SPECIFYING PROGRAMMING LANGUAGE SEMANTICS:

A Tutorial and Critique of a Paper by Hoare and Lauer

by

I. Greif and A. Meyer

Massachusetts Institute of Technology
Cambridge, Massachusetts

1. Introduction

Hoare and Lauer [1974] have advocated using a variety of styles of programming language definitions to fit the variety of users from implementers to program verifiers. They consider the question of whether different definitions and specifications determine the same language by showing that the definitions are what they call "consistent". However, their treatment skirts the question of whether their definitions can each be taken to specify the language adequately.[1] Although, as we will show, any one of the kinds of semantics they discuss -- operational, relational, deductive -- can be used to specify meaning uniquely, Hoare and Lauer do not make the case in their paper. In fact, both their relational and deductive definitions are satisfied by several different semantics, only one of which is desired.

Thus, the main point of this paper is to clarify the characteristics of a proper specification of language semantics and to formulate alternative specifications each of which is equally good as the language definition. We basically agree with Hoare and Lauer that several specifications can and should be given, but are disturbed by confusions about such specifications, some of which are illustrated in their paper. In particular we refer to confusions between the mathematical object which is designated to be the meaning of a program and methods for specifying that object; the similar confusion between predicate and expression; between consistency and equivalence of two definitions; between completeness of a theory and its having a unique model. While these issues are familiar in mathematical logic, we take this opportunity to survey them in the context of programming language semantics.

This paper can be read without prior familiarity with Hoare and Lauer's paper. The authors plan another paper extending this work which will include a more comprehensive bibliography.

2. The Programming Language

Following Hoare and Lauer, we will examine alternative types of definitions of a trivial language with primitive statements, *while* statements, and statement lists. The syntax, omitting details of the form of predicate expressions is as follows:

<program> ::= <primitive statement> | <while statement> | <program>;<program> | *NOP*

<while statement> ::= *while* <predicate expression> *do* <program>

As is usual with abstract syntax, we will not concern ourselves with ambiguity in parsing or with detailed syntax of expressions and primitive statements.

We assume that programs run on machines with states. We treat the states simply as abstract elements in some fixed set S, ignoring their internal structure In many familiar examples primitive statements define total functions from states to states, but we need not make this assumption. Primitive statements may be *partial*, i.e. for some state s there may be no related state, and *nonfunctional*[2], i.e for some states s there may be more than one related state. A primitive statement, A, thus has an effect on states which can be defined by giving a relation $R_A \subset S \times S$ such that $(s, s') \in R_A$ iff A executed in s can terminate in state s'. For example, if states associate values with variables, and primitive statements are assignments "$u := e$" where u is a variable and e is an expression which can be evaluated in any state, then an assignment statement relates certain states which differ only at the value of the variable u.

A predicate P is a mapping from states to truth values. Predicate expressions p, q,.. appear in programs. We will use P, Q,.., respectively, to denote the predicates corresponding to these expressions. For simplicity, we assume that predicate expressions always yield values, so that the predicate P associated with an expression p is true or false at each state and is never undefined.

Hoare and Lauer give two "operational" definitions of this simple language The first is an abstract machine which can execute program steps The second is a function which maps programs into their computations, where a computation is a finite sequence of states, namely the successive states which are reached during execution of the program. These two definitions are said to be "consistent" in that for any program a both define the same relation R_a of initial to final states. This relation $R_a \subset S \times S$ is such that $(s, s') \in R_a$ if and only if, when started in state s, program a halts (it has a finite computation) and the final state of that computation is s'.

We use the following notation throughout what follows:

a, b, c	programs,
A	primitive statements,
s, t	states (elements of the set S of all states),
p, q	predicate expressions,
P, Q	predicates on states,
L, M, R	binary relations on states (subsets of S×S).

Each of these letters may appear with subscripts or multiply primed, e.g., s_2, s', Q'', etc.

3. Relational Semantics

A relational semantics assigns to each program a an "initial-state, final-state" relation. We can express the proper relational semantics for our language directly by defining the relations induced by programs as in [deBakker, 1975; Pratt, 1976; deBakker and Meertens, 1975]. Hoare and Lauer give an axiom system for triples $s(a)s'$ such that (s, s') is in the relation induced by a. We present both a complete axiom system and a deductive system for these triples.

3.1 The Standard Relational Semantics

A simple definition of the relation R_a to be associated with any program a can be given by induction on the syntax of programs, using only familiar mathematical operations on relations. In order to do this it is convenient to define R_p for any predicate expression p to be $\{(s, s) \mid P(s)\}$. For $R_1, R_2 \subset S\times S$, let R_1^* be the reflexive transitive closure of R_1, and $R_1 \circ R_2$ the composition of R_1 and R_2. We assume that relations R_A for each primitive statement A are given. That is, we assume that we know what the primitive statements mean. Let I be the identity relation $\{(s, s) \mid s \in S\}$. Then the relations associated with programs are defined as follows:

R1. $R_{NOP} = I$,

R2. $R_{a;b} = R_a \circ R_b$,

R3. $R_{\textit{while}\ p\ \textit{do}\ a} = (R_p \circ R_a)^* \circ R_{\neg p}$.

These relations describe the standard semantics for our language. To see this, note that NOP does not change the state and that the program $a;b$ started in state s will end in state s' iff there is a state t such that a started in s ends in state t and b started in t ends in s'. Similarly, for *while* loops there must be a sequence of states between initial state, s, and final state, s', such that in each state but the last P is true and each pair of successive states is related by R_a. In the final state $\neg P$ must be true. If $\neg P(s)$ is true for some state s, then *while* p *do* a acts like a NOP, that is, $(s, s) \in R_{\textit{while}\ p\ \textit{do}\ a}$.

This relational semantics obviously gives exactly the same meaning to programs as do the interpretive and computational semantics of Hoare and Lauer. We shall henceforth refer to this as the standard relational semantics **R**.

3.2 Some Axioms for the Standard Relational Semantics

Hoare and Lauer choose to specify the standard relational semantics by giving a system of axioms for statements of the form "started in s, program a terminates in state s'." We shall refer to such assertions as "transition assertions", and follow Hoare and Lauer in using the notation $s(a)s'$ to denote such a statement. Thus $s(a)s'$ is true for **M** iff $(s, s') \in M_a$, where **M** is an arbitrary relational semantics which assigns to each program a some relation $M_a \subset S\times S$.

