stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 155/80 DECEMBER

J.A. BERGSTRA & J.V. TUCKER

ON THE REFINEMENT OF SPECIFICATIONS AND HOARE'S LOGIC

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, 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: 03D45, 03D80, 68B15, 03D35, 03D75, 68B10

ACM - Computing Reviews - category: 4.34, 5.24

On the refinement of specifications and Hoare's logic[*)]

by

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

ABSTRACT

We develop the basic proof theory of Hoare's logic for the partial correctness of while-programs whose underlying data types are defined by first-order axiomatic specifications. Our objective is to study the effects of refining data type specifications on the program correctness proofs they support. It is shown that any finite selection of refinements is stable relative to a given asserted program, but that this stability is a strictly local property of families of specifications.

KEY WORDS & PHRASES: *Hoare's logic, partial correctness, proof theory, first-order specifications for data types, refinements of specifications*

---

*) This paper is not for review as it is meant for publication elsewhere.

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

INTRODUCTION

Consider a programming environment with a program language, and a facility to define a data type as a set of primitive operators Σ which satisfy a set of axiomatic properties E. In the construction and maintenance of programs in such an environment once can easily be faced with the problem of "matching" a data type specification (Σ,E) to a proof that a particular program S, based upon Σ, is correct relative to particular input-output conditions p,q. Implicitly or explicitly, the verification of $\{p\}S\{q\}$ acts as a proof-theoretic criterion for the correctness of the axiomatisation E. In order to prove the asserted program, it may be necessary, or convenient, to refine a specification E into another specification E' because $\{p\}S\{q\}$ is true in an intended semantics, but the axioms E are too general (read: too weak!) to prove the appropriate information about the underlying data types.

For example, this activity of refining specifications to obtain correctness proofs will be part and parcel of any verification system whose design conforms, even superficially, to that used by Igarashi, London and Luckham in their pioneering work [12] on the Stanford PASCAL verifier. And, of course, refinement will assume prominence in systems supporting data abstraction. This is evident in the development of the PASCAL verifier [13], but in a system such as AFFIRM one finds the refinement of specifications and program verification placed on an equal footing [14].

Whatever the environment, one can enquire: *To what extent are the infinitely many ways of refining a specification independent of the required program verification? Given that a selection of specifications individually "encode" enough information to prove an asserted program* $\{p\}S\{q\}$, *surely the asserted program can be verified from the information about its data types which is common to all members of the family?* We shall formalise this question and give it an answer.

For simplicity, let us assume that the assignment and control constructs available in the language of the environment are those of <u>while</u>-programs, and that a data type is specified by a set of primitive operators Σ satisfying a collection of first-order axioms E. The partial correctness of a program belonging to such an environment can be naturally analysed by the familiar axioms and proof rules first described in HOARE [11] *providing* one allows only assertions *provable* from the specification E to govern the Rule of Consequence. The resulting formal system, based upon the first-order assertion language L over Σ, we term *Hoare's logic for the specification* E and we denote it HL(E). It is worth noticing that our simple environment and its associated Hoare logics represent precisely the theoretical foundation of the Stanford VCG for the little fragment of PASCAL determined by the while-construct.

A specification E' is said to *refine* a specification E if any assertion provable from E is provable from E'. For a family of refinements $R_E = \{E_i : i \in I\}$ of a specification E we define the *core* of R_E by

$$\mathrm{CORE}(R_E) = \{p \in L\colon E_i \vdash p, \text{ for each } i \in I\}.$$

Obviously, $E \subset \mathrm{CORE}(R_E)$. A rather straightforward formal interpretation of our question reads thus: *Given an asserted program* $\{p\}S\{q\}$ *and a specification* E, *if for each choice* E_i *from a family of refinements* R_E *we know* $\mathrm{HL}(E_i) \vdash \{p\}S\{q\}$ *then does this guarantee* $\mathrm{HL}(\mathrm{CORE}(R_E)) \vdash \{p\}S\{q\}$?

Here is the answer.

THEOREM. *Let* E *be a first-order specification and* $\{p\}S\{q\}$ *an asserted program. Let* $R_E = \{E_i : i \in I\}$ *be a family of refinements of* E *such that* $\mathrm{HL}(E_i) \vdash \{p\}S\{q\}$ *for each* $i \in I$. *If* I *is finite then the family is stable in the sense that*

$$\mathrm{HL}(\mathrm{CORE}(R_E)) \vdash \{p\}S\{q\}.$$

However, if I *is infinite then the family may well be unstable*: *there is a specification* E *and an asserted program* $\{p\}S\{q\}$, *and a countably infinite family* $R_E = \{E_i : i \in I\}$ *of refinements of* E *such that* $\mathrm{HL}(E_i) \vdash \{p\}S\{q\}$ *but* $\mathrm{HL}(\mathrm{CORE}(R_E)) \nvdash \{p\}S\{q\}$.

