stichting
mathematisch
centrum

AFDELING INFORMATICA IW 136/80 APRIL
(DEPARTMENT OF COMPUTER SCIENCE)

J.A. BERGSTRA & J.V. TUCKER

THE FIELD OF ALGEBRAIC NUMBERS FAILS TO POSSESS EVEN A NICE SOUND, IF RELATIVELY INCOMPLETE, HOARE-LIKE LOGIC FOR ITS <u>WHILE</u>-PROGRAMS

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

1980 Mathematics subject classification: 03D35, 03D75, 68B10

ACM-Computing Review-category: 5.24

The field of algebraic numbers fails to possess even a nice sound, if relatively incomplete, Hoare-like logic for its while-programs *)

by

J.A. Bergstra**) & J.V. Tucker

ABSTRACT

Under a weak definition of a Hoare logic for while-programs, interpreted in a structure A, we show that many familiar structures fail to admit even a nice sound, if relatively incomplete, Hoare logic for the partial correctness of their while-program computations. Among our examples are Presburger Arithmetic, the field of real algebraic numbers, and the field of algebraic numbers.

KEY WORDS & PHRASES: *program correctness; Hoare-like logics; decidable theories; halting problem; Presburger Arithmetic; real and algebraically closed fields*

---

*) This report will be submitted for publication elsewhere.

**) Department of Computer Science, University of Leiden, Wassenaarseweg 80, Postbus 9512, 2300 RA LEIDEN, The Netherlands.

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 WAND [19] 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 COOK [7]. We take it for granted that the reader is familiar with the papers HOARE [10], COOK [7] and WAND [19]: with these prerequisites or the invaluable survey paper APT [1], we can discuss our examples in more technical terms.

Let A be any algebraic system and let $\mathcal{WP}$ 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 $\mathcal{S}$ of $\mathcal{WP}$ to interpretation A, one may identify the study of partial correctness for $\mathcal{WP}$ computations over A as the study of a set PC(A), the *partial correctness theory* for $\mathcal{WP}$ 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* $HL_0(A)$ 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 first-order theory Th(A) of A as axioms. For any sensible program semantics $\mathcal{S}$, $HL_0(A)$ is *sound* in the sense that $HL_0(A) \subset PC(A)$. In [19], Wand constructed a simple, but artificial structure A for which Hoare's logic may not be complete in the sense that $HL_0(A) = PC(A)$. After settling on a weak criterion for a set of asserted programs to qualify as a Hoare logic we will build up a little general theory from which one can read off this fact.

<u>THEOREM</u>. *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* $HL(A) \supset HL_0(A)$ *is r.e. but not recursive.*

(2) PC(A) *is co.r.e. but not recursive; in fact,* PC(A) *is a complete* Π^0_1 *set.*

In particular, A *has no sound and complete Hoare logic for its <u>while</u>-programs. Indeed,* 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. $HL_0(N)$ is sound and complete, of course. The three components Th(N), $HL_0(N)$ and PC(N) are highly non-constructive for they are not arithmetical sets, but they are of the same complexity, each having Turing degree 0^{ω}. 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 effective but in three disparate ways (up to Turing equivalence).

In view of the fact that for any finite structure A, $HL_0(A) = PC(A)$, 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 <u>loop</u>-programs over Presburger Arithmetic: see CHERNIAVSKY & KAMIN [5].

In this way one is lead 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. On the other hand, there is a particularly striking incompleteness theorem in CLARKE [6] which says that for very complicated program languages there can be no Hoare-like logic for the partial correctness of their computations on *finite* algebras. Of course, for while-programs, augmented by many programming constructs, explicit Hoare logics which are complete for finite structures are known, see CLARKE [6] and the survey paper APT [1].

After a brief resumé of background material, we give precise definitions for the concepts we use and develop their basic properties. In section 3 we establish a general sufficient condition for the phenomena just described while section 4 works out some of its applications.

Finally we would like to cite an ancillary motive for considering Hoare logics and their incompleteness properties. This paper is a companion to our [3], written with J. Tiuryn, which deals with technical issues in a theoretical analysis of the thesis that a program language semantics can be uniquely defined by a system of proof rules for its constructs. Since knowledge of [3] is not required here we leave it to the interested reader to consult that paper for further information on this related subject.