Their axioms[3] are as follows:

HL1. $s(A)s' \leftrightarrow (s, s') \in R_A$,

HL2. $s(a;b)s' \leftrightarrow \exists t[s(a)t \wedge t(b)s']$,

HL3. $s(\textit{while}\ p\ \textit{do}\ a)s' \rightarrow \neg P(s')$,

HL4. $\forall s_1, s_2[(Q(s_1) \wedge P(s_1) \wedge s_1(a)s_2) \rightarrow Q(s_2)] \rightarrow$
$[(Q(s) \wedge s(\textit{while}\ p\ \textit{do}\ a)s') \rightarrow Q(s')]$,

HL5. $s(NOP)s' \leftrightarrow s = s'$.

They go on to prove that the standard relational semantics **R** is a model of HL1-5, that is, every instance of HL1-5 is true for **R**, so that any conclusion which logically follows from these axioms will be true of the standard semantics. Of course this meets only half the requirements for specifying the semantics, since one must also show that any transition assertion which is true of the standard semantics follows logically from the axioms. Unfortunately HL1-5 do not imply all the true assertions, contrary to the "intuitive confidence in the completeness of the theory" expressed by Hoare and Lauer (cf. [Hoare and Lauer] p. 144), as we now illustrate.

We can understand the significance of HL1-5 as follows. If **M** is a model of Hl, we can conclude that $M_A = R_A$ for each atomic statement A. Similarly, from HL5 we conclude that $M_{NOP} = I = R_{NOP}$, and from HL2 that $M_{a;b} = M_a \circ M_b$. It follows that $M_a = R_a$ for every *while*-free program a whenever **M** is a model of HL1, 2, 5.

Now consider the particular "divergent loop" relational semantics $\mathbf{L}$ defined as follows:

$$L_a = R_a \text{ if } a \text{ is } \textit{while}\text{-free},$$

$$L_a = \phi \text{ otherwise.}$$

Then $\mathbf{L}$ is obviously a model of HL1, 2, 5. But $s(\textit{while}\ \mathrm{p}\ \textit{do}\ a)s'$ is always false for $\mathbf{L}$, so HL3-4 are true, vacuously, for $\mathbf{L}$. Hence $\mathbf{L}$ is also a model of HL1-5.

The divergent loop semantics corresponds to an implementation in which the interpreter simply loops unconditionally whenever it starts to execute a *while* statement. Since $\mathbf{L}$ is a model, statements which logically follow from HL1-5 must always be true of this implementation. In particular, no transition assertion involving a program containing a *while*-loop follows from HL1-5, and so it seems hard to imagine circumstances in which HL1-5 would serve as an adequate characterization of the standard semantics. (However, in section 4.4 we shall indicate a natural sense in which HL1-5 do in fact specify $\mathbf{R}$.)

3.3 A Complete Set of Axioms for the Standard Relational Semantics

There is no inherent obstacle to presenting axioms in the spirit of HL1-5 which correctly and completely specify the intended semantics. Indeed, adding two more axioms will suffice:

HL6. $\neg P(s) \rightarrow s(\textit{while}\ \mathrm{p}\ \textit{do}\ a)s$,

HL7. $P(s) \wedge s(a)s' \wedge s'(\textit{while}\ \mathrm{p}\ \textit{do}\ a)t \rightarrow s(\textit{while}\ \mathrm{p}\ \textit{do}\ a)t$.

It is easy to verify that the standard semantics is a model of HL1-7. In the appendix we prove:

Theorem 1: The standard relational semantics is the only model of HL1-7.

We remark that HL1-7 can be shown to be independent, i.e., Theorem 1 is not true when any one of HL1-7 is omitted.

3.4 A Deductive System for the Standard Relational Semantics

Another, perhaps more straightforward, way to specify the standard relational semantics is to give a system of axioms and inference rules for deducing transition statements. One such system is:

Axioms:

T1. $s(A)s'$, for all $s, s' \in S$ such that $(s, s') \in R_A$,

T2. $s(NOP)s$,

T3. $s(\textit{while}\ \mathrm{p}\ \textit{do}\ a)s$, for all $s \in S$ such that $\neg P(s)$.

Inference Rules:

T4. $s(a)t,\ t(b)s' \vdash s(a;b)s'$,

T5 $s(a)t,\ t(\textit{while}\ \mathrm{p}\ \textit{do}\ a)s' \vdash s(\textit{while}\ \mathrm{p}\ \textit{do}\ a)s'$, for all $s \in S$ such that $P(s)$.

Let Th(T1-5) be the set of transition statements provable from T1-3 using T4-5. A routine proof, the details of which we omit, implies that $s(a)s' \in$ Th(T1-5) if and only if $(s, s') \in R_a$. That is,

Theorem 2: The set of transition assertions derivable in the system T1-5 is equal to the set of transition assertions true for the standard relational semantics.

Thus, the deductive system T1-5 specifies the same relational semantics as R1-3, and either can serve as the definitive specification. (We caution the reader not to confuse this deductive *specification* of a relational semantics with the deductive *semantics* of Hoare and Lauer mentioned in section 4.5 of this paper.)

4 Partial-Correctness Semantics

Assertions of the form "if P holds before executing a, then if and when a halts, Q will hold" occur frequently when the behavior of programs is being described. Such assertions are called *partial correctness assertions* (pca's) and are abbreviated $P\{a\}Q$.

We define a *partial correctness semantics* for our programming language to be an arbitrary set of pca's. Any relational semantics $\mathbf{M}$ naturally determines a corresponding partial correctness semantics $\overline{\mathbf{M}}$ consisting of those pca's $P\{a\}Q$ which are true when the initial-state, final-state relation of a is that given by $\mathbf{M}$.

The thesis that a programming language semantics could be specified by giving all the "before-after" assertions true of programs has been espoused by Dijkstra [1975, 1976]. An effort by Hoare and Wirth [1973] to specify the semantics of a fragment of PASCAL using pca's supports the practical applicability of this thesis. Our desire to investigate this general thesis motivates our definition and analysis of partial correctness semantics in this section.