Once one has carefully thought through the basic proof theory of Hoare's logic, the stability of finite families of refinements is quite easy to prove. But the instability of infinite families is harder to demonstrate because the computation-theoretic ideas involved carry a number of exercises in first-order model theory as overheads. The instability result here should be compared with a theorem about arithmetical computation which was proved in our [7]: *for any asserted program* $\{p\}S\{q\}$ *and any infinite family* $R = \{E_i : i \in I\}$ *of refinements of Peano Arithmetic, if* $HL(E_i) \vdash \{p\}S\{q\}$ *for each* $i \in I$ *then* $HL(CORE(R)) \vdash \{p\}S\{q\}$. This is a particularly pleasing result since Peano Arithmetic is merely a refinement of the standard algebraic specification for arithmetic, designed to generate those assertions provable by induction.

After some preliminaires, we define Hoare's logic for a specification and look at its proof theory; in Section 3 we prove the theorem. It should be noticed that the theorem does not concern semantical questions and that this gives it a certain novelty in the theoretical literature on program correctness. Most theoretical investigations and applications of Hoare's ideas about axiomatisation have contained a strong semantic bias since Cook's study [8]; see, for example, DE BAKKER [2] and the invaluable survey APT [1]. Practice, on the other hand, seems to have been preoccupied with proof theory.

This note is part of a series of articles about Hoare's logic and data type specifications: various incompleteness and completeness properties of the logic are re-examined in [4,6]; algebraic specifications are studied in [5]; and the proof theory of the logic over arithmetic is the subject of [7]. All these articles derive from [3], written in collaboration with J. Tiuryn, about the use of correctness formulae in defining the semantics of a programming system with first-order specified data types; but strictly speaking, none are required reading for the present paper.

We gratefully acknowledge conversations about verification with our colleagues R.J.R. Back and A. de Bruin.

1. ASSERTIONS, SPECIFICATIONS AND PROGRAMS

Prerequisite to any study of Hoare's logic are the primary sources HOARE [11] and COOK [8], but the reader would do well to consult the survey article APT [1].

The first-order language $L = L(\Sigma)$ of some signature Σ is based upon a set of variables $x_1, x_2, \ldots$ and its constant, function and relational symbols are those of Σ together with the boolean constants <u>true</u>, <u>false</u> and the equality relation. We assume L possesses the usual logical connectives and quantifiers; and the set of all algebraic expressions of L we denote $T(\Sigma)$.

If E is a set of assertions of L then the set of all formal theorems of E is denoted Thm(E); we write $E \vdash p$ for $p \in \mathrm{Thm}(E)$. Such a set E of formulae is usually called a theory, but in the present context we obviously need the more suggestive term *specification*, for L will serve as both an assertion/program specification language and a data type specification language.

1.1. <u>DEDUCTION THEOREM</u>. *Let* E *be a specification and let* p,q *be assertions. Then the following are equivalent*:
(1) $E \cup \{\hat{p}\} \vdash q$
(2) $E \vdash \hat{p} \to q$.
where β *is the universal closure of* p.

A specification E' is a *refinement* of a specification E if $\mathrm{Thm}(E) \subset \mathrm{Thm}(E')$. And two specifications E,E' are (*logically*) *equivalent* if $\mathrm{Thm}(E) = \mathrm{Thm}(E')$. If E is a specification and $R_E = \{E_i : i \in I\}$ is a family of refinements of E then we define the *core* of R_E by

$$\mathrm{CORE}(R_E) = \bigcap_{i \in I} \mathrm{Thm}(E_i).$$

Using the syntax of L, the set $\mathcal{WP} = \mathcal{WP}(\Sigma)$ of all <u>while</u>-programs over Σ is defined in the customary way.

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

Such are the ingredients of Hoare's logic for a specification, but we also need their semantics in the proof of the theorem. Let us summarize the meanings for the various components and remark on the use of L as a data type specification language.

The semantics of a signature is a structure. For any structure A of signature Σ, the semantics of the first-order language L over Σ as determined by A has its standard definition in model theory and this we assume to be understood. The validity of $p \in L$ over structure A we write $A \models p$. The class of all models of a specification E is denoted Mod(E); we write $\text{Mod}(E) \models p$ to mean that for every $A \in \text{Mod}(E)$, $A \models p$. Gödel's Completeness Theorem says this about specifications:

$$E \vdash p \text{ if, and only if, } \text{Mod}(E) \models p.$$

As far as the proof theory of a data type axiomatisation E is concerned, the semantics of the specification *is* Mod(E).

So consider the algebraic specification methods for data types where one invariably has a *particular* semantic model in mind for a specification. Following ADJ[9], it is usual to settle on the initial model I(E) of Mod(E) as the unique meaning for an algebraic axiomatisation E. The logic of E is oblivious of this (or any other) particular choice because it yields only those facts true in *all* models of E. Refinements are a natural accessory of algebraic specifications: one starts with a simple algebraic specification (Σ,E) to establish the correctness of the desired data type semantics A and then adds to E various assertions true in A as the need arises in program correctness proofs (say). Peano arithmetic illustrates this perfectly. Refinements are also a necessary accessory of algebraic specifications for although the algebraic methods can define virtually any data type one wants, the kinds of assertion provable from algebraic formulae are rather restricted; see [5] for a thorough discussion of this problem.

For the semantics of $\mathcal{WP}$ as determined by a structure A, we leave the reader free to choose any sensible account of while-program computations which applies to an arbitrary structure: COOK [8]; the graph-theoretic semantics in GREIBACH [10]; the denotational semantics described in DE BAKKER [2]. What constraint must be placed on this choice is merely the necessity of verifying the soundness of Hoare's logic (Theorem 2.9).

