any input state $a \in A^n$ either both $S(a)$ and $S_M(a)$ terminate with the values of their common variables identical, or both $S(a)$ and $S_M(a)$ diverge.

Putting together the semantics of L and $\mathcal{WP}$ determined by interpretation A we obtain the *partial correctness theory* PC(A) defined just as in the Introduction.

Our definition of a computable algebraic structure derives from Rabin [22] and Mal'cev [19], independent papers devoted to founding a general theory of computable algebras and their computable morphisms.

Let A be a structure of finite signature. Then A is *computable* if there exists a recursive subset Ω of the set of natural numbers ω and a surjection $\alpha:\Omega\to A$ such that

(1) the relation $\equiv_\alpha$ defined on Ω by $n\equiv_\alpha m \Leftrightarrow \alpha n=\alpha m$ in A is recursive; and

(2) for each k-ary operation σ and each k-ary relation R of A there exist recursive functions $\hat{\sigma}$ and $\hat{R}$ which commute the following diagrams:

$$\begin{tikzcd} A^k \arrow[r, "\sigma"] & A \\ \Omega^k \arrow[u, "\alpha^k"] \arrow[r, "\hat{\sigma}"] & \Omega \arrow[u, "\alpha"'] \end{tikzcd} \qquad \begin{tikzcd} A^k \arrow[r, "R"] & \{0,1\} \\ \Omega^k \arrow[u, "\alpha^k"] \arrow[ur, "\hat{R}"'] & \end{tikzcd}$$

wherein $\alpha^k(x_1,\ldots,x_k)=(\alpha x_1,\ldots,\alpha x_k)$ and R is identified with its characteristic function.

We shall use a number of concepts and results from the theory of the recursive functions: *Turing* and *many-one reducibilities*; *completeness*; *recursively inseparable sets*; *the arithmetic hierarchy*. With the exception of relativised Turing computability, particularly clear accounts of these subjects can be found in Mal'cev [20] which we shall cite as we go along. The basic reference for recursion theory remains Rogers [23] however, and this should be consulted for any idea or fact not explained or referenced here.

2. Hoare logics

Let A be a structure. The *standard Hoare logic* for $\mathcal{WP}$ over A with assertion language L has the usual axioms and proof rules for manipulating asserted **while**-programs and these can be found in [1, 4, 10, 13]. Worthy of an explicit citation is the rule of inference known as the *Consequence Rule*,

$$\frac{p\to p_1,\ \{p_1\}S\{q_1\},\ q_1\to q}{\{p\}S\{q\}}$$

and, in connection with it, the *oracle* of axioms: Each member of Th(A) is an axiom.

The set of all triples of the form $\{p\}S\{q\}$, or *asserted programs*, derivable from these axioms by the proof rules we denote $\mathrm{HL}_0(A)$; we write $\mathrm{HL}_0(A)\vdash\{p\}S\{q\}$ in place of $\{p\}S\{q\}\in\mathrm{HL}_0(A)$.

2.1. Theorem. *For any algebraic structure A, $\mathrm{HL}_0(A)$ is sound in the sense that $\mathrm{HL}_0(A)\subset\mathrm{PC}(A)$ and is recursively enumerable in* Th(A).

The first statement is contained in Section 5 of [10]. The second statement is implicit in Section 6 of [10] and is obvious anyway; this latter property we take as our definition of a Hoare logic:

A *Hoare logic for* $\mathcal{WP}$ *over* A *with assertion language* L is any subset HL(A) of $L \times \mathcal{WP} \times L$ which is recursively enumerable relative to Th(A).

A Hoare logic HL(A) is *sound* if, and only if, HL(A) $\subset$ PC(A) and it is (*relatively*) *complete* if HL(A) = PC(A).

These definitions are implicit in [17] and [9].

2.2. Proposition. *Let A be any algebraic structure and* HL(A) *a Hoare logic for* $\mathcal{WP}$ *on A. Then*

(1) HL(A) *is r.e. in* Th(A) *and*

(2) PC(A) *is co-r.e. in* Th(A).