We show how a relational semantics can determine a partial correctness semantic and vice versa. We give a complete deductive system for pca's and an axiom system for pca's. The significance of specifications which have many relational models is considered, and we analyze several such specifications.

4.1 The Standard Partial Correctness Semantics

Definition 1: Let R be a binary relation on states. $P\{a\}Q$ is *true for the relation* R iff $(\forall s,s')\,(P(s) \wedge (s, s') \in R \rightarrow Q(s'))$. $P\{a\}Q$ is *true for a relational semantics* $\mathbf{M}$ iff it is true for M_a.

The partial correctness semantics containing exactly the pca's which are true for $\mathbf{R}$ is referred to as the *standard partial correctness semantics*, $\mathfrak{R}$.

An arbitrary set of partial correctness assertions for a program a also determines a relation. The relation is the maximum relation, M, such that all the pca's in the set are true for M. (That there always is such a maximum relation is shown in the appendix, Lemma C1.) The rationale for taking this relation to be the one determined by a partial correctness semantics is nicely expressed by Schwarz [1974]:

> "Asserting a partial correctness statement is essentially asserting that certain environments are not the results of executing some command starting in certain other environments. This is a negative requirement, it does not force any environment to be the result of any execution. Since this is the inherent nature of the formalism it indicates that the proper kind of definition of the semantics determined by a system should have the form: 'largest possible semantics.' "

Definition 2: Let $\mathfrak{M}$ be an arbitrary set of pca's and for any program a, let $\mathfrak{M}_a$ be the set of all pca's in $\mathfrak{M}$ of the form $P\{a\}Q$. Then $max(\mathfrak{M}_a)$ is the maximum relation M such that all pca's in $\mathfrak{M}_a$ are true for M.

We prove in the appendix that taking the maximum relations determined by any partial correctness semantics provides a way to recover an underlying relational semantics if there is one. Formally we have

Lemma 1: Let $\mathfrak{M}$ be the set of pca's true for a relational semantics $\mathbf{M}$. Then $M_a = max(\mathfrak{M}_a)$. In particular, $R_a = max(\mathfrak{R}_a)$.

The significance of Definition 2 and Lemma 1 is that $\mathfrak{R}$ and $\mathbf{R}$ convey exactly the same information -- either one uniquely determines the other.[4] This implies that, if we prefer, we can choose a partial correctness semantics to specify meaning. Such a partial correctness semantics sacrifices nothing provided by a relational semantics, since any desired relational semantics, $\mathbf{M}$, can always be recovered from an appropriate partial correctness semantics, namely, the one which consists of the partial correctness assertions true for $\mathbf{M}$.

4.2 Deducing Partial Correctness

The standard partial correctness semantics can, like the standard relational semantics, be specified by a simple system of axioms and inference rules. The notion of the *weakest antecedent*, $[R]Q$, of a predicate Q under a relation R is used in the axioms for primitive instructions. Informally, $[R]Q$ is the predicate on states which is true of a state s providing that: if and when a program with initial-state, final-state relation R halts after being started in s, the predicate Q will hold.

Definition 3: Let R be a binary relation on states. For any predicate Q on states, the *weakest antecedent* of Q under R is a predicate, $[R]Q$ on states defined by

$$([R]Q)(s) \text{ iff } (\forall s')[(s, s') \in R \rightarrow Q(s')].$$

It follows immediately from Definitions 1 and 3 that $([M_a]Q)\{a\}Q$ is true for any relational semantics $\mathbf{M}$. We abbreviate "$\forall s(P(s) \rightarrow Q(s))$" by "$\models(P \rightarrow Q)$" and note that $P\{a\}Q$ is true for $\mathbf{R}$ iff $\models(P \rightarrow [R_a]Q)$, which is why $[M_a]Q$ is called "weakest". (cf. [Pratt, 1976; Harel, Meyer, Pratt, 1975; Schwarz, 1974]).

The following system is usually referred to as the Floyd-Hoare system for partial correctness.

Axioms:

FH1. $P\{NOP\}P$,

FH2. $([R_A]Q)\{A\}Q$,

Inference Rules:

FH3. $P\{a\}P',\ P'\{b\}Q \vdash P\{a; b\}Q$,

FH4. $(P \wedge Q)\{a\}Q \vdash Q\{\textit{while } p \textit{ do } a\}(Q \wedge \neg P)$,

FH5. $P\{a\}Q \vdash (P \wedge P')\{a\}(Q \vee Q')$

We prove in the appendix,

Theorem 3: The set of pca's derivable from FH1-5, is equal to $\mathfrak{R}$, the standard partial correctness semantics.

We have formulated Theorem 3 to emphasize our view of the system FH1-5 as a specification of a mathematical object, namely the set Th(FH1-5) of derivable pca's. The more familiar viewpoint in the literature would be to presume that truth of pca's was always to be reckoned relative to the standard relational semantics. Theorem 3 could then be formulated as saying that FH1-5 is *sound* -- only true pca's can be derived -- and *complete* -- all the true pca's can be derived.

The system FH2-4 corresponds to the Deductive Theory[5] DI-3 of Hoare and Lauer, p. 146. The system FH1-4 is *not* complete, but we will see in section 4.5 that there is a sense in which the imcomplete system FH1-4 specifies the standard semantics.

Although a deductive system resembling FHI-5 is the more usual specification of the standard partial correctness semantics, we can also write an axiom system to specify it. The axioms are suggested straightforwardly by the deductive system.

PCI. $[P\{NOP\}Q] \leftrightarrow \models(P \rightarrow Q)$,

PC2. $[P\{A\}Q] \leftrightarrow \models(P \rightarrow [R_A]Q)$,

PC3. $[P\{a;b\}Q] \leftrightarrow \exists P'(P\{a\}P' \wedge P'\{b\}Q)$,

PC4. $[Q\{\textit{while } p \textit{ do } a\}Q'] \leftrightarrow \exists Q''[\,[(P \wedge Q'')\{a\}Q''] \wedge \models(Q \rightarrow Q'') \wedge \models((Q'' \wedge \neg P) \rightarrow Q')\,]$.

To say how these axioms specify the partial correctness semantics we must give a technical meaning to the term *model* and distinguish several special kinds of models.