To the asserted programs we assign *partial correctness semantics*: the asserted program $\{p\}S\{q\}$ is *valid on a structure* A (in symbols: $A \models \{p\}S\{q\}$) if for each initial state $a \in \text{states}(A)$, $A \models p(a)$ implies either $S(a)$ terminates and $A \models q(S(a))$ or $S(a)$ diverges. And the asserted program $\{p\}S\{q\}$ is *valid for a specification* E if it is valid on *every* model of E;

in symbols, $E \models \{p\}S\{q\}$ or $\mathrm{Mod}(E) \models \{p\}S\{q\}$.

The *partial correctness theory of a structure* A is the set

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

and the *partial correctness theory of a specification* E is the set

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

Clearly,

$$PC(E) = \cap_{A \in \mathrm{Mod}(E)} PC(A).$$

2. HOARE'S LOGIC

Hoare's logic for $\mathcal{WP} = \mathcal{WP}(\Sigma)$ with assertion language $L = L(\Sigma)$ and specification $E \subset L$, 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 scheme: for $e \in T(\Sigma)$ and x a variable of L, the asserted program

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

is an axiom, where $p[e/x]$ stands for the result of substituting e for free occurrences of x in p.

2. Composition rule:

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

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. Specification axiom: Each member of Thm(E) is an axiom.

The set of asserted programs derivable from these axioms by the proof rules we denote HL(E), but if E = {t} for some $t \in L$ then we use HL(t). As usual we write HL(E) $\vdash$ {p}S{q} in place of {p}S{q} $\in$ HL(E).

2.1. REFINEMENT LEMMA. *Let* E *and* E' *be specifications. If* E' *is a refinement of* E *then* HL(E) $\subset$ HL(E'). *Thus, if* E *and* E' *are equivalent specifications then* HL(E) = HL(E').

Actually, it is this first lemma which authorises our use of the term *refinement* in the present context for our interest in the logic of specifications and assertions is shaped by the logic of partial correctness it must support. Lemma 2.1 is obviously true as, indeed, are the next two proof-theoretical facts:

2.2. FINITENESS LEMMA. *Let* E *be a specification and* {p}S{q} *an asserted program. If* HL(E) $\vdash$ {p}S{q} *then there is a finite set* $F = \{t_i : i \in I\}$ *of assertions such that* E $\vdash$ t_i *for each* $i \in I$ *and* HL(F) $\vdash$ {p}S{q}; *in particular, there is a single assertion* $t = \bigwedge_{i \in I} t_i$ *such that* E $\vdash$ t *and* HL(t) $\vdash$ {p}S{q}.

2.3. PROOF DECOMPOSITION LEMMA. *Let* E *be a specification and let* p,q *be assertions. Then*

(1) Assignment: $HL(E) \vdash \{p\}x := e\{q\}$ *if, and only if,* $E \vdash p \to q[e/x]$

(2) Composition: $HL(E) \vdash \{p\}S_1;S_2\{q\}$ *if, and only if, for some assertion* r,
$HL(E) \vdash \{p\}S_1\{r\}$ *and* $HL(E) \vdash \{r\}S_2\{q\}$

(3) Conditional: $HL(E) \vdash \{p\}$ if b then S_1 else S_2 fi$\{q\}$ *if, and only if,*
$HL(E) \vdash \{p\wedge b\}S_1\{q\}$ *and* $HL(E) \vdash \{p\wedge\neg b\}S_2\{q\}$

(4) Iteration: $HL(E) \vdash \{p\}$ while b do S_0 od $\{q\}$ *if, and only if, for some assertion* r,
$E \vdash p \to r$, $HL(E) \vdash \{r\wedge b\}S\{r\}$, *and* $E \vdash r \wedge \neg b \to q$.

The ease with which one can calculate in a formal system is decided by its derived rules. Hoare's logic enjoys many derived rules which turn natural semantical properties into formal proof-theoretical laws with few syntactical concessions. We shall list a few of these rules, but we will prove only the first.

2.4. LEMMA. *Let* E *be a specification and* $\{\{p_i\}S\{q_i\}: i \in I\}$ *a finite set of asserted programs. Then the following is a derived rule of* HL(E)

$$\frac{\{p_i\}S\{q_i\} \text{ for each } i \in I}{\{\bigvee_{i\in I} p_i\}S\{\bigvee_{i\in I} q_i\}}\ .$$

PROOF. This is proved by induction on the structure of S for which the basis is the assignment statement.

Assignment: $S ::= x := e$. Assume $HL(E) \vdash \{p_i\}x := e\{q_i\}$ for each $i \in I$. Then by Lemma 2.3, $E \vdash p_i \to q_i[e/x]$ for each $i \in I$. We can now calculate formally as follows:

$$E \vdash \bigwedge_{i\in I}(p_i \to q_i[e/x])$$
$$E \vdash \bigwedge_{i\in I}(p_i \to \bigvee_{i\in I} q_i[e/x])$$
$$E \vdash (\bigvee_{i\in I} p_i) \to (\bigvee_{i\in I} q_i[e/x]).$$

Whence $HL(E) \vdash \{\bigvee_{i\in I} p_i\}x := e\{\bigvee_{i\in I} q_i\}$ by Lemma 2.3.