Proof. Of course statement (1) follows by definition. Consider (2). Let $p, q \in L$ and $S \in \mathcal{WP}$. For each $k \in \omega$, let $Q_k(p, S, q)$ be this sentence in L, derived from Lemma 1.1:

$$\forall x[p(x) \rightarrow \{\exists y(\mathrm{COMP}_{S,k}(x, y) \wedge q(y)) \vee \neg\exists y \,.\, \mathrm{COMP}_{S,k}(x, y)\}].$$

Now observe that

$$\begin{aligned}\{p\}S\{q\} \in \mathrm{PC}(A) &\Leftrightarrow A \models \{p\}S\{q\}\\ &\Leftrightarrow A \models Q_k(p, S, q) \quad \text{for each } k\\ &\Leftrightarrow \forall k \,.\, [Q_k(p, S, q) \in \mathrm{Th}(A)].\end{aligned}$$

Thus PC(A) is co-r.e. in Th(A).

2.3. Theorem. *There exists a sound and relative complete Hoare-like logic* HL(A) *for* $\mathcal{WP}$ *on A if, and only if,* PC(A) is recursive in Th(A).

Proof. Trivially, if PC(A) is recursive in Th(A) then it qualifies as a Hoare logic which is sound and relatively complete. On the other hand, if HL(A) is some sound and relatively complete Hoare logic then PC(A) = HL(A) and, by Proposition 2.2, HL(A) is both r.e. and co-r.e. in Th(A).

A basic reference point for the next section is this particular case of Theorem 2.3.

2.4. Corollary. *Let A be an algebra with decidable first-order theory. Then A has a sound and relatively complete Hoare logic for* $\mathcal{WP}$ *over A if, and only if, its partial correctness theory is decidable.*

3. The halting problem and decidable theories

Let $\{P_e\colon e\in\omega\}$ be a recursive enumeration of $\mathcal{WP}$ over the signature of algebra A. In the case $A=N$, the standard model of arithmetic, the halting problem for $\mathcal{WP}$ over N can be defined as

$$K=\{(e,n)\colon P_e(n)\downarrow\}\subset\omega\times\omega.$$

And it is well known that K is an r.e., non-recursive set (because **while**-programs compute the recursive functions on N). Indeed, K is a complete r.e. set, meaning: *every r.e. subset of ω is many-one reducible to K.* (Remember that $X\subset\omega$ is *many-one reducible* to $Y\subset\omega$ if there exists a recursive function $f\colon\omega\to\omega$ such that $n\in X\Leftrightarrow f(n)\in Y$; in symbols $X\leqslant_m Y$.) We want to define a number-theoretic halting problem for $\mathcal{WP}$ on any A and we shall do this by syntactically modelling the natural algebraic halting problem $\{(e,a)\colon P_e(a)\downarrow\}\in\omega\times A$ restricted to the minimal Σ-subalgebra $\text{MIN}_\Sigma(A)$ of A. The algebra $\text{MIN}_\Sigma(A)$ is, by definition, the Σ-subalgebra of A generated from the constants of A by its operations. Its connection with syntax is that it is the image of the valuation map $v\colon T(\Sigma)\to A$ which is defined by assigning to each operation symbol and constant symbol in t the function and element they name in A and then evaluating. Thus $T(\Sigma)$ is a recursive set of names for the elements of $\text{MIN}_\Sigma(A)$.

By a *state formula* we mean a formula in L of the form $\bigwedge_{i=1}^n x_i=t_i$ where x_i is a variable of L and $t_i\in T(\Sigma)$ is a term of L, $1\leqslant i\leqslant n$.

