LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TM-308

FLOYD-HOARE LOGIC DEFINES SEMANTICS

ALBERT R. MEYER

MAY 1986

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

Floyd-Hoare Logic Defines Semantics

Albert R. Meyer
M.I.T. Lab. for Computer Science
545 Technology Square
Cambridge, MA 02139, USA
ARPANET: MEYER@MIT-XX

23 May 1986

Abstract. The first-order partial correctness assertions provable in Floyd-Hoare logic about an uninterpreted **while**-program scheme determine the scheme up to equivalence. This settles an open problem of Meyer and Halpern. The simple proof of this fact carries over to other partial correctness axiomatizations given in the literature for wider classes of ALGOL-like program schemes.

Keywords: Partial correctness, relative completeness, deduction theorem, Skolem, program scheme.

General Terms: Languages, verification, theory.

CR Categories: D.3.1 [**Programming Languages**]: Formal Definitions and Theory -- *semantics, syntax*; D.3.2 [**Programming Languages**]: Language Classifications -- *applicative languages, Lisp.*

This work was supported in part by NSF Grant No. A511190-DCR and by ONR grant No. N00014-83-K-0125. This report is a nearly identical to a paper to appear in the IEEE Symposium on Logic in Computer Science, Cambridge, MA, June, 1986.

Table of Contents

1. Introduction.

Can the semantics of a programming language be specified by axioms for proving assertions about programs? Without restrictions on the assertion language and rules of proof, a positive answer is straightforward (*cf.* [10]). However, in the fundamental case of partial correctness assertions with *first-order* pre- and post-conditions, the story is more complicated. Bergstra, Tiuryn, Tucker [3], and independently, Meyer, Halpern [18], confirmed this author's conjecture that the first-order partial correctness theory of a program scheme is theoretically sufficient to determine the scheme up to equivalence. This theorem applies to a quite general notion of *recursively enumerable* (r.e.) schemes -- infinite, nondeterministic r.e. flowchart schemes whose assignment statements may include array assignments, *e.g.*, of the form $a[x]:=z$, and random assignments *e.g.*, of the form $x:=?$, and whose tests may be any first-order formula (not necessarily open) of predicate calculus with equality.

Moreover, both Bergstra, *et al.* and Meyer/Halpern also observed that there are *r.e.* sets of valid first-order partial correctness assertions which determine the semantics of **while**-program schemes. Thus, in the abstract sense in which any r.e. set may be regarded as a proof system, an axiomatic semantics (as opposed to denotational or operational semantics, *cf.* [10]) for **while**-programs could be said to have been obtained. Nevertheless, the r.e. sets described in [3], [18] were not generated by actual proof systems, and the problem of finding a reasonable system of axioms and rules which proved enough valid partial correctness assertions to determine the semantics of **while**-programs was left open ([18], p. 556). In particular, it was not even known whether the most familiar proof system for proving partial correctness of **while**-programs, the Floyd-Hoare system [7], [13], determined semantics.

Leivant [14] recently proved that the Floyd-Hoare axiom system does indeed provide an axiomatic semantics for **while**-programs. His proof combines (i) a new, simplified proof for the special case of **while**-program schemes of the main theorem (restated as Proposition 1 below) about axiomatic semantics of r.e schemes, together with (ii) a characterization of the proof-theoretic power of Hoare's logic sketched in [15] (see also [17]).[1]

In this note we piece together a few familiar facts about Floyd-Hoare logic, which, together with the *statement* -- as opposed to a proof -- of Proposition 1, provide an easy proof that this logic is sufficient to determine semantics of **while**-programs. Moreover, the general properties of Floyd-Hoare logic used in the proof are possessed by many extensions of Floyd-Hoare logic to richer programming languages with features such as higher-order recursive procedures and blocks with local variables (*cf.* [6], [4], [1], [5]), so

---

[1] Bergstra and Klop [2] present an extended study showing that Floyd-Hoare logic determines semantics of **while**-programs interpreted over single structures, but their results do not seem to cover uninterpreted schemes.

we may conclude that these logics too provide axiomatic semantics for these richer languages.

We assume the reader has seen some version of the Floyd-Hoare axioms (given in many of the references); the exact details are not needed. In the next section we briefly review some key notations and definitions.