The induction step divides into 3 cases.

Composition: $S ::= S_1;S_2$. Assume $HL(E) \vdash \{p_i\}S_1;S_2\{q_i\}$ for each $i \in I$. Then by Lemma 2.3, there exist assertions $\{r_i : i \in I\}$ such that for each $i \in I$,

$$HL(E) \vdash \{p_i\}S_1\{r_i\} \quad \text{and} \quad HL(E) \vdash \{r_i\}S_2\{q_i\}.$$

By the induction hypothesis,

$$HL(E) \vdash \{\bigvee_{i\in I} p_i\}S_1\{\bigvee_{i\in I} r_i\} \text{ and } HL(E) \vdash \{\bigvee_{i\in I} r_i\}S_2\{\bigvee_{i\in I} q_i\}$$

and by Lemma 2.3, we deduce

$$HL(E) \vdash \{\bigvee_{i\in I} p_i\}S_1;S_2\{\bigvee_{i\in I} q_i\}.$$

<u>Conditional</u>: S ::= <u>if</u> b <u>then</u> S_1 <u>else</u> S_2 <u>fi</u>. Assume $HL(E) \vdash \{p_i\}S\{q_i\}$ for each $i \in I$. Then by Lemma 2.3, we know that for each $i \in I$,

$$HL(E) \vdash \{p_i \wedge b\}S_1\{q_i\} \quad \text{and} \quad HL(E) \vdash \{p_i \wedge \neg b\}S_2\{q_i\}.$$

By the induction hypothesis,

$$HL(E) \vdash \{\bigvee_{i\in I}(p_i \wedge b)\}S_1\{\bigvee_{i\in I} q_i\} \text{ and } HL(E) \vdash \{\bigvee_{i\in I}(p_i \wedge \neg b)\}S_2\{\bigvee_{i\in I} q_i\}.$$

Since $\wedge$ distributes over $\vee$, we can rewrite these formal theorems as

$$HL(E) \vdash \{(\bigvee_{i\in I} p_i) \wedge b\}S_1\{\bigvee_{i\in I} q_i\} \text{ and } HL(E) \vdash \{(\bigvee_{i\in I} p_i) \wedge \neg b\}S_2\{\bigvee_{i\in I} q_i\}$$

and thanks to Lemma 2.3 we are done.

<u>Iteration</u>: S ::= <u>while</u> b <u>do</u> S_0 <u>od</u>. Assume $HL(E) \vdash \{p_i\}S\{q_i\}$ for each $i \in I$. Then by Lemma 2.3, there exist assertions $\{r_i : i \in I\}$ such that for each $i \in I$,

$$E \vdash p_i \rightarrow r_i \quad HL(E) \vdash \{r_i \wedge b\}S_0\{r_i\} \text{ and } E \vdash r_i \wedge \neg b \rightarrow q_i.$$

Obviously, we can obtain by simple logical calculations the theorems

$$E \vdash \bigvee_{i\in I} p_i \rightarrow \bigvee_{i\in I} r_i \quad \text{and} \quad E \vdash (\bigvee_{i\in I} r_i) \wedge \neg b \rightarrow \bigvee_{i\in I} q_i$$

and as the induction hypothesis applied to S_0 yields

$$HL(E) \vdash \{(\bigvee_{i\in I} r_i) \wedge b\} S_0 \{\bigvee_{i\in I} r_i\}$$

(with the help of the distribution law for $\wedge$ over $\vee$) we are done by Lemma 2.3.

Q.E.D.

2.5. LEMMA. *Let* E *be a specification and* $\{\{p_i\}S\{q_i\}: i \in I\}$ *be a finite set of asserted programs. Then the following is a derived rule of* HL(E)

$$\frac{\{p_i\}S\{q_i\} \quad \textit{for each } i \in I}{\{\bigwedge_{i\in I} p_i\}S\{\bigwedge_{i\in I} q_i\}.}$$

2.6. LEMMA. *Let* E *be a specification and* $\{p\}S\{q\}$ *an asserted program. Then the following is a derived rule of* HL(E)

$$\frac{\{p\}S\{q\}}{\{\exists y.p\}S\{q\}}$$

where y *is not a variable of* S *and not a free variable of* q.

The following theorem is quite fundamental for any reasoning *about* Hoare's logic for a specification.

2.7. DEDUCTION LEMMA. *Let* E *be a specification, let* t *be (the universal closure of) an assertion and let* $\{p\}S\{q\}$ *be an asserted program. Then the following are equivalent*

(1) $HL(E\cup\{t\}) \vdash \{p\}S\{q\}$

(2) $HL(E) \vdash \{t\wedge p\}S\{q\}$.

PROOF. That (2) implies (1) is obvious: clearly, $E \vdash t \rightarrow (p \rightarrow t\wedge p)$ and so by the Deduction Theorem 1.1 for first-order logic $E \cup \{t\} \vdash p \rightarrow t \wedge p$. From (2) and the Refinement Lemma 2.1 we know that $HL(E\cup\{t\}) \vdash \{t\wedge p\}S\{q\}$;

thus by the Rule of Consequence it follows that $HL(E \cup \{t\}) \vdash \{p\}S\{q\}$. The reverse implication is proved by induction on the structure of S for which the basis is the assignment statement.