Let $\{\phi_i\colon i\in\omega\}$ be a recursive enumeration of all state formulae. Then by the *halting problem for $\mathcal{WP}$ on A* we shall here mean the set $K(A)\subset\omega\times\omega$ defined by

$$K(A)=\{(e,i)\colon P_e \text{ and } \phi_i \text{ have the same variables, say}$$
$$x=(x_1,\ldots,x_n),\text{ and } A\models\phi_i(x)\to P_e(x)\downarrow\},$$

clearly, $K(N)$ is (recursively isomorphic to) K.

3.1. Lemma. *The set* $\neg K(A)$ *is many-one reducible to* PC(A). *In particular, if* $K(N)\leqslant_m K(A)$ *then* PC(A) *is not recursive.*

Proof. This is immediate because $(e,i)\notin K(A)$ if, and only if, either the variables of P_e and ϕ_i fail to match or $\{\phi_i\}P_e\{\textbf{false}\}\in\text{PC}(A)$.

We generate our examples from this technical fact.

3.2. Theorem. *Suppose* Th(A) *to be decidable and that* $K(N)$ *is many-one reducible to* $K(A)$. *Let* HL(A) *be any sound Hoare logic for $\mathcal{WP}$ on A extending the standard Hoare logic* $\text{HL}_0(A)$; *that is* $\text{HL}_0(A)\subset\text{HL}(A)\subset\text{PC}(A)$. *Then*

(1) HL(A) *is r.e. but not recursive*;

(2) PC(A) *is co-r.e. but not recursive*; *indeed,* PC(A) *is a complete co-r.e. set.*

In particular, A has no sound and complete Hoare logic for its **while**-*programs.*

Proof. The absence of completeness for Hoare logics is an application of Corollary 2.4 to statement (2). Statement (2) is an immediate consequence of Proposition 2.2 and Lemma 3.1. Thus the concern for completeness can be settled quite easily. More difficult is the proof that A has no sound, but incomplete, *recursive* Hoare logic. Consider statement (1).

Let U and V be two disjoint r.e. subsets of ω which are recursively inseparable. This means there does not exist a recursive set R such that $U \subset R$ and $V \subset \neg R$ (to see why such sets exist consult [20, p. 210]).

Since $K(N) \leqslant_m K(A)$, and $K(N)$ is many–one complete for all r.e. sets, we can choose recursive functions $u, v, f, g : \omega \to \omega$ such that

$$n \in U \Leftrightarrow A \models \phi_{u(n)}(x) \to P_{f(n)}(x)\downarrow,$$

$$n \in V \Leftrightarrow A \models \phi_{v(n)}(y) \to P_{g(n)}(y)\downarrow$$

wherein $x = (x_1, \ldots, x_r)$, $y = (y_1, \ldots, y_s)$ and these depend on n.

Without loss of generality we can assume these expressions between formulae and programs to have the following normal forms:

(i) both $P_{f(n)}$ and $P_{g(n)}$ have the form

$P = S$; **while** b **do** S' **od**

where S and S' are loop-free programs;

(ii) both $P_{f(n)}$ and $P_{g(n)}$ have disjoint sets of variables;

(iii) the formulae $\phi_{u(n)}$ and $\phi_{v(n)}$ are A-equivalent: $A \models \phi_{u(n)} \leftrightarrow \phi_{v(n)}$.

Each condition can be met by applying recursive transformations of programs and formulae. Step (i) is provided for by Lemma 1.2 and steps (ii) and (iii) are trivial to arrange effectively. Thus we assume these transformations have been effected and, retaining the notation u, v, f, g for the normalised reduction maps, take

$P_{f(n)} \equiv S_{f(n)}$; **while** $b_{f(n)}$ **do** $S'_{f(n)}$ **od**

$P_{g(n)} \equiv S_{g(n)}$; **while** $b_{g(n)}$ **do** $S'_{g(n)}$ **od**.