2. Syntax and Semantics.

A *signature* is a set of function and predicate symbols, each with an associated nonnegative integer *arity*. A (first-order) *structure*, $\mathcal{M}$, with signature, Σ, consists of a nonempty set called the *domain* of $\mathcal{M}$, and an *interpretation* mapping the symbols in Σ to total functions and predicates of corresponding arity on the domain. A *valuation* over $\mathcal{M}$ is a mapping from a set of (first-order) variable symbols to elements of the domain. In the case that array assignments occur in program schemes, the valuation also maps *array* variables to total functions of corresponding arity on the domain. (The first-order and array variable symbols are assumed to be disjoint from each other and from the signature Σ.)

Valuations serve to model the *state* of computer memory, with the value of each variable interpreted as the current value of a corresponding memory location or array. Assignment statements update the memory and thus are modelled by mappings on valuations. So any program scheme, α, and structure, $\mathcal{M}$ whose signature contains that of α, determines a binary *next-state relation*, $R_{\alpha\mathcal{M}}$ between valuations over $\mathcal{M}$ of the variables of α. That is, $sR_{\alpha\mathcal{M}}t$ means that under the interpretation given by $\mathcal{M}$, starting with valuation s, there is a (nondeterministic) execution of the successive assignment and test instructions in α which terminates with t. We omit the formal definitions (*cf.* [11], [19]).

For program schemes α, β, we write $\alpha \subset \beta$ to indicate that $R_{\alpha\mathcal{M}}$ is contained in $R_{\beta\mathcal{M}}$ for all $\mathcal{M}$ of suitable signature. Then α is *equivalent* to β iff $\alpha \subset \beta$ and $\beta \subset \alpha$.

Let F be a first-order formula with signature Σ and free variables V. Then any structure $\mathcal{M}$ whose signature contains Σ, together with any valuation over $\mathcal{M}$ of a set of variables containing V, determines a truth value for F in the usual way. The formula F is *valid in* $\mathcal{M}$, written $\mathcal{M}|\!=\!F$, iff F is true for all valuations over $\mathcal{M}$ of the free variables of F. Let *Valid*($\mathcal{M}$) be the set of F such that $\mathcal{M}|\!=\!F$. We say F is *valid*, written $|\!=\!F$, iff it is valid in all structures $\mathcal{M}$ of suitable signature.

For any (possibly infinite) set, T, of first-order formulas, let *Theory(T)* be the set of first-order formulas provable by the usual rules of predicate calculus with the formulas in T as additional axioms. For formulas T *without array variables*, the completeness theorem for predicate calculus implies that $F \in$ *Theory(T)* iff $\mathcal{M}|\!=\!F$ for all $\mathcal{M}$ such that

$\mathcal{M}|=\wedge T$. (With array variables present, the implication from right to left may trivially fail. For example, let a and b be one-dimensional (*i.e.*, unary) array variables, and let T consist of the single formula $a[0]=0$. Then T semantically implies $b[0]=0$ because a is implicitly universally quantified in T, but the formula $b[0]=0$ is not in *Theory(T)*.)

The *partial correctness assertion* $\{F\}\alpha\{G\}$ reads informally as, "If F holds at the start of execution of α, then G holds whenever α halts." More precisely, let $\mathcal{M}$ be a structure whose signature contains the signatures of F, G, and α, and let s be a valuation over $\mathcal{M}$ of the free variables of F, G, and α. Then $\mathcal{M},s|=\{F\}\alpha\{G\}$ iff ($s|=F$ implies $t|=G$ for all t such that $sR_{\alpha\mathcal{M}}t$). The partial correctness assertion P is *valid in* $\mathcal{M}$, written $\mathcal{M}|=P$, iff $\mathcal{M},s|=P$ for all valuations s over $\mathcal{M}$ of the free variables of P. Similarly, P is *valid*, written $|=P$, iff it is valid in all structures $\mathcal{M}$ whose signature contains that of P.

If T is a set of first-order formulas and P is a partial correctness assertion about a **while**-program scheme, then we write $T\vdash P$ to mean that P is provable from the axioms and rules of Floyd-Hoare logic using *Theory(T)* as first-order axioms. In particular, $\vdash\{F\}\alpha\{G\}$ means that the assertion is provable using only the valid formulas of predicate calculus as axioms. We shall not repeat the familiar axioms and rules of inference of the Floyd-Hoare system here, but we remind the reader that the axioms and the consequents of each of the rules are partial correctness assertions. There are no axioms or rules of inference for deducing first-order formulas. Indeed the only place where first-order formulas appear by themselves (viz., not as pre- or post-conditions within partial correctness assertions), is as antecedents in the

Rule of consequence (Hoare [13]):

$$\frac{F\Rightarrow F_1,\ \{F_1\}\alpha\{G_1\},\ G_1\Rightarrow G}{\{F\}\alpha\{G\}}$$

3. Semantics from Partial Correctness.

The partial correctness theory of an r.e. program scheme determines its semantics in the following precise sense:

Proposition 1. (*Separation*, Bergstra, Tiuryn, Tucker [3], Meyer, Halpern [18]) For any r.e. program schemes α, β such that $\alpha\not\subset\beta$, there are first-order formulas F, G such that

(i) $|=\{F\}\beta\{G\}$, *i.e.*, $\{F\}\beta\{G\}$ is valid, and
(ii) $\{F\}\alpha\{G\}$ is not valid.