<u>Assignment</u>: $S ::= x := e$. Assume $HL(E\cup\{t\}) \vdash \{p\}x := e\{q\}$. Then $E \cup \{t\} \vdash p \to q[e/x]$ by Lemma 2.3 and $E \vdash t \to (p \to q[e/x])$ by the Deduction Theorem 1.1 for first-order logic. But $E \vdash t \wedge p \to q[e/x]$ and so $HL(E) \vdash \{t \wedge p\}x := e\{q\}$ by Lemma 2.3.

The induction step divides into 3 cases.

<u>Composition</u>: $S ::= S_1;S_2$. Assume $HL(E\cup\{t\}) \vdash \{p\}S_1;S_2\{q\}$. By Lemma 2.3, there is some assertion r such that

$$HL(E\cup\{t\}) \vdash \{p\}S_1\{r\} \text{ and } HL(E\cup\{t\}) \vdash \{r\}S_2\{q\}.$$

Now $E \cup \{t\} \vdash r \to (t \wedge r)$ and so $HL(E\cup\{t\}) \vdash \{p\}S_1\{t \wedge r\}$ by the Rule of Consequence. By the induction hypothesis,

$$HL(E) \vdash \{t \wedge p\}S_1\{t \wedge r\} \text{ and } HL(E) \vdash \{t \wedge r\}S_2\{q\}$$

and hence $HL(E) \vdash \{t \wedge p\}S_1;S_2\{q\}$ by the Composition Rule.

<u>Conditional</u>: $S ::= \underline{if}\ b\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}$. Assume $HL(E\cup\{t\}) \vdash \{p\}S\{q\}$. By Lemma 2.3, we know that

$$HL(E\cup\{t\}) \vdash \{p \wedge b\}S_1\{q\} \text{ and } HL(E\cup\{t\}) \vdash \{p \wedge \neg b\}S_2\{q\}.$$

By the induction hypothesis,

$$HL(E) \vdash \{(t \wedge p) \wedge b\}S_1\{q\} \text{ and } HL(E) \vdash \{(t \wedge p) \wedge \neg b\}S_2\{q\}$$

and hence $HL(E) \vdash \{t \wedge p\}S\{q\}$ by the Conditional Rule.

<u>Iteration</u>: $S ::= \underline{while}\ b\ \underline{do}\ S_0\ \underline{od}$. Assume $HL(E\cup\{t\}) \vdash \{p\}S\{q\}$. By Lemma 2.3, there is some assertion r such that

$$E \cup \{t\} \vdash p \to r \quad HL(E\cup\{t\}) \vdash \{r\wedge b\}S_0\{r\} \text{ and } E \cup \{t\} \vdash r \wedge \neg b \to q.$$

Now $E \cup \{t\} \vdash r \to t \wedge r$ so applying the Rule of Consequence to the asserted program, and the Deduction Theorem 1.1 for first-order logic to the logical theorems, we obtain

$$E \vdash t \to (p\to r) \quad HL(E\cup\{t\}) \vdash \{r\wedge b\}S_0\{t\wedge r\} \text{ and } H \vdash t \to (r\wedge\neg b\to q)$$

and with some further logical rewriting and the induction hypothesis applied to S_0 we get

$$E \vdash t \wedge p \to t \wedge r \quad HL(E) \vdash \{(t\wedge r)\wedge b\}S_0\{t\wedge r\} \text{ and } E \vdash (t\wedge r) \wedge\neg b \to q.$$

By the iteration clause of Lemma 2.3, $HL(E) \vdash \{t\wedge p\}S\{q\}$.

Q.E.D.

The following fact has an essential rôle to play in the proof of our theorem.

2.8. LEMMA. *Let* E *be a specification and let* $\{t_i : i \in I\}$ *be a finite set of assertions. If* $HL(E\cup\{t_i\}) \vdash \{p\}S\{q\}$ *for each* $i \in I$ *then* $HL(E\cup\{\bigvee_{i\in I} t_i\}) \vdash \{p\}S\{q\}$.

PROOF. Assume $HL(E\cup\{t_i\}) \vdash \{p\}S\{q\}$ for each $i \in I$. Then by the Deduction Lemma 2.7, $HL(E) \vdash \{t_i\wedge p\}S\{q\}$ for each $i \in I$. By the derived rule Lemma 2.4, we have $HL(E) \vdash \{(\bigvee_{i\in I} t_i)\}S\{q\}$ and so the result follows by the Deduction Lemma 2.7.

Q.E.D.

And finally we record this well known theorem which will be needed for technical reasons in the next section.

2.9. SOUNDNESS THEOREM. *Let* E *be a specification. Then* $HL(E) \subset PC(E)$.

This is what is said in the corollary to Theorem 1 in COOK [8].