By piecing these programs together we define a recursive function $d : \omega \to \omega$. Let $P_{d(n)}$ be the following program wherein $TURN$ is a boolean variable:

```
$S_{f(n)}$; $S_{g(n)}$; $TURN$ := true;
while $b_{f(n)} \wedge b_{g(n)}$ do if $TURN$ then $S'_{f(n)}$ else $S'_{g(n)}$ fi;
                               $TURN := \neg TURN$;
                        od;
```

It is easy to check that

for all $n \notin U$, $A \models \{\phi_{u(n)}\}P_{d(n)}\{TURN = \mathbf{true}\}$,

for all $n \notin V$, $A \models \{\phi_{v(n)}\}P_{d(n)}\{TURN = \mathbf{false}\}$.

And, moreover, since $\phi_{u(n)}$ and $\phi_{v(n)}$ are A-equivalent, that

$$A \models \phi_{u(n)} \to P_{d(n)}\downarrow \text{ if, and only if, } n \in U \cup V.$$

3.3. Lemma.

$$n \in U \textit{ implies } \mathrm{HL}_0(A) \vdash \{\phi_{u(n)}\}P_{d(n)}\{TURN = \mathbf{false}\},$$
$$n \in V \textit{ implies } \mathrm{HL}_0(A) \vdash \{\phi_{v(n)}\}P_{d(n)}\{TURN = \mathbf{true}\}.$$

On proving the lemma we can involve any $\mathrm{HL}_0(A) \subset \mathrm{HL}(A) \subset \mathrm{PC}(A)$ in a separation of U, V. Thus, for any such Hoare Logic $\mathrm{HL}(A)$ define $\lambda : \omega \to \omega$ by

$$\lambda(n) = \begin{cases} 0, & \text{if } \mathrm{HL}(A) \vdash \{\phi_{u(n)}\}P_{d(n)}\{TURN = \mathbf{false}\}, \\ 1, & \text{otherwise.} \end{cases}$$

Clearly λ is recursive in $\mathrm{HL}(A)$ and, by the above constructions and Lemma 3.3, λ separates U, V since $n \in U \Rightarrow \lambda(n) = 0$ and $n \in V \Rightarrow \lambda(n) = 1$. If $\mathrm{HL}(A)$ were recursive then this would contradict the inseparability of U and V. Thus $\mathrm{HL}(A)$ is r.e. but not recursive.

Lemma 3.3 is obtained from this general fact.

3.4. Completeness Theorem for terminating closed programs. *Let A be any algebra and let $\mathrm{HL}_0(A)$ be the standard Hoare logic for $\mathcal{WP}$ over A with assertion language L. Let ϕ, ψ be state formulae and let S be a* **while**-*program having the same variables $x = (x_1, \ldots, x_n)$. If*

$$A \models \phi(x) \to S(x)\downarrow \quad \textit{and} \quad A \models \{\phi\}S\{\psi\}$$

then $\mathrm{HL}_0(A) \vdash \{\phi\}S\{\psi\}$.

Proof. This is done by induction on the complexity of S. The basis and most cases of the induction step are easy and are omitted. We consider only the case

$$S = \mathbf{while}\ b\ \mathbf{do}\ S_0\ \mathbf{od}.$$

So suppose for such S that $A \models \phi(x) \to S(x)\downarrow$ and $A \models \{\phi\}S\{\psi\}$; and assume Lemma 3.4 is true of S_0.

Let the computation which ϕ determines from S on $\mathrm{MIN}_\Sigma(A)$ involve l executions of S_0. And let $\phi^0, \ldots, \phi^l$ be state formulae defining the initial states at each of these executions together with the final state. Thus, these formulae are defined inductively by $\phi^0 = \phi$ and $\phi^i =$ that formula, unique up to A-equivalence, such that $A \models \{\phi^i\}S_0\{\phi^{i+1}\}$.

Setting $\theta = \bigvee_{i=0}^{l} \phi^i$ we see clearly from its construction that

$$A \models \phi(x) \to \theta(x) \quad \text{and} \quad A \models \theta(x) \wedge \neg b(x) \to \psi(x)$$

and that we have now to prove $\mathrm{HL}_0(A) \vdash \{\theta \wedge b\}S_0\{\theta\}$.