A mathematical object is said to be a *model* for a set of assertions if all the assertions are true for the object. We have already used this notion in section 3 where the objects were relational semantics and the assertions were transition assertions. We will be using different kinds of objects, for example, both relations and sets of pca's, as models of sets of assertions. The assertions themselves may simply be pca's, or they may be more complicated kinds of mathematical assertions such as PCI-4. (We will not need and therefore omit a more precise explanation of what "mathematical assertions" are than is provided by the example of PCI-4.) The following definition establishes how a partial correctness semantics can serve as a model.

Definition 4: A partial correctness semantics Π satisfies a mathematical assertion (in which pca's may appear as subassertions) iff the assertion's value can be calculated to be *true* when precisely the pca's in Π are assigned the value *true*. A partial correctness semantics is a *model* for a set of assertions if it satisfies every assertion in the set. A partial correctness semantics is a *partial correctness model* for an axiom system (such as PCI-4) if it is a model for the set of all instances of those axioms.

Theorem 4(deBakker[6]): $\mathbf{R}$ is the only partial correctness model of PCI-4.

The proof is in the appendix.

Again, we have formulated this theorem to emphasize our view of PCI-4 as uniquely specifying a particular partial correctness semantics. We consider next the more usual view of PCI-4 as specifying a relational semantics.

We have just considered FHI-5 and PCI-4 as direct specifications of partial correctness semantics. However, since relational semantics determine truth values for pca's by Definition 1, we can regard a relational semantics as a possible model of a set of pca's or similar mathematical assertions. Therefore we can also consider FHI-5 and PCI-4 as specifications of relational semantics according to their relational models. Thus we can rephrase Theorems 2, 3 and 4 in part by saying that $\mathbf{R}$ is a model of TI-5, FHI-5 and PCI-4.[7]

Notice that despite Theorems 2 and 3, we cannot say that $\mathbf{R}$ is the only model of TI-5 or FHI-5. For example, the "empty" semantics which assigns the empty relation to every program is a model of FHI-5, and the semantics which assigns the "total" relation S×S to every program is a model of TI-5.

A set of pca's will generally fail to have a unique relational model because pca's are "anti-monotone" in the following sense. If $\mathbf{M}$ and $\mathbf{N}$ are relational semantics then we shall say that $\mathbf{N}$ is larger than $\mathbf{M}$ iff $N_a \supset M_a$ for all programs a. Then by Definition 1 we see that if $P\{a\}Q$ is true for $\mathbf{N}$, and $\mathbf{N}$ is larger than $\mathbf{M}$, then $P\{a\}Q$ is also true for $\mathbf{M}$. Thus, since $\mathbf{R}$ is a model of FHI-5, so is any relational semantics smaller than $\mathbf{R}$.[8]

On the other hand, Theorem 3 and Lemma 1 together imply that $\mathbf{R}$ is larger than any model of FHI-5, so we can conclude

Theorem 5: The standard relational semantics is the largest model of FHI-5.

Similarly, transition assertions are "monotone" in the sense that if $s(a)s'$ is true for $\mathbf{M}$, and $\mathbf{N}$ is larger than $\mathbf{M}$, then $s(a)s'$ is true for $\mathbf{N}$. We conclude from Theorem 2 that

Theorem 6: The standard relational semantics is the smallest model of TI-5.

Finally, we can deduce from Theorem 4 that

Theorem 7: The standard relational semantics is the only model of PCI-4

Thus, Theorems 5, 6, and 7 reveal precisely the different ways in which $\mathbf{R}$ is determined uniquely by the specifications FHI-5, TI-5, PCI-4.

We should point out that Theorem 7 is technically a slightly weaker result than Theorem 4. Theorem 7 in effect asserts that among the partial correctness semantics which are determined by relational semantics, only the partial correctness semantics Π determined by $\mathbf{R}$ is a model of PCI-4. On the other hand, Theorem 4 asserts that among *all* partial correctness semantics, not just those determined by relations, Π is the unique model.

4.5 Implications Between Incomplete Semantical Specifications

The need to deal with specifications having several models of a given kind was allowed for by Hoare and Lauer in their formulation of what they call "consistency" between semantical specifications. They say that one specification is consistent with another iff every model of the latter is a model of the former.

Notice that this definition is asymmetrical, and so conflicts with ordinary usage of the word "consistency." For this reason, we shall refer to "implication" between specifications, that is, specification S implies specification T iff every model of S is a model of T.

Semantical specifications with more than one model can be useful. We have just seen that while FHI-5 and TI-5 technically speaking have many models, nevertheless they uniquely specify $\mathbf{R}$ in a natural way as the largest relational model and smallest relational model, respectively. Even more generally there may be situations in which any of several models would suffice for some application, and we wish only to specify this set of appropriate models -- not necessarily distinguishing a canonical model in the set by some criterion such as maximality or minimality. For example, in the specification of practical programming languages it is typical to leave undefined the meaning of certain syntactically well-formed programs. In such cases there will be many acceptable semantics differing only on the meanings, e.g., error messages, they assign to "meaningless" programs.

In addition to the axioms HLI-5 considered above, Hoare and Lauer offer the first four rules FHI-4 of the Floyd-Hoare system (cf. footnote 5) as a specification with multiple relational models, and they seem to suggest that these multiple models represent possible acceptable semantics. However, when we look more closely at the Hoare-Lauer and Floyd-Hoare axioms we shall see that an example of a specification which could be met by many acceptable semantics does not arise here; there is only one intended model of these particular specifications, although it takes some effort to discover the sense in which these specifications determine that model.

In particular, Hoare and Lauer observe [Hoare and Lauer, Theorem 4] that HLI-5 implies the first four Floyd-Hoare rules FHI-4.[9] For some reason they do not consider the converse question of whether FHI-4 implies HLI-5. In fact, it does not; not even the full Floyd-Hoare system FHI-5 implies HLI-5. This is because any $\mathbf{M}$ smaller than $\mathbf{R}$ is a relational model of FHI-5, so that, for example, the empty semantics is a model of FHI-5 but not of HLI-5.

However, Hoare and Lauer's proof that HLI-5 implies FHI-4 actually establishes a slightly stronger result which we can use to reveal the connections between HLI-5, FHI-4, and $\mathbf{R}$.