3. PROOF OF THE THEOREM

Let $R_E = \{E_i : i \in I\}$ be a finite family of refinements of the specification E and assume that $HL(E_i) \vdash \{p\}S\{q\}$ for each $i \in I$. By the Finiteness Lemma 2.2, we can choose assertions t_i such that $E_i \vdash t_i$ and $HL(t_i) \vdash \{p\}S\{q\}$ for each $i \in I$. By Lemma 2.8, we know $HL(\bigvee_{i\in I} t_i) \vdash \{p\}S\{q\}$, but $\bigvee_{i\in I} t_i \in CORE(R_E)$ and so we are done.

Now consider the case of an infinite family of refinements. Our counter-example is combinatorially related to two-way unbounded lists and arrays and it could be described exlcusively in terms of such structures. For technical clarity, however, we have found that our argument is better served by the example's looser relationship with arithmetic.

The basic specification is (Σ, E) where $\Sigma = \{a,b,N,L\}$ and a,b are constants and N,L are unary operator symbols; and E contains two algebraic axioms

$$NL(X) = X \quad \text{and} \quad LN(X) = X.$$

The models of E are precisely those structures composed of a set equipped with a permutation, its inverse and two distinguished points. But for the moment one may think of N,L as the next and last operators on *two* lists with roots a,b. For example, the initial algebra of Mod(E) picks out the model depicted in Figure 3.1 which we identify with two copies of integer arithmetic $\mathbb{Z} \wedge \mathbb{Z}$.

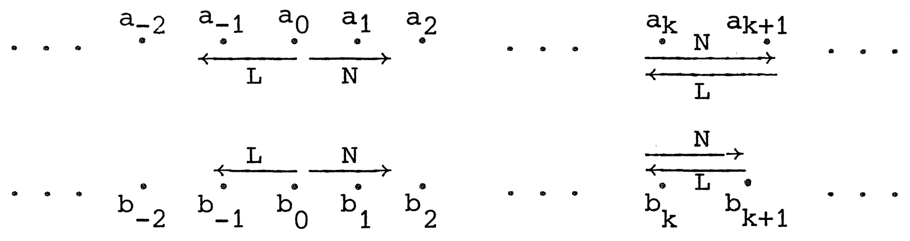

Figure 3.1

The asserted program $\{p\}S\{q\}$ we shall study is defined by

$$S ::= \underline{\text{while}}\ x \neq b\ \underline{\text{do}}\ x := N(x)\ \underline{\text{od}}$$

$$p \equiv x = a \quad \text{and} \quad q \equiv \underline{\text{false}}\ .$$

If $\{p\}S\{q\}$ were provable then this would guarantee that a,b are the roots of distinct lists, or arithmetics, as one can neither move, or count, up from a to b nor down from b to a. Notice that $\{p\}S\{q\}$ is valid on the initial model $\mathbb{Z} \wedge \mathbb{Z}$ because $S(a_0)$ diverges, but it is *not* provable in HL(E) because it is not valid in a model of E such as

$$k\text{-}\mathbb{Z} = (\{\ldots,-2,-1,0,1,2,\ldots\};\ 0,k,x+1,x-1)$$

where a names 0, and b names k and $k \geq 0$ (by the Soundness Theorem 2.9).

Let $E_i = E \cup \{N^j(a) \neq b\colon 0 \leq j < i\} \cup \{N^i(a) = a\}$ for $i \in \omega$. The axioms of E_i are intended to force S to diverge on any input named by a because they introduce a cycle of length i generated by N applied to a from which b is procluded. Notice E_i is not valid in $\mathbb{Z} \wedge \mathbb{Z}$ but it is valid in $\mathbb{Z}_i \wedge \mathbb{Z}$ where $\mathbb{Z}_i$ is integer arithmetic modulo i.

3.1. LEMMA. *For each* $i \in \omega$, $HL(E_i) \vdash \{p\}S\{q\}$.

PROOF. Now $S ::= \underline{\text{while}}\ x \neq b\ \underline{\text{do}}\ x := N(x)\ \underline{\text{od}}$ so consider the body $x := N(x)$. By the Assignment Axiom Scheme, we know that for $j < i$

$$HL(E_i) \vdash \{(x=N^{j+1}(a))[N(x)/x]\}\ x := N(x)\{x=N^{j+1}(a)\}.$$

But the precondition is just $N(x) = N^{j+1}(a)$ and trivially $E_i \vdash x = N^j(a) \rightarrow N(x) = N^{j+1}(a)$. By the Rule of Consequence, we know that for $j < i$

$$HL(E_i) \vdash \{x = N^j(a)\}\ x := N(x)\{x = N^{j+1}(a)\}$$

and by Lemma 2.4

$$HL(E_i) \vdash \{\textstyle\bigvee_{j=0}^{i-1} x = N^j(a)\}\ x := N(x)\ \{\textstyle\bigvee_{j=0}^{i-1} x = N^{j+1}(a)\}.$$

Because $E_i \vdash N^i(a) = a$ and $E_i \vdash \bigvee_{j=0}^{i-1} x = N^{j+1}(a) \rightarrow \bigvee_{j=0}^{i-1} x = N^j(a)$ we can apply the Rule of Consequence to obtain

$$HL(E_i) \vdash \{r \wedge x \neq a\}\ x := N(x)\{r\}$$

wherein $r \equiv \bigvee_{j=0}^{i-1} x = N^j(a)$. By the Iteration Rule, we derive

$$HL(E_i) \vdash \{r\}\ \underline{\text{while}}\ x \neq a\ \underline{\text{do}}\ x := N(x)\ \underline{\text{od}}\ \{r \wedge \neg(x \neq a)\}$$

and since $E_i \vdash p \rightarrow r$ and $E_i \vdash (r \wedge \neg(x \neq a)) \rightarrow q$, the Rule of Consequence yields

$$HL(E_i) \vdash \{p\}S\{q\}$$

Q.E.D.