But $A \models \theta(x) \wedge b(x) \leftrightarrow \bigvee_{i=1}^{l-1} \phi^i(x)$ and $A \models \bigvee_{i=1}^{l} \phi^i(x) \rightarrow \theta(x)$. Therefore, it is sufficient to show

$$\mathrm{HL}_0(A) \vdash \left\{ \bigvee_{i=0}^{l-1} \phi^i \right\} S_0 \left\{ \bigvee_{i=0}^{l-1} \phi^{i+1} \right\}.$$

The induction hypothesis says that for each $0 \leq i < l$

$$\mathrm{HL}_0(A) \vdash \{\phi^i\} S_0 \{\phi^{i+1}\}$$

and to string these proofs together it is enough to apply the following derived proof rule of $\mathrm{HL}_0(A)$: for any $p_1, p_2, q_1, q_2 \in L$ and any $S \in \mathcal{WP}$

$$\frac{\{p_1\}S\{q_1\}, \{p_2\}S\{q_2\}}{\{p_1 \vee p_2\}S\{q_1 \vee q_2\}}.$$

To verify this is indeed a derived rule of $\mathrm{HL}_0(A)$ is an easy induction on proof lengths.

3.5. Notes on the literature. Theorem 3.2 gives an explicit and definitive form to observations made by several authors about the connection between the termination of programs and the completeness of Hoare's logic. All these observations are subsumed by the following remark in Section 2.7 of [1] which is a starting point for the formulation of Theorem 3.2:

If the halting problem for programs in $\mathcal{WP}$ is undecidable on an interpretation A then $\mathrm{PC}(A)$ *is not r.e. because* $A \models$ **{true}**S**{false}** *if, and only if,* $S \in \mathcal{WP}$ *fails to halt on all initial values of its variables. Therefore, if* $\mathrm{HL}_0(A)$ *is r.e. then* $\mathrm{HL}_0(A) \neq \mathrm{PC}(A)$.

Once one has framed the concepts of the *formal halting problem* $K(A)$ and of a *Hoare-like logic*, one can obtain the last statement of Theorem 3.2 from this general remark, but not the theorem's main statements (1) and (2), of course. The origin of Apt's observation is the argument of Theorem 2 in [10] which shows that $\mathrm{HL}_0(A)$ is incomplete in case A is Presburger Arithmetic; and we again find it in use in [9], there applied to finite structures A which interpret much richer programming languages than $\mathcal{WP}$ with undecidable halting problems for $|A| \geq 2$. Actually, the termination of programs over finite interpretations was considered, independently of Hoare's logic, in Langmaack [14, 15]; for example the observation applies there to sharpen one of Clarke's incompleteness results to include structures A with $|A| = 1$, and it appears often in Langmaack and Olderog's extensive studies of Hoare-like logics for the higher program languages, see [16] and the references there cited.

However, Theorem 3.2 should not be confused with any interesting fact about the incompleteness of Hoare-like logics addressing the issue of *total* correctness. As it happens $\mathrm{HL}_0(A)$ is not even sound for asserted programs with their total correctness semantics; and indeed L is unable to support any logic for total correctness as may easily be surmised from remarks in Section 2.6 of [1].

4. Examples

The basic reference for information about decidable first-order theories is Ershov *et al.* [11]. Here we choose to mention a few structures with decidable theories which lead to easily appreciated examples for incompleteness:

(1) Presburger's Arithmetic having domain ω, constant $0 \in \omega$ and operation the successor function on ω;

(2) any algebraically closed field such as the complex numbers or algebraic numbers;

(3) any real closed field such as the real numbers or real algebraic numbers.

In each case it is easy to verify the halting problem hypothesis in Theorem 3.2 providing, of course, one chooses fields of characteristic zero: the standard halting problem K is recursively isomorphic to $K(A)$ when A is Presburger Arithmetic and since this structure can be embedded in the field of rational numbers Q, which is the prime subfield of any field F of characteristic zero, the reduction $K(N) \leqslant_m K(Q) \leqslant_m K(F)$ follows.