An inference rule such as any of FH3-5, T4-5 will be called *sound* for a relational semantics $\mathbf{M}$, if, whenever the conditions (such as those for T5) for applicability of the rule are satisfied and the antecedent(s) of the rule is true for $\mathbf{M}$, so is the consequent. In other words, an inference rule is sound if application of it preserves truth.

Lemma 2: If $\mathbf{M}$ is a model of FHI-2 and the inference rules FH3-4 are sound for $\mathbf{M}$, then $\mathbf{M}$ is a model of FHI-5.

Proof: It is easy to see that FH5 is sound for all $\mathbf{M}$.∎

Theorem 8: $\mathbf{R}$ is the largest model of FHI-2 for which the inference rules FH3-4 are sound.

Proof: We let the reader convince himself that FH3-4 are sound for the standard relational semantics $\mathbf{R}$ (cf. [Hoare and Lauer], Theorem 4). Thus, $\mathbf{R}$ is "a" model; that it is "the largest" model is immediate from Theorem 5 and Lemma 2.∎

Lemma 3(Hoare and Lauer): Let $\mathbf{M}$ be a model of HLI-5. Then $\mathbf{M}$ is a model of FHI-2 and the inference rules FH3-4 are sound for $\mathbf{M}$.

We shall not repeat the proof (cf. [Hoare and Lauer], page 147).

Theorem 9: $\mathbf{R}$ is the largest model of HLI-5.

Proof: Immediate from Theorem 7 and Lemma 3.∎

The preceding theorems thus reveal the sense in which HLI-5 and the first four Floyd-Hoare rules FHI-4 serve as semantical specifications equivalent to the others we have considered -- a rather obscure sense which was left implicit in [Hoare and Lauer].[10]

Our point here is that while we agree with Hoare and Lauer that relationships like implications between specifications with multiple models are important ideas, it is even more important to have a clear understanding of the family of models which are to be regarded as meeting the specifications. This is illustrated by the fact that the semantics $\mathbf{L}$ of section 3.2 is a relational model both of HLI-5 and FHI-5, yet we certainly do not mean to accept an implementation of our language in which all *while*-loops diverge.

5. Conclusion

We have looked at two kinds of semantics -- relational and partial correctness -- and several means of specifying a semantics -- inductive definitions, axiom systems, deduction systems. Each semantics can be specified in several ways. There was no particular technical problem in rigorously defining how specifications determined semantics.

The set of all partial correctness assertions true for our trivial programming language gives exactly the same information as the relational semantics; a specification which determines the pca semantics also determines the relational semantics. This is true despite the fact that in a certain narrow technical sense partial correctness assertions cannot be used to express termination of programs. Either kind of semantics can be specified using an axiom system or a deductive system; either semantics determines the other, independent of means of specification.

The results of this paper can be extended to other semantics and other means of specification, for example, predicate transformer semantics. For deterministic programs, the results are similar to those for pca's. The situation is more complicated for non-deterministic programs (cf. [Hoare, 1978; Harel and Pratt, 1978]).

Syntax played a secondary role in this paper. Only programs were syntactic objects; predicates were treated as mathematical, set-theoretic objects. The next refinement of the study begun here involves restricting predicates to those which are definable in some agreed-upon formal notation, e.g., first or second order logics of appropriate structures. When we restrict predicates in this way the situation becomes more complicated - and more interesting - and the conclusions we reached above about the equivalence of various kinds of semantics must be refined. Thus, there are cases where the set of all true definable pca's may not determine the proper relational semantics; in other cases a restricted deductive theory may contain only a subset of all true definable pca's and yet determine the right semantics. We postpone to a later paper further discussion of predicate transformer semantics and the restriction to definable predicates.

In sum, we have illustrated that attempting to specify the meaning of a language in several ways can be made to work -- at least for very simple programming languages when we place no restrictions on the language for predicates. However, care had to be taken to indicate how each specification was to be understood before it could be applied by any of the variety of possible users.

6. References

[The reference section of this paper does not constitute a complete bibliography on the subject.]

deBakker, J.W. 1975. Flow of Control in the Proof Theory of Structured Programming. *16th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society. Long Beach, Ca. pp 29-33.

deBakker, J.W. 1977. Recursive Programs as Predicate Transformers. Proc. IFIP Conf. on Formal Specifications of Programming Constructs. St. Andrews, Canada. pp 7.1-7.15.

deBakker, J.W. and L.G.L.T. Meertens. 1975. On the Completeness of the Inductive Assertion Method. *Journal of Computer and Systems Science*, 11, pp 323-357.

Dijkstra, E.W.D. 1975. Guarded Commands, Non-determinacy and Formal Derivation of Programs. *CACM* 18, 8. pp 453-457.

Dijkstra, E.W.D. 1976. *A Discipline of Programming*, Prentice-Hall, Englewood Cliffs, N.J., 217 pp.

Harel, D. 1978. Logics of Programs: Axiomatics and Descriptive Power. Laboratory for Computer Science Technical Report 200, M.I.T., Cambridge, Mass., 152 pp.

Harel, D., A.R. Meyer, and V. Pratt. 1977. Computability and Completeness in Logics of Programs. *Proc. of 9th Annual ACM Symposium on Theory of Computing*. Boulder, Colorado. pp 261-268.

Harel, D. and V. Pratt. 1978. Nondeterminism in Logics of Programs. *Conference Record of the Fifth Annual Symposium on Principles of Programming Languages*. Tuscon, Arizona. pp 203-213.

Hoare, C.A.R. 1978. Some Properties of Predicate Transformers *JACM* 25, 3 pp 461-480.

Hoare, C.A.R. 1969. An Axiomatic Basis for Computer Programming. *CACM* 12, 10. pp 576-583.

Hoare, C.A.R. and P. Lauer. 1974. Consistent and Complementary Formal Theories of the Semantics of Programming Languages. *Acta Informatica* 3, pp 135-155.

Hoare, C.A.R. and N. Wirth. 1973. An Axiomatic Definition of the Programming Language PASCAL. *Acta Informatica* 2, pp 335-355.

Manna, Z. 1974. *Mathematical Theory of Computation*, McGraw-Hill, New York, 429 pp.