Put another way, suppose $\alpha\not\subset\beta$. Then there is always a first-order partial correctness assertion about α and β which *witnesses* their inequivalence. That is, to the challenge, "What is the computational difference between α and β?" one can always give an answer, "If F is true before execution of β, then G is true afterward, but this is not

necessarily the case for α."

Our main result is that in the case that β is a **while**-program scheme, a valid witness assertion not only exists, but can be *proved* valid in Floyd-Hoare logic -- or any similar logic with the general properties described below. Namely, we have:

Theorem. (*Provable Separation*) For any r.e. program scheme α and **while**-program scheme β such that $\alpha \not\subseteq \beta$, there are first-order formulas F, G such that

(i) $\vdash\{F\}\beta\{G\}$, *i.e.*, $\{F\}\beta\{G\}$ is provable, and
(ii) $\{F\}\alpha\{G\}$ is not valid.

We remark that the Theorem holds even when a nondeterministic choice construct and/or random assignments are allowed in **while**-program schemes, assuming that the Floyd-Hoare system is extended with the usual axiom scheme for random assignments and a rule for choice constructs, *cf.* [11]. The schemes may also contain arbitrary first-order tests. However, the proof below does *not* directly apply when array assignments are allowed in **while**-program schemes for reasons indicated below. However, it does not seem that array asignments create fundamental difficulties, and we expect that our methods can be extended to handle array assignments as well.

To prove the Theorem, we recollect the following familiar facts.

Proposition 2. (*Relative Completeness*, Cook [6]) Let $\mathcal{M}$ be an *expressive* structure. Then Floyd-Hoare logic is complete relative to the theory of $\mathcal{M}$:

$$\mathcal{M}\models\{F\}\alpha\{G\} \text{ iff } \mathit{Valid}(\mathcal{M})\vdash\{F\}\alpha\{G\}.$$

The exact definition of expressive structure is not needed. It is sufficient to observe:

Proposition 3. (*Expansion*) Every structure can be *expanded* (*cf.* [21]) to an expressive structure.

Proof. Clarke [4] observes that finite structures are expressive. Harel ([11], p. 30) remarks that every structure can be *extended* to an *arithmetic universe*; it is easy to see that if a structure is infinite, it actually *expands* to an arithmetic universe. Also every arithmetic universe is expressive ([11], Theorem 3.2). □

Like the rules of predicate calculus, the Floyd-Hoare proof rules are finitary, so we have:

Proposition 4. (*Compactness*) If $T\vdash\{F\}\alpha\{G\}$, then $T_{fin}\vdash\{F\}\alpha\{G\}$ for some *finite* set $T_{fin}\subseteq T$.

Finally, the Floyd-Hoare system obeys the following familiar property:

Proposition 5. (*Deduction Theorem*) If $T\cup\{H\}\vdash\{F\}\alpha\{G\}$ for some set, T, of formulas and some first-order formula, H, without free first-order variables, then $T\vdash\{F\wedge H\}\alpha\{G\wedge H\}$.

Proof. From the deduction theorem for predicate calculus by induction on the length of the Floyd-Hoare proof. □

We have used the phrase "without free first-order variables" instead of "closed" because we cannot rule out the possibility that the formula H contains free array variables. We remark that the Deduction Theorem fails if assignments to array variables free in H can appear in α, which is why array assignments were disallowed.

We now prove our Theorem as follows.

Proof of separation by Floyd-Hoare logic. Suppose $\alpha \not\subset \beta$ for some r.e. scheme α and **while**-program scheme β. By Proposition 1, there are first-order formulas F, G such that $\{F\}\beta\{G\}$ is valid and $\{F\}\alpha\{G\}$ is not. Since this last assertion is not valid, there must be a structure $\mathcal{M}$ such that $\mathcal{M} \not\models \{F\}\alpha\{G\}$. It follows directly from the definitions that $\mathcal{M} \not\models \{F \wedge H)\}\alpha\{G \wedge H\}$ for any formula H such that $\mathcal{M} \models H$.