For a finer comparison with the standard situation $A = N$ we prefer to choose computable structures (and also we have in mind the rôle of computable interpretations in [17]). Presburger Arithmetic is clearly computable. To obtain computable fields of kinds (2) and (3) one applies the following theorems from Rabin [22] and Madison [18] respectively. Let F be a computable field. Then the algebraic closure of F is computable. If, in addition, F has a computable ordering then the real closure of F is computable.

5. Concluding remarks on expressiveness and completeness

Cook's discussion of the completeness of Hoare's logic begins with the fact that $\mathrm{HL}_0(A)$ is incomplete when A is Presburger Arithmetic and continues with the observation that a particular source of difficulty in attempting to assess the completeness of Hoare's system is that the assertion language on which it is based may not be able to define all the invariants for loops. Thus, Cook defines a not necessarily first-order assertion language AL to be expressive for $\mathcal{WP}$ over structure A if for each assertion $\phi \in \mathrm{AL}$ and program $S \in \mathcal{WP}$ the strongest postcondition $\mathrm{sp}_A(\phi, S)$ is definable in AL. In our situation AL is fixed as the first-order language L so expressiveness is a property of interpretations; for example, the standard model of arithmetic is expressive, but Presburger Arithmetic is not. Cook showed that *if* AL *is expressive for* $\mathcal{WP}$ *over* A *then* $\mathrm{HL}_0(A)$ *is complete* and one now says that Hoare's logic is *complete in the sense of Cook* because whenever the assertion language is not troubled by its own internal inadequacies then the system is indeed complete. Does this perceptive theoretical analysis resolve the incompleteness phenomena noticed for Presburger Arithmetic and extensively analysed in this paper? We believe it does not.

The problems begin with the paucity of structures which turn out to be expressive and with the existence of *natural* structures for which nothing resembling a complete Hoare logic can be made even for **while**-programs. A theorem about the expressive structures by de Millo, Lipton and Snyder, announced in [17], emphasises the first point:

If L is expressive for $\mathcal{WP}$ over A then either (1) *a standard model of arithmetic can be defined in A or* (2) *for each $S \in \mathcal{WP}$ there is $n = n(S) \in \omega$ such that S reaches at most n states in any computation over A.*

Thus such structures either contain arithmetic in a *first-order definable way* and so are subject to the logical pathologies characteristic of arithmetic or else they are essentially uniformly locally finite structures on which one can compute very little (see [24]).

Secondly, come problems with the rôle of expressiveness in completeness arguments. Finite structures are expressive and so Clarke's work shows that when the programming languages grow more complicated then expressiveness can no longer guarantee the existence of a complete Hoare logic. On the other hand, expressiveness is not even a necessary condition for completeness: in [7] we point out structures A where $\mathrm{HL}_0(A)$ is complete, but L is not expressive for $\mathcal{WP}$ over A. (However we have not yet found computable structures for which this is the case.)

Thirdly, it should be remembered that the quest for completeness for Hoare-like logics $\mathrm{HL}(A)$, using the first-order theory of the interpretation A as oracle, ignores the original idea that the systems should be based upon an axiomatic specification of the data types on which the programs compute (see [13]). It seems to us that this connection between data type specifications and formalised correctness concerns is one which is worth recovering even at the expense of the fine completeness theorems Cook was able to provide; an experiment of this kind, relying solely on algebraic specifications for data types, is [6] and its completeness theorem should be compared with Theorem 3.2 and Theorem 4.1 in [2]. A bold attempt to think through the depressing theoretical problems which beset total correctness has been made by Back in [3].