Pratt, V. 1976. Semantical Considerations on Floyd-Hoare Logic *17th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society. Long Beach, Ca. pp 109-121.

Schwarz, J.S. 1974. Semantics of Partial Correctness Formalisms. Ph.D. Thesis, Syracuse University. Syracuse, N.Y. 126 pp.

7. Appendix

7.1 Proof of Theorem 1.

Theorem 1: The standard relational semantics is the only model of HL1-7.

Let T be the set of transitive assertions $s\langle a\rangle s'$ true for a relational model $\mathbf{T}$ of HL1-7. We will prove by induction on a that $(s, s') \in R_a$ iff $s\langle a\rangle s' \in T$. Thus $\mathbf{T} = \mathbf{R}$.

If a is NOP then by HL5, $s\langle NOP\rangle s' \in T \leftrightarrow s = s' \leftrightarrow (s, s') \in R_{NOP}$. Similarly if a is a primitive statement, A, then by HL1, $s\langle A\rangle s' \in T \leftrightarrow (s, s') \in R_A$.

If a is $b;c$ then by HL2, induction, and R2,

$$\begin{aligned} s\langle b,c\rangle s' \in T &\leftrightarrow \exists t[s\langle b\rangle t \in T \wedge t\langle c\rangle s' \in T] \\ &\leftrightarrow \exists t[(s, t) \in R_b \wedge (t, s') \in R_b] \\ &\leftrightarrow (s, s') \in R_b \circ R_c \\ &\leftrightarrow (s, s') \in R_{b,c} \end{aligned}$$

The case of *while* statements follows directly from the following lemmas A1 and A2

Lemma A1: $(s, s') \in R_{\textit{while p do } b} \rightarrow s(\textit{while p do } b)s' \in T$.

Proof:

Definition A1: For states s, s', program b and predicate P, let $dist_{b,p}(s, s')$ be the least nonnegative integer k, if any, such that there is a sequence $s_0, \ldots, s_k$ of states with the property that

(i) $s_0 = s$,

(ii) $s_k = s'$, and

(iii) $P(s_i) \wedge (s_i, s_{i+1}) \in R_b$ for all nonnegative integers $i<k$;

if no such k exists the distance $dist_{b,p}(s,s')$ is said to be infinite.

We take the following two facts as obvious from Definition A1. First, if $dist_{b,p}(s,s') = n+1$, then P(s) and there is an s_1 such that $(s, s_1) \in R_b$ and $dist_{b,p}(s_1,s') = n$. Second, $(s,s') \in R_{\textit{while p do } b}$ iff $dist_{b,p}(s, s')$ is finite and $\neg P(s')$.

Lemma A1 follows by induction on $dist_{b,p}(s ,s')$. If the distance is zero, then $s = s'$ and from the second fact above we conclude that $\neg P(s)$. Then by HL6, $s(\textit{while p do } b)s' \in T$.

By the first fact above, if $dist_{b,p}(s, s') = n + 1$ we have P(s) and $(s, s_1) \in R_b$ for some s_1 such that $dist_{b,p}(s_1, s') = n$. From $(s, s_1) \in R_b$, by induction we have $s(b)s_1 \in T$. By induction on n, we have $s_1(\textit{while p do } b)s' \in T$. Therefore, by HL7, $s(\textit{while p do } b)s' \in T$.∎

Lemma A2: $s(\textit{while p do } b)s' \in T \rightarrow (s, s') \in R_{\textit{while p do } b}$

Proof: Let $Q(t)$ be the predicate $(s, t) \in (R_p \circ R_b)^*$.

Claim: $Q(s_1) \wedge P(s_1) \wedge s_1(b)s_2 \rightarrow Q(s_2)$.

Proof of Claim: $s_1(b)s_2 \in T \rightarrow (s_1, s_2) \in R_b$ by main induction on a, hence the claim follows trivially.∎

We have $Q(s)$ by definition, and $s(\textit{while p do } b)s' \in T$ by hypothesis. By HL4, we can conclude $Q(s')$, and by HL3 and $s(\textit{while p do } b)s' \in T$, we have $\neg P(s')$.

Now $Q(s') \wedge \neg P(s') \rightarrow (s, s') \in R_{\textit{while p do } b}$ by definition of $R_{\textit{while p do } b}$.∎

7.2 Proof of Lemma 1.

Let Π_a be a set of pca's of the form $P\{a\}Q$.

Definition C1. $max(\Pi_a) = \{(s,t) \mid P(s) \rightarrow Q(t) \text{ for all } P\{a\}Q \in \Pi_a\}$.

Let R be a binary relation on states.

Definition C2. $R_a(R) = \{P\{a\}Q \mid P\{a\}Q \text{ is true for R}\}$.

The following lemma formally establishes that Definition 2 and C1 are equivalent.

Lemma C1. $R \subset max(\Pi_a)$ iff $R_a(R) \supset \Pi_a$.

Proof: Suppose $R \subset max(\Pi_a)$ and that $P\{a\}P' \in \Pi_a$.

Then for any s, s', $P(s) \wedge (s, s') \in R$ implies $P(s) \wedge (s, s') \in max(\Pi_a)$. By definition of *max* and the fact that $P\{a\}P' \in \Pi_a$, we can then conclude $P'(s')$. Thus $P\{a\}P' \in R_a(R)$, by the definition of $R_a(R)$.

Now assume that $\Pi_a \subset R_a(R)$ and $(s, s') \in R$. For any $Q\{a\}Q' \in \Pi_a$ such that $Q(s)$, we have by the definition of $R_a(R)$ that $Q'(s')$. Thus, $(s, s') \in max(\Pi_a)$. ∎

Lemma C2: $R = max(R_a(R))$

Proof: $R \subset max(R_a(R))$ by Lemma C1.

To show equality, suppose $(s_1, s_2) \notin R$. Let E_s be the predicate true only of state s. Then $E_{s_1}\{a\}\neg E_{s_2} \in R_a(R)$ by definition of $R_a(R)$, so $(s_1, s_2) \notin max(R_a(R))$ by definition of *max*.∎

Note that if $\mathbf{M}$ is a relational semantics and Π is the set of pca's true for $\mathbf{M}$, then $\Pi_a = R_a(M_a)$, so Lemma 1 follows immediately from Lemma C2.