1. PRELIMINARIES ON ALGEBRAS AND PROGRAMS

In this preparatory section we shall map out the technical prerequisites for the paper. In addition to the three important sources HOARE [10], COOK [7] and WAND [19], the reader would do well to consult the survey article APT [1].

By an *algebraic system, algebraic structure* or, simply, an *algebra* we shall mean a relational structure $A = (A; c_i, \sigma_j, R_k)$ 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 sentences of L which are true in A is called the *first-order theory* of A and is denoted Th(A); see CHANG & KEISLER [4]. 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: COOK [7]; the graph-theoretic semantics in GREIBACH [9]; the sophisticated denotational semantics described in DE BAKKER [2]. 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 naîve 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 [18]; 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* $\ell \in \omega$ *there is a formula* $\mathrm{COMP}_{S,\ell}(x,y)$ *of* L, *wherein* $y = (y_1,\ldots,y_n)$ *are new variables, such that for any* A *and any* $a,b \in A^n$, $A \models \mathrm{COMP}_{S,\ell}(a,b)$ *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 [15].

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*

$$S_M \equiv S_1;\ \underline{\text{while}}\ b\ \underline{\text{do}}\ S_2\ \underline{\text{od}},$$

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 system derives from RABIN [16] and MAL'CEV [13], independent papers devoted to founding a general theory of computable algebras and their computable morphisms:

Let A be an algebra of finite signature. Then A is *computable* if there exists a recursive subset Ω of the set of natural numbers ω and a surjection $\alpha\colon \Omega \to A$ such that (1) the relation $\equiv_\alpha$ defined on Ω by $n \equiv_\alpha m \Longleftrightarrow \alpha n = \alpha m$ 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 [14] which we shall cite as we go along. The basic reference for recursion theory remains ROGERS [17] however, and this should be consulted for any idea or fact not explained or referenced here.

2. HOARE LOGICS

Let A be an algebra. The *standard Hoare logic* for $\mathcal{WP}$ over A with assertion language L has the following axioms and proof rules for manipulating asserted programs: let $S,S_1,S_2 \in \mathcal{WP}$; $p,q,p_1,q_1, r \in L$; $b \in L$, a quantifier-free formula.

1. Assignment axiom: for $t \in T(\Sigma)$ and x a variable of L

$$\{p[t/x]\}x := t\{p\}$$

   where $p[t/x]$ stands for the result of substituting t for free occurrences of x in p.

2. Composition rule:

$$\frac{\{p\}S_1\{r\},\{r\}S_2\{q\}}{\{p\}S_1;S_2\{q\}}$$

3. Conditional rule:

$$\frac{\{p\wedge b\}S_1\{q\},\{p\wedge\neg b\}S_2\{q\}}{\{p\}\ \underline{\text{if}}\ b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\ \underline{\text{fi}}\ \{q\}}$$

4. Iteration rule:

$$\frac{\{p\wedge b\}S\{p\}}{\{p\}\ \underline{\text{while}}\ b\ \underline{\text{do}}\ S\ \underline{\text{od}}\ \{p\wedge\neg b\}}$$

5. Consequence rule:

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

   And, in connection with 5,

6. Oracle axiom: 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 $HL_0(A)$; we write $HL_0(A) \vdash \{p\}S\{q\}$ in place of $\{p\}S\{q\} \in HL_0(A)$.

2.1. THEOREM. *For any algebraic structure* A, $HL_0(A)$ *is sound in the sense that* $HL_0(A) \subset PC(A)$ *and is recursively enumerable in* $Th(A)$.

The first statement is contained in §5 of COOK [7]. The second statement is implicit in §6 of COOK [7] 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 × $\mathcal{WP}$ × L which is recursively enumerable relative to Th(A).

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

These definitions are implicit in LIPTON [11] and CLARKE [6].

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* Σ_1^0 *in* Th(A) *and* (2) PC(A) *is* Π_1^0 *in* Th(A).