References

[1] K.R. Apt, Ten years of Hoare's logic, a survey, in: F.V. Jensen, B.H. Mayoh and K.K. Møller, Eds., *Proc. 5th Scandinavian Logic Symposium* (Aalborg University Press, Aalborg, 1979) 1–44. A second edition of this paper will appear in ACM Transactions on Programming Languages and Systems.

[2] K.R. Apt, J.A. Bergstra and L.G.L.T. Meertens, Recursive assertions are not enough—or are they?, *Theoret. Comput. Sci.* **8** (1979) 73–87.

[3] R.J.R. Back, Proving total correctness of nondeterministic programs in infinitary logic, *Acta Informat.*, to appear.

[4] J.W. de Bakker, *Mathematical Theory of Program Correctness* (Prentice-Hall, London, 1980).

[5] J.A. Bergstra, J. Tiuryn and J.V. Tucker, Floyd's principle, correctness theories and program equivalence, *Theoret. Comput. Sci.*, to appear.

[6] J.A. Bergstra and J.V. Tucker, Algebraically specified programming systems and Hoare's logic, Mathematical Centre, Department of Computer Science Research Report IW 143, Amsterdam (1980).

[7] J.A. Bergstra and J.V. Tucker, Expressiveness and the completeness of Hoare's logic, Mathematical Centre, Department of Computer Science Research Report IW 149, Amsterdam (1980).

[8] J. Cherniavsky and S. Kamin, A complete and consistent Hoare axiomatics for a simple programming language, *J. ACM* **26** (1979) 119–128.

[9] E.M. Clarke, Programming language constructs for which it is impossible to obtain good Hoare-like axioms, *J. ACM* **26** (1979) 129–147.

[10] S.A. Cook, Soundness and completeness of an axiom system for program verification, *SIAM J. Comput.* **7** (1978) 70–90.

[11] Y.L. Ershov, I.A. Lavrov, A.D. Taimanov and M.A. Taitslin, Elementary theories, *Russian Math. Surveys* **20** (4) (1965) 35–105.

[12] S.A. Greibach, *Theory of Program Structures: Schemes, Semantics, Verification* (Springer, Berlin, 1975).

[13] C.A.R. Hoare, An axiomatic basis for computer programming, *Comm. ACM* **12** (1969) 576–580.

[14] H. Langmaack, On procedures as open subroutines I, *Acta Informat.* **2** (1973) 311–333.

[15] H. Langmaack, On procedures as open subroutines II, *Acta Informat.* **3** (1974) 227–241.

[16] H. Langmaack and E.R. Olderog, Present day Hoare-like systems for programming languages with procedures: power, limits and most likely extensions, in: J.W. de Bakker and J. van Leeuwen, *Automata, Languages and Programming* (Springer, Berlin, 1980) 363–373.

[17] R.J. Lipton, A necessary and sufficient condition for the existence of Hoare logics, *18th IEEE Symposium on Foundations of Computer Science*, Providence, RI (1977) 1–6.

[18] E.W. Madison, A note on computable real fields, *J. Symbolic Logic* **35** (1970) 239–241.

[19] A.I. Mal'cev, Constructive algebras, I., *Russian Math. Surveys* **16** (1961) 77–129.

[20] A.I. Mal'cev, *Algorithms and Recursive Functions* (Wolters-Noordhoff, Groningen, 1970).

[21] G. Mirkowska, Algorithmic logic and its applications in the theory of programs II, *Fund. Inform.* **1** (1977) 147–165.

[22] M.O. Rabin, Computable algebra, general theory and the theory of computable fields, *Trans. Amer. Math. Soc.* **95** (1960) 341–360.

[23] H. Rogers, *Theory of Recursive Functions and Effective Computability* (McGraw-Hill, New York, 1967).

[24] J.V. Tucker, Computing in algebraic systems, in: F.R. Drake and S.S. Wainer, Eds., *Recursion Theory, its Generalisations and Applications* (Cambridge University Press, Cambridge, 1980).

[25] M. Wand, A new incompleteness result for Hoare's system, *J. ACM* **25** (1978) 168–175.