7.3 Proof of Theorem 3

Theorem 3: The set of pca's derivable from FH1-5, is equal to $\mathbf{R}$, the standard partial correctness semantics.

The proof that $P\{a\}Q$ true for $\mathbf{R}$ implies $P\{a\}Q$ derivable is by induction on the structure of programs. If $P\{NOP\}Q$ is true for $\mathbf{R}$, then by Definition 1 and R1 we conclude that P implies Q. Hence $P\{NOP\}Q$ is derivable by applying FH5 to the FH1 axiom $P\{NOP\}P$.

If $P\{A\}Q$ is true for $\mathbf{R}$, then by Definitions 1 and 3, P implies $[R_A]Q$, so $P\{A\}Q$ is derivable by applying FH5 to the FH2 axiom $([R_A]Q)\{A\}Q$.

If $P\{a;b\}Q$ is true for $\mathbf{R}$, then $P\{a\}([R_b]Q)$ must be true for $\mathbf{R}$, as the reader can verify from Definitions 1, 3, and R2. Also, $([R_b]Q)\{b\}Q$ is true for $\mathbf{R}$ by Definitions 1 and 3. By induction we may conclude that $P\{a\}([R_b]Q)$ and $([R_b]Q)\{b\}Q$ are derivable, and therefore $P\{a;b\}Q$ is derivable by applying FH3. Finally, suppose $P_1\{\textit{while p do } a\}P_2$ is true for $\mathbf{R}$. Let $Q = [R_{\textit{while p do } a}]P_2$. Then again it follows directly from the definitions that

(1) P_1 implies Q,

(2) $Q \wedge \neg P$ implies P_2, and

(3) $(Q \wedge P)\{a\}Q$ is true for R.

Then by induction, we conclude from (3) that $(Q \wedge P)\{a\}Q$ is derivable. Applying FH4, we can therefore derive $Q\{\textit{while } p \textit{ do } a\}(Q \wedge \neg P)$. But we can apply FH5 to the latter assertion to derive $(P_1 \wedge Q)\{\textit{while } p \textit{ do } a\}(P_2 \vee (Q \wedge \neg P))$ which by (1) and (2) is the same as $P_1\{\textit{while } p \textit{ do } a\}P_2$.

We omit the proof that if $P\{a\}Q$ is derivable then $P\{a\}Q$ is true for $\mathbf{R}$.∎

7.4 Proof of Theorem 4

Theorem 4: $\mathbf{R}$ is the only partial correctness model of PCI-4.

Proof: The following lemma summarizes some facts about weakest antecedent which are used in the proof.

Lemma B1: Let R, R_1, and R_2 be relations on states.

(a) $\exists P'(\models(P \rightarrow [R]P') \wedge \models(P' \rightarrow Q))$ iff $\models(P \rightarrow [R]Q)$.

(b) $[R_1][R_2]Q = [R_1 \circ R_2]Q$

(c) $\models([R^*]Q \rightarrow Q)$

(d) $\models([R^*]Q \rightarrow [R][R^*]Q)$

Proof of B1:

(a) the if direction is trivial and the only if direction is equivalent to soundness of FHI-5.

(b) and (c) follow from definition 3. We omit the details.

(d) We use the following facts which again follow directly from definitions:

(i) $R^* = I \cup R \cup R^2 \cup \ldots = I \cup R \circ R^*$
where $R_1 \cup R_2$ is the union of R_1 and R_2.

(ii) $\models([R_1 \cup R_2]Q \leftrightarrow \models([R_1]Q \wedge [R_2]Q))$.

By (i) $[R^*]Q(s)$ equals $[I \cup R \circ R^*]Q(s)$. Thus $[R^*]Q(s)$ implies (by ii) $[I]Q(s) \wedge [R \circ R^*]Q(s)$ implies (by definition 3) $Q(s) \wedge [R \circ R^*]Q(s)$ which implies $[R \circ R^*]Q(s)$. Thus $\models([R^*]Q \rightarrow [R \circ R^*]Q)$.∎

Let $\overline{\Pi}$ be any partial correctness model of PCI-4. We show that $\overline{\Pi}_a = \mathbf{R}_a$ by induction on structure of program a.

$P\{NOP\}Q \in \overline{\Pi}$ iff (by PCI) $\models(P \rightarrow Q)$ iff (by def. 1) $P\{NOP\}Q \in \mathbf{R}$.

$P\{A\}Q \in \overline{\Pi}$ iff (by PC2) $\models(P \rightarrow [R_A]Q)$ iff (by defs. 1 and 3) $P\{A\}Q \in \mathbf{R}$.

Suppose $a = b;c$. Then $Q\{a\}Q' \in \overline{\Pi}$

iff (by PC3) $\exists P'(Q\{b\}P' \in \overline{\Pi}$ and $P'\{c\}Q' \in \overline{\Pi})$

iff (by induction) $\exists P'(Q\{b\}P' \in \mathbf{R} \wedge P'\{c\}Q \in \mathbf{R})$

iff $\exists P'((\models Q \rightarrow [R_b]P') \wedge \models(P' \rightarrow [R_c]Q'))$

iff (by lemma B1 (a)) $\models(Q \rightarrow [R_b][R_c]Q')$

iff (by lemma B1 (b)) $\models(Q \rightarrow [R_b \circ R_c]Q')$

iff $Q\{a\}Q' \in \mathbf{R}$

We need the following two lemmas for the case of $a = \textit{while } P \textit{ do } b$.

Lemma B2: $Q\{\textit{while } p \textit{ do } b\}Q' \in \mathbf{R}$ implies

$\exists Q''(\models((P \wedge Q'') \rightarrow [R_b]Q'') \wedge \models(Q \rightarrow Q'') \wedge \models((Q'' \wedge \neg P) \rightarrow Q'))$.

Proof of B2: $Q\{\textit{while } p \textit{ do } b\}Q' \in \mathbf{R}$ implies
$\models(Q \rightarrow [R_{\textit{while } p \textit{ do } b}]Q')$ which implies,
by R3, $\models(Q \rightarrow [(R_p \circ R_b)^* \circ R_{\neg p}]Q')$.

By lemma B1 (b) this implies
$\exists Q_1(\models(Q \rightarrow [(R_p \circ R_b)^*]Q_1) \wedge \models(Q_1 \rightarrow [R_{\neg p}]Q'))$.