PROOF. Of course statement (1) follows by definition. Consider (2). Let p,q ∈ L and S ∈ $\mathcal{WP}$. For each k ∈ ω, 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

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

Thus PC(A) is Π_1^0 in Th(A). Q.E.D.

2.3. THEOREM. *There exists a sound and relative complete Hoare* 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). Q.E.D.

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 $\{P_e : e \in \omega\}$ be a recursive enumeration of $\mathcal{WP}$ for 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

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

And it is well known that K is an r.e., non-recursive set (because <u>while</u> programs compute the recursive functions on N). Indeed, K is a complete Σ_1^0 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 $f:\omega \to \omega$ such that $n \in X \iff f(n) \in Y$; in symbols $X \leq_m Y$.) 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 $\{(e,a) : P_e(a)\downarrow\} \in \omega \times A$ restricted to the minimal Σ-subalgebra $MIN_\Sigma(A)$ of A. The algebra $MIN_\Sigma(A)$ 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 $v: T(\Sigma) \to A$ 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 $MIN_\Sigma(A)$.

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, $1 \leq i \leq n$.

Let $\{\phi_i : i \in \omega\}$ 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

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

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* $K(N) \leq_m K(A)$ *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 $\{\phi_i\}P_e\{\underline{\text{false}}\} \in PC(A)$. Q.E.D.

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* $HL_0(A)$; *that is* $HL_0(A) \subset HL(A) \subset PC(A)$. *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* Π_1^0 *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 usual 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 MAL'CEV [14,p.210]).

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

$$n \in U \iff A \models \phi_{u(n)}(x) \to P_{f(n)}(x)\downarrow$$
$$n \in V \iff A \models \phi_{v(n)}(y) \to P_{g(n)}(y)\downarrow$$

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

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

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:

$$\begin{array}{l} S_{f(n)};\ S_{g(n)};\ \mathit{TURN} = \underline{\text{true}} \\ \underline{\text{while}}\ b_{f(n)} \wedge b_{g(n)}\ \underline{\text{do}}\ \underline{\text{if}}\ \mathit{TURN}\ \underline{\text{then}}\ S'_{f(n)}\ \underline{\text{else}}\ S'_{g(n)}\ \underline{\text{fi}}; \\ \qquad\qquad\qquad\qquad \mathit{TURN} := \neg\mathit{TURN}; \\ \qquad\qquad\qquad\quad \underline{\text{od}}; \end{array}$$

It is easy to check that

$$\text{for all } n \notin U,\ A \models \{\phi_{u(n)}\}P_{d(n)}\ \{\mathit{TURN} = \underline{\text{true}}\}$$
$$\text{for all } n \notin V,\ A \models \{\phi_{v(n)}\}P_{d(n)}\ \{\mathit{TURN} = \underline{\text{false}}\}.$$

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. <u>LEMMA</u>. $n \in U$ *implies* $HL_0(A) \vdash \{\phi_{u(n)}\}P_{d(n)}\{\mathit{TURN} = \underline{\text{false}}\}$

$n \in V$ *implies* $HL_0(A) \vdash \{\phi_{v(n)}\}P_{d(n)}\{\mathit{TURN} = \underline{\text{true}}\}$.

On proving the lemma we can involve any $HL_0(A) \subset HL(A) \subset PC(A)$ in a separation of U,V. Thus, for any such Hoare Logic HL(A) define $\lambda:\omega \to \omega$ by

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

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

Lemma 3.3 is obtained from this general fact.

3.4. Completeness for terminating closed programs lemma

Let A *be any algebra and let* $HL_0(A)$ *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 $HL_0(A) \vdash \{\phi\}S\{\psi\}$.

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

$$S \equiv \underline{\text{while}}\ b\ \underline{\text{do}}\ S_0\ \underline{\text{od}}.$$

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 $MIN_\Sigma(A)$ involve ℓ executions of S_0. And let $\phi^0,\ldots,\phi^\ell$ 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 ϕ^i = that formula, unique up to A-equivalence, such that $A \models \{\phi^i\}S_0\{\phi^{i+1}\}$.

Setting $\theta = \vee_{i=0}^{\ell} \phi^i$ 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 $HL_0(A) \vdash \{\theta \wedge b\}S_0\{\theta\}$.