Since the interpretation of symbols not mentioned in a partial correctness assertion do not affect the truth of the assertion, $\mathcal{M}$ and its expansions satisfy the precisely same partial correctness assertions whose signatures are contained in that of $\mathcal{M}$. So, by Proposition 3, we could have chosen $\mathcal{M}$ to be expressive. Since $\{F\}\beta\{G\}$ is valid, we trivially have $\mathcal{M} \models \{F\}\beta\{G\}$, so by Proposition 2, $Valid(\mathcal{M}) \vdash \{F\}\beta\{G\}$. Now by Proposition 4, there is a finite subset $T_{fin} \subseteq Valid(\mathcal{M})$ such that $T_{fin} \vdash \{F\}\beta\{G\}$. Since formulas and their first-order universal closures have the same first-order theories, we may assume the formulas in T_{fin} do not have free first-order variables. By Proposition 5, letting H be the conjunction of the formulas in T_{fin}, we have $\vdash \{F \wedge H\}\beta\{G \wedge H\}$. Moreover, since $\mathcal{M} \models H$, we have $\mathcal{M} \not\models \{F \wedge H\}\alpha\{G \wedge H\}$. So $\{F \wedge H\}\beta\{G \wedge H\}$ is the desired provable witness to the inequivalence of α and β. □

Relative completeness theorems in the sense of Cook (Proposition 2) are known for several ALGOL-like extensions of **while**-program schemes such as blocks with local variables and procedures with call-by-value parameters [6], recursive procedures with simple procedure parameters and even with higher-order procedure parameters under suitable syntactic restrictions [1], [22], [23]. The argument above applies directly to establish the Provable Separation Theorem for these logics of ALGOL-like program schemes.

In fact, for various weaker kinds of relative completeness which have been considered in the literature, the proof also carries over by establishing suitable variations of the Expansion proposition. For example, several program logics for higher-order recursive procedures have been shown to be complete relative to expressive *Herbrand definable* interpretations [5], [8], [9]. But (i) every structure (with countable signature) is elementarily equivalent to a countable structure by the downward Skolem-Löwenheim Theorem, (ii) elementarily equivalent structures have the same first-order partial

correctness theory (*cf.* [20]), and (iii) any countable structure can straightforwardly be expanded to an expressive Herbrand definable structure. With these remarks, our argument now implies that Provable Separation holds for these additional program logics. The argument should also readily apply to logics which are merely complete relative to arithmetic universes [11] when the logics include enough rules to do some rudimentary reasoning about finite domains. In this way we expect an axiomatic semantics for *full* ALGOL-like program schemes without syntactic restrictions can be established, though we have not as yet worked out the details.

Leivant has indicated that his methods will generalize similarly, although the proof sketched in [14] appears to rest on properties special both to **while**-programs and Floyd-Hoare logic.

We remark that we cannot hope to construct an effective proof system for proving the *non*validity of witness assertions. This follows because α is divergent iff the partial correctness assertion $\{\mathbf{true}\}\alpha\{\mathbf{false}\}$ is valid; but the **while**-program schemes equivalent to the totally divergent scheme are not r.e. [12], [16].

4. Provable Separation by Universal Formulas.

Finally, we note that the form of the pre- and post-conditions in the assertions witnessing semantical separation can be very simple. This follows by a straightforward Skolemization argument.

> **Corollary.** The witness assertions for the Separation by Floyd-Hoare Logic Theorem can be chosen so that the pre-condition, F, is a universal formula and the post-condition, G, is quantifier-free.

Proof. Meyer and Halpern [18] show that the post-condition G in Proposition 1 can be quantifier-free. From the proof of Floyd-Hoare separation, we have F, H, and a structure $\mathcal{M}$ such that

(i) $\vdash\{F\wedge H\}\beta\{G\wedge H\}$,
(ii) $\mathcal{M}\models H$, and
(iii) $\mathcal{M}\not\models\{F\}\alpha\{G\}$.

By the rule of consequence, we have from (i) that $\vdash\{F\wedge H\}\beta\{G\}$. Also, (ii) and (iii) imply that $\mathcal{M}\not\models\{F\wedge H\}\alpha\{G\}$. Now let U be the universal formula obtained by Skolemizing $(F\wedge H)$. Then we claim $\{U\}\beta\{G\}$ is the desired witness.