Then by definition 3,
$\exists Q_1(\models(Q \rightarrow [(R_p \circ R_b)^*]Q_1) \wedge \models((Q_1 \wedge \neg P) \rightarrow Q'))$.

Let $Q'' = [(R_p \circ R_b)^*]Q_1$. Then by definition of Q'', $\models(Q \rightarrow Q'')$. By lemma B1(c), $\models(Q'' \rightarrow Q_1)$. This fact and $\models((Q_1 \wedge \neg P) \rightarrow Q')$, imply that $\models((Q'' \wedge \neg P) \rightarrow Q')$. Thus we need only show that $\models((Q'' \wedge P) \rightarrow [R_b]Q'')$. By lemma B1 (c) and (d), $\models(Q'' \rightarrow [R_p \circ R_b]Q'')$ which by B1(b) and definition 3 implies $\models((Q'' \wedge P) \rightarrow [R_b]Q'')$.∎

Lemma B3: $Q\{\textit{while } p \textit{ do } b\}\ Q' \notin \mathbf{R}$ implies

$\neg\exists Q''(\models(Q \rightarrow Q'') \wedge \models((Q'' \wedge P) \rightarrow [R_b]Q'') \wedge \models((Q'' \wedge \neg P) \rightarrow Q'))$.

Proof of B3: If $Q\{\textit{while } p \textit{ do } b\}\ Q' \notin \leftrightarrow \mathbf{R}$ then by definition 1

(*) $\exists s,s'(Q(s) \wedge (s,s') \in R_{\textit{while } p \textit{ do } b} \wedge \neg Q(s'))$.

Since $(s,s') \in R_{\textit{while } p \textit{ do } b}$ and $R_{\textit{while } p \textit{ do } b} = (R_p \circ R_b)^* \circ R_{\neg p}$, there is a sequence of states $s_0, \ldots, s_k$ such that either $k=0$, $s=s'$ and $\neg P(s')$ or the following conditions hold:

$k>0$
$s=s_0$
$s_k=s'$
$P(s_i)$ for $i<k$
$\neg P(s_k)$
$(s_i, s_{i+1}) \in R_b$ for $i<k$.

Now assume Q″ exists and satisfies

(i) $\models(Q \rightarrow Q'')$

(ii) $\models((Q'' \wedge P) \rightarrow [R_b]Q'')$

(iii) $\models((Q'' \wedge \neg P) \rightarrow Q')$.

If k=0, then (i) and (iii) imply $Q(s) \rightarrow Q'(s')$ is true, which contradicts (*).

If k>0, then by (i) $Q''(s_0)$ is true. By induction we find that $Q''(s_i)$ is true for $i \leq k$. Then (iii) implies that $Q'(s')$ is true, again contradicting (*).∎

Suppose a = *while* p *do* b. Then $Q\{a\}Q' \in \Pi$ iff (by PC4)

$\exists Q''[(P \wedge Q'')\ \{b\}Q'' \in \Pi$ and
$\models(Q \rightarrow Q'')$ and $\models((Q'' \wedge \neg P) \rightarrow Q')]$

iff (by induction)
$\exists Q''[(P \wedge Q'')\{b\}Q'' \in \mathbf{R}$ and
$\models(Q \rightarrow Q'')$ and $\models((Q'' \wedge \neg P) \rightarrow Q')]$

iff $\exists Q''[\models((P \wedge Q'') \rightarrow [R_b]Q'') \wedge \models(Q \rightarrow Q'') \wedge \models((Q'' \wedge \neg P) \rightarrow Q')]$

iff (by lemmas B1 and B2) $Q\{a\}Q' \in \mathbf{R}$.∎

8. Notes

1. In a later paper [Hoare, 1978] Hoare emphasizes the verification aspects of proof rules rather than their use as an alternative specification of the semantics. Nevertheless, others [cf. Dijkstra, 1975] have used proof rules as a means of definition. It is this issue which we address.

2. We avoid the use of the word "nondeterministic" here because nondeterminism is a property of how final states *are computed* from initial states, rather than merely being a property of which initial states *map* to which final states. Nonfunctional relations can only arise from nondeterministic programs, but functional relations do not necessarily arise only from deterministic programs.

3. Technically speaking, HL1-5 are axiom schemes in which a, b may be any programs, Q any predicate, etc.

4. A similar observation is made by Pratt [1976].

5. They omit FH1 -- a minor oversight. Their D3 is misprinted, but it is clear from their Lemma 9 that they intended to state FH4.

6. Theorem 4, in particular the characterization of *while* statements by PC4, is implicit in Lemma 2.3 of deBakker [1975].

7. Technically speaking, we should say that $\mathbf{R}$ is a model of Th(FH1-5) and Th(T1-5) where Th($\mathbf{S}$) refers to the set of theorems deducible in deductive system $\mathbf{S}$.

8. This seems to be the technical content of the frequently heard that pca's cannot be used to *demonstrate* termination, (cf. e.g. [Hoare, 1969], [Manna, 1974]). The remark is correct, but must not be taken to mislead the reader into thinking that pca's are an inherently inadequate semantics. As we have seen in Theorem 6, the complete set of true pca's determines *everything* about $\mathbf{R}$ despite the anti-monotonicity of pca's.

9. Hoare and Lauer refer to "theorems...proved in the relational theory", rather than mentioning models explicitly. But since the relational theory HL1-5 consists solely of axioms without inference rules, we must assume they refer to the usual model-theoretic notion of theorem, namely, an assertion is a theorem when it is true of all models of the axioms. With this interpretation, their formulation of consistency is equivalent to ours.

10. To emphasize the obscurity of Theorem 7, we note that although FH1-4 specify $\mathbf{R}$ according to Theorem 7, it is not true that Th(FH1-4) = Th(FH1-5) or even that $\mathbf{R}$ is the largest model of Th(FH1-4).

We regard the characterization of $\mathbf{R}$ in Theorem 8 as subtle because there is no particular reason to look at largest models in the context of an axiomatization like HL1-5. Indeed, we saw that by adding HL6-7 only one model is possible, so there is no reason to expect or rely on a condition like maximality to force uniqueness.