To complete the proof of the theorem we have to demonstrate this next fact:

3.2. LEMMA. *If* $R_E = \{E_i : i \in \omega\}$ *then* $HL(CORE(R_E)) \nvdash \{p\}S\{q\}$.

PROOF. Assume for a contradiction that $HL(CORE(R_E)) \vdash \{p\}S\{q\}$. Then by the Finiteness Lemma 2.2 we may choose an assertion $t \in CORE(R_E)$ such that

(1) $\quad HL(t) \vdash \{p\}S\{q\}$.

For this statement (1) we shall find a contradiction.

Let D be the following set of assertions which are intended to rule out finite cycles in the operator N and to ensure a and b are mutually inaccessible:

$$D = \{N^i(X) \neq X : i \in \omega\} \cup \{N^i(a) \neq b,\ N^i(b) \neq a : i \in \omega\}.$$

For example, D is valid in $\mathbb{Z} \wedge \mathbb{Z}$, but we wish to show that the specification $E \cup \{t\} \cup D$ has a model in order to guarantee the consistency of (1) with the special requirements on the operator N.

3.3. <u>LEMMA</u>. *The specification* $E \cup \{t\} \cup D$ *has a model.*

<u>PROOF</u>. We use the Compactness Theorem. Any finite subset of $T = E \cup \{t\} \cup D$ is included in a finite initial segment $T_K = E \cup \{t\} \cup D_K$ where

$$D_K = \{N^i(X) \neq X: 0 \leq i < K\} \cup \{N^i(a) \neq b, N^i(b) \neq a: 0 \leq i < K\}$$

and K is sufficiently large. Consider the structure $\mathbb{Z}_K \wedge \mathbb{Z}_K$ made from two copies of integer arithmetic mod K and depicted in Figure 3.2

$$\begin{tikzcd} a_0 \arrow[r, shift left, "N"] & a_1 \arrow[l, shift left, "L"] & \cdots & a_{K-1} \arrow[r, shift left, "N"] & a_0 \arrow[l, shift left, "L"] & b_0 \arrow[r, shift left, "N"] & b_1 \arrow[l, shift left, "L"] & \cdots & b_{K-1} \arrow[r, shift left, "N"] & b_0 \arrow[l, shift left, "L"] \end{tikzcd}$$

<u>Figure</u> 3.2

Clearly, $A \models D_K$ but, in addition, $A \models E_K$ and hence $A \models E \cup \{t\}$. Thus T_K has a model. Since every finite subset of T has a model, T has a model by the Compactness Theorem. Q.E.D.

We now need a technical fact about the relationship between t and $E \cup D$.

3.4. <u>LEMMA</u>. *The specification* $E \cup D$ *admits quantifier elimination: for each assertion* $r \in L$ *there is a quantifier-free assertion* r^* *such that* $E \cup D \vdash r \leftrightarrow r^*$.

<u>PROOF</u>. Let $T = E \cup D$. Now T is a universally axiomatised first-order theory so, by a theorem of Robinson, if T is model-complete then T admits quantifier elimination (see SACKS [15, p.67]). Another theorem of Robinson says that T is model complete if, and only if, for each model A of T, $T \cup$ DIAGRAM(A) is complete (SACKS [15, p.36]). It is a routine matter to prove that for any model A of T, the set of formulae $T \cup$ DIAGRAM(A) is ω_1-categorical. Thus, by the Łoś-Vaught Test (SACKS [15, p.34]) this set of assertions is complete. Q.E.D.

Using Lemma 3.4 we can choose a quantifier-free assertion t^* such that

$E \cup D \vdash t \leftrightarrow t^*$ and then choose a finite subset D^* of D such that $E \cup D^* \vdash t \leftrightarrow t^*$.

We shall construct a structure A which is a model for $E \cup \{t^*\} \cup D^*$ and in which for some $\ell \in \omega$, $A \models N^\ell(a) = b$. Assuming this is done, the contradiction to statement (1) is soon found:

Clearly $A \not\models \{p\}S\{q\}$ because S can terminate in ℓ steps from $x = a$ in A. Since $E \cup D^* \vdash t \leftrightarrow t^*$, we have that $A \models t \leftrightarrow t^*$ and $A \models t^*$ and so $A \models t$. Thus, $\{p\}S\{q\}$ is invalid on a model of $E \cup \{t\}$. By the Soundness Theorem 2.9

$$HL(E\cup\{t\}) \not\vdash \{p\}S\{q\}$$

and so obviously $HL(t) \not\vdash \{p\}S\{q\}$ which is the required contradiction.

3.5. LEMMA. *The specification* $E \cup \{t^*\} \cup D^*$ *has a model in which for some* $\ell \in \omega, F^\ell(a) = b$.