But $A \models \theta(x) \wedge b(x) \leftrightarrow \vee_{i=0}^{\ell-1} \phi^i(x)$ and $A \models \vee_{i=1}^{\ell} \phi^i(x) \to \theta(x)$. Therefore, it is sufficient to show

$$HL_0(A) \vdash \{\vee_{i=0}^{\ell-1} \phi^i\}S_0\{\vee_{i=0}^{\ell-1} \phi^{i+1}\}.$$

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

$$HL_0(A) \vdash \{\phi^i\}S_0\{\phi^{i+1}\}$$

and to string these proofs together it is enough to apply the following derived proof rule of $HL_0(A)$: 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 $HL_0(A)$ is an easy induction on proof lengths. Q.E.D.

4. EXAMPLES

The basic reference for information about decidable first-order theories is ERSHOV *et al* [8]. 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. 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 LIPTON [11]). Presburger Arithmetic is clearly computable. To obtain computable fields of kinds (2) and (3) one applies the following theorems from RABIN [16] and MADISON [12] 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.

REFERENCES

[1] APT, K.R., *Ten years of Hoare's logic, a survey* in F.V. JENSEN, B.H. MAYOH & K.K. MØLLER (eds.) *Proceedings from 5th Scandinavian Logic Symposium,* Aalborg University Press, Aalborg, 1979, 1-44.

[2] DE BAKKER, J.W., *Mathematical theory of program correctness,* Prentice-Hall International, London, 1980.

[3] BERGSTRA, J.A., J. TIURYN & J.V. TUCKER, *Correctness theories and program equivalence,* Mathematical Centre, Department of Computer Science Research Report IW 119, Amsterdam, 1979. (To appear in Theoretical Computer Science.)

[4] CHANG, C.C. & H.J. KEISLER, *Model theory,* North-Holland, Amsterdam, 1973.

[5] CHERNIAVSKY, J. & S. KAMIN, *A complete and consistent Hoare axiomatics for a simple programming language,* J. Association Computing Machinery 26 (1979) 119-128.

[6] CLARKE, E.M., *Programming language constructs for which it is impossible to obtain good Hoare-like axioms,* J. Association Computing Machinery 26 (1979) 129-147.

[7] COOK, S.A., *Soundness and completeness of an axiom system for program verification,* SIAM J. Computing 7 (1978) 70-90.

[8] ERSHOV, Y.L., I.A. LAVROV, A.D. TAIMANOV & M.A. TAITSLIN, *Elementary theories,* Russian Mathematical Surveys, 20 (4) (1965) 35-105.

[9] GREIBACH, S.A., *Theory of program structures: schemes, semantics, verification*, Springer-Verlag, Berlin, 1975.

[10] HOARE, C.A.R., *An axiomatic basis for computer programming*, Communctions Association Computing Machinery 12 (1969) 576-580.

[11] LIPTON, R.J. *A necessary and sufficient condition for the existence of Hoare logics*, 18th IEEE Symposium on Foundations of Computer Science, Providence, R.I., 1977, 1-6.

[12] MADISON, E.W., *A note on computable real fields*, J. Symbolic Logic 35 (1970) 239-241.

[13] MAL'CEV, A.I., *Constructive algebras, I.*, Russian Mathematical Surveys, 16 (1961) 77-129.

[14] _______, *Algorithms and recursive functions*, Wolters-Noordhoff, Groningen, 1970.

[15] MIRKOWSKA, G., *Algorithmic logic and its applications in the theory of programs II*, Fundamenta Informaticae 1 (1977) 147-165.

[16] RABIN, M.O., *Computable algebra, general theory and the theory of computable fields*, Transactions American Mathematical Society, 95 (1960) 341-360.

[17] ROGERS, H., *Theory of recursive functions and effective computability*, McGraw-Hill, New York, 1967.

[18] TUCKER, J.V., *Computing in algebraic systems*, Mathematical Centre, Department of Computer Science Research Report IW 130, Amsterdam, 1980.

[19] WAND, M., *A new incompleteness result for Hoare's system*, J. Association Computing Machinery, 25 (1978) 168-175.