To see this, observe first that $\vdash\{U\}\beta\{G\}$ by the rule of consequence because $U\Rightarrow(F\wedge H)$ is valid. Second, since any structure satisfying $(F\wedge H)$ expands to one satisfying U, we can choose any expansion, $\mathcal{N}$, of $\mathcal{M}$ satisfying U and be sure that $\mathcal{N}\not\models\{U\}\alpha\{G\}$. □

These observations were suggested by similar ones in [14] which they simplify and

strengthen. Note that this form of pre- and post-conditions is about as simple as possible, since entirely quantifier-free pre- and post-conditions are not sufficient for separation [18].

Acknowledgement. Thanks to Ed Clarke for comments and to Daniel Leivant for relaying and explaining his results.

5. References.

1. Apt, K.R. Ten years of Hoare's logic: a survey, Part I. *ACM Trans. on Prog. Lang. and Systems* **3** (1981), 431-483.

2. Bergstra, J.A. and J.W. Klop. Proving program inclusion using Hoare's logic. *Theoretical Computer Science* **30** (1984), 1-48.

3. Bergstra, J., J. Tiuryn, and J. Tucker. Floyd's principle, correctness theories and program equivalence. *Theoretical Computer Science* **17** (1982), 113-149.

4. Clarke, E.M., Jr. Programming language constructs for which it is impossible to obtain good Hoare axiom systems. *J. ACM* **26** (1979), 129-147.

5. Clarke, E.M., Jr., S.M. German, and J.Y. Halpern. On effective axiomatizations of Hoare logics. *J. ACM* **30** (1983), 612-636.

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

7. Floyd, R.W. Assigning meaning to programs. In *Proc. Symp. in Applied Mathematics: Mathematical Aspects of Computer Science*, Schwartz, J.T., Ed., 1967, 19-32.

8. German, S., E.M. Clarke, Jr., and J. Halpern. Reasoning about procedures as parameters. In *Logic of Programs, Proceedings 1983, Lect. Notes in Comp. Sci.* **164**, Clarke, Edmund M.,Jr. and Dexter Kozen, Eds., Springer-Verlag, 1984, 206-220.

9. Goerdt, A. A Hoare calculus for functions defined by recursion on higher types. In *Logics of Programs, Brooklyn, June 1985, Lect. Notes in Comp. Sci.* **193**, Parikh, R., Ed., Springer-Verlag, 1985, 106-117.

10. Greif, I. and A.R. Meyer. Specifying the semantics of **while**-programs. *ACM Trans. on Programming Lang. and Systems* **3** (1981), 484-507.

11. Harel, D. First-Order Dynamic Logic. *Lect. Notes in Comp. Sci.* **68**, Springer-Verlag, 1979.

12. Harel, D., A.R. Meyer and V. Pratt. Computability and completeness in logics of programs: preliminary report. In 9^{th} *ACM Symposium on Theory of Computing*, 1977, 261-268. Revised version, M.I.T. Lab. for Computer Science TM-97, (Feb. 1978)

16 pages.

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

14. Leivant, Daniel. Hoare's logic captures program semantics. Extended summary. 1985, Unpublished report, Dept. of Computer Science, Carnegie-Mellon Univ.

15. Leivant, Daniel. Logical and imperative reasoning about imperative programs. In *12th ACM Symp. on Principles of Programming Languages*, 1985, 132-140.

16. Luckham, D.C., D.M. Park and M.S. Paterson. On formalized computer programs. *J. Computer and System Sciences* **4** (1970), 220-249.

17. Makowsky, J.A. and I. Sain. On the equivalence of weak second order and nonstandard time semantics for various program verification systems. In *IEEE Symp. Logic in Computer Science*, Meyer, A., Ed., 1986, 00-00. To appear.

18. Meyer, A.R. and J.Y. Halpern. Axiomatic definitions of programming languages: a theoretical assessment. *J. ACM* **29** (1982), 555-576.

19. Meyer, A.R., and R. Parikh. Definability in dynamic logic. *J. Computer and System Sciences* **23** (1981), 279-298.

20. Meyer, A.R. and J. Tiuryn. Equivalences Among Logics of Programs. *J. Computer and System Sciences* **29** (1984), 160-169.

21. Monk, J. Mathematical Logic. *Graduate Texts in Mathematics* **37**, Springer-Verlag, 1976.

22. Olderog, E. Sound and complete Hoare-like calculi based on copy rules. *Acta Informatica* **16** (1981), 161-197.

23. Olderog, E. R. Hoare's logic for program with procedures -- what has been accomplished? In *Logic of Programs, Proceedings 1983, Lect. Notes in Comp. Sci.* **164**, Clarke, Edmund and Dexter Kozen, Eds., Springer-Verlag, 1984, 383-395.