PROOF. By Lemma 3.3, $E \cup \{t\} \cup D$ has a model B and since $E \cup D \vdash t \leftrightarrow t^*$ we know $E \cup \{t^*\} \cup D$ is valid in B. Now it is straightforward to check that the substructure of B generated by the constants a, b is isomorphic to $\mathbb{Z} \wedge \mathbb{Z}$; and since $E \cup \{t^*\} \cup D$ consists of *universal axioms only* it is the case that

$$\mathbb{Z} \wedge \mathbb{Z} \models E \cup \{t^*\} \cup D.$$

(Here we need the simplification of t to t^*, of course.)

Consider the map $\phi_k : \mathbb{Z} \wedge \mathbb{Z} \to \mathbb{Z} \wedge \mathbb{Z}$ defined by

$$\phi_k(a_i) = a_i \quad \text{and} \quad \phi_k(b_i) = a_{i+k}.$$

Each ϕ_k is an endomorphism of $\mathbb{Z} \wedge \mathbb{Z}$ and obviously

$$\phi_k(\mathbb{Z}\wedge \mathbb{Z}) \models F^k(a) = b.$$

By inspection, we can choose some k sufficiently large to guarantee that

$$\phi_k(\mathbb{Z}\wedge\mathbb{Z}) \models E \cup \{t^*\} \cup D^*.$$

To see that these extra axioms can be satisfied we consider each of the 3 sets in turn. First, $\phi_k(\mathbb{Z}\wedge\mathbb{Z}) \models E$ for *any* k because E contains only equations and ϕ_k is a homomorphism. Next, consider the quantifier-free assertion t^*. If one chooses $k > L = \|t^*\|$, the length of t^*, then ϕ_k cannot identify any of the inequalities making up t^*. It is easy to see in this case that for $k > L$, $\phi_k(\mathbb{Z}\wedge\mathbb{Z}) \models t^*$. Thirdly, since D^* is finite it is included in some finite segment D_K of D as defined in the proof of Lemma 3.3. If one chooses $k > K$ then $\phi_k(\mathbb{Z}\wedge\mathbb{Z}) \models D^*$ because no loops are introduced "below" k in the sense that

$$\phi_k(\mathbb{Z}\wedge\mathbb{Z}) \models \{F^i(a) \neq b,\ F^i(b) \neq a\colon 0 \leq i < k\}.$$

Therefore, choosing some $\ell > \max(L,K)$ leads to a model $A = \phi_\ell(\mathbb{Z}\wedge\mathbb{Z})$ such that

$$A \models E \cup \{t^*\} \cup D^* \cup \{F^\ell(a) = b\}.$$

Q.E.D.

REFERENCES

[1] APT, K.R., *Ten years of Hoare's logic, a survey* in F.V. JENSEN, B.H. MAYOH and K.K. MØLLER (eds), *Proceedings from 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] DE BAKKER, J.W., *Mathematical theory of program correctness,* Prentice-Hall International, London, 1980.

[3] BERGSTRA, J.A., J. TIURYN & J.V. TUCKER, *Floyd's principle, correctness theories and program equivalence,* Mathematical Centre, Department of Computer Science Research Report IW 145, Amsterdam, 1980. (To appear in Theoretical Computer Science.)

[4] BERGSTRA, J.A. & J.V. TUCKER, *Some natural structures which fail to possess a sound and decidable Hoare-like logic for their* while-*programs* (To appear in Theoretical Computer Science. An earlier edition of this paper is registered at the Mathematical Centre as report IW 136/80).

[5] BERGSTRA, J.A. & J.V. TUCKER, *Algebraically specified programming systems and Hoare's logic,* Mathematical Centre, Department of Computer Science Research Report IW 143, Amsterdam, 1980.

[6] BERGSTRA, J.A. & J.V. TUCKER, *Expressiveness and the completeness of Hoare's logic,* Mathematical Centre, Department of Computer Science Research Report IW 149, Amsterdam, 1980.

[7] BERGSTRA, J.A. & J.V. TUCKER, *Hoare's logic and Peano's arithmetic,* Mathematical Centre, Department of Computer Science Research Report, Amsterdam, (in preparation).

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

[9] GOGUEN, J.A., J.W. THATCHER & E.G. WAGNER, *An initial algebra approach to the specification, correctness and implementation of abstract data types,* in R.T. YEH (ed.), *Current trends in programming methodology* IV, *Data structuring,* Prentice-Hall, Engelwood Cliffs, New Jersey, 1978, 80-149.

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

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

[12] IGARASHI, S., R.L. LONDON & D.C. LUCKHAM, *Automatic program verification I: a logical basis and its implementation,* Acta Informatica 4 (1975) 145-182.

[13] LUCKHAM, D.C. & N. SUZUKI, *Verification of array, record and pointer operations in PASCAL,* ACM-Transactions on Programming Languages and Systems 1 (1979) 226-244.

[14] MUSSER, D.R., *Abstract data type specification in the AFFIRM system,* IEEE Transactions on Software Engineering 6(1) (1980) 24-32.

[15] SACKS, G.E., *Saturated model theory,* W.A. Benjamin, Inc., Reading, Massachusetts, 1972.