Computing 21, 273—294 (1979)

Computing

Program Invariants as Fixedpoints

E. M. Clarke, Durham, North Carolina

Received July 27, 1977; revised May 5, 1978

Abstract — Zusammenfassung

Program Invariants as Fixedpoints. We argue that soundness and relative completeness theorems for Floyd-Hoare Axiom Systems ([3], [5], [18]) are really fixedpoint theorems. We give a characterization of program invariants as fixedpoints of functionals which may be obtained in a natural manner from the text of a program. We show that within the framework of this fixedpoint theory, soundness and relative completeness results have a particularly simple interpretation. Completeness of a Floyd-Hoare Axiom System is equivalent to the existence of a fixedpoint for an appropriate functional, and soundness follows from the maximality of this fixedpoint. The functionals associated with regular procedure declarations are similar to the *predicate transformers* of Dijkstra; for non-regular recursions it is necessary to use a generalization of the predicate transformer concept which we call a *relational transformer*.

Programminvarianten als Fixpunkte. Es wird dargelegt, daß die Sätze für Widerspruchsfreiheit und Vollständigkeit für Systeme, die auf Floyd-Hoare-Axiomen basieren ([3], [5], [18]), tatsächlich Fixpunktsätze sind. Die Programminvarianten werden als Fixpunkt-Funktionale charakterisiert, die man auf natürliche Weise vom Programmtext herleiten kann. Es wird gezeigt, daß innerhalb des Rahmen dieser Fixpunkttheorie die Ergebnisse bezüglich Widerspruchsfreiheit und Vollständigkeit eine besonders einfache Interpretation besitzen. Die Vollständigkeit eines Floyd-Hoare-Axiomensystems ist äquivalent zur Existenz eines Fixpunktes für ein geeignetes Funktional. Die Widerspruchsfreiheit folgt aus der Maximalität dieses Fixpunktes. Die Funktionale für reguläre Prozedurdeklarationen ähneln Dijkstras Prädikat-Transformern. Für nichtreguläre Rekursionen braucht man eine Verallgemeinerung des Prädikat-Transformer-Konzepts, das hier relationaler Transformer genannt wird.

1.

1.1 Background

Proof systems for correctness of computer programs may be treated as formal logical systems. Of particular interest are deductive systems for partial correctness based on the use of *invariant assertions*. Examples of such systems are described by Floyd [10] and Hoare [14]. Formulas in Floyd-Hoare Axiom Systems are triples $\{P\}\ A\ \{Q\}$ where A is a statement of the programming language and P and Q are predicates in the language of first order predicate calculus (*the assertion language*). A partial correctness formula $\{P\}\ A\ \{Q\}$ is true iff whenever P is satisfied by the initial program state and A is executed, then either A will fail to terminate or Q will be satisfied by the final program state. An axiom or *rule of inference* is associated with each statement type in the programming language, e. g.

0010-485X/79/0021/0273/$ 04.40

$$\frac{\{P \wedge b\}\ A\ \{P\}}{\{P\}\ \text{WHILE } b \text{ DO } A\ \{P \wedge \sim b\}}$$

Proofs of correctness for programs are constructed by using the rules of inference for the individual statements together with a proof system for the assertion language.

Once a method of proof has been formalized, it becomes important to determine which steps in the proof process are most difficult. With Floyd-Hoare Axiom System experience shows that there are two main sources of difficulty: (1) choosing the correct program invariants (e.g. P in the WHILE axiom above) and (2) demonstrating the truth of formulas in the underlying assertion language. This observation is justified by the work of Cook [3] on relative completeness theorems for Floyd-Hoare Axiom Systems. Cook gives an axiom system for a subset of Algol including the WHILE statement and nonrecursive procedures. He proves that (a) if the assertion language satisfies a natural expressibility condition which guarantees the existence of program invariants, and (b) if there is a complete proof system for the assertion language (e.g. the set of true formulas of the assertion language), then a partial correctness formula is true iff it is provable. Extension of Cook's work to other language features are discussed by Gorelick ([13], recursive procedures), Owicki ([21], concurrent processes), Clarke ([2], procedure parameters under various restrictions on scope of variables), and Cherniavsky ([1], loop languages). Incompleteness results for language features, such as call-by-name parameter passing, are given in Clarke [2] and in Lipton and Snyder [17].

1.2 New Results of this Paper

Completeness results appear to be an important tool in investigations of program correctness; however, many basic open questions remain about the derivation and interpretation of such results. Although proofs of soundness and completeness are often long and tedious, it seems that the underlying ideas are quite simple. Are there general theorems from which many different completeness results may be obtained as special cases? What is the relationship between Cook's definition of completeness and the definition of completeness used in the earlier work of Manna [18] and deBakker and Meertens [5]? Gorelick [13], for example, has shown that a set of three axioms gives completeness for a language with parameterless recursive procedures. DeBakker and Meertens [5], on the other hand, *prove* that an infinite pattern of inductive assertions is necessary to obtain completeness for the same class of programming languages. Is there a way of reconciling these apparently contradictory results?

In this paper we argue that the soundness and relative completeness theorems of Cook *et al.* are really *fixedpoint theorems*. We give a characterization of program invariants as maximal *fixedpoints* of functionals which may be obtained in a natural manner from the program text. We further show that within the framework of this fixedpoint theory soundness and relative completeness theorems have a very simple interpretation. Completeness of a Floyd-Hoare Axiom System is equivalent to the existence of a fixedpoint for an appropriate functional, and sound-

ness follows from the maximality of this fixedpoint. Although fixedpoint techniques were used in the study of program invariants as early as 1970 by Park [20], we believe that this is the first discussion of the extremely close relationship that exists between such fixedpoint results and the soundness and relative completeness theorems of Cook and others.

The functionals associated with *regular recursive procedures* (i.e. flowchartable recursions) are similar to the predicate transformers of Dijkstra ([8], [6], [12]). In order to treat non-regular recursions we must generalize the notion of a program invariant from a single predicate to a binary relation on predicates which is preserved by procedure calls. The appropriate functional in this case is a generalization of the predicate transformer concept which we call a *relational transformer*. The relational transformer maps a programming language statement into the partial correctness relation determined by that statement. Since this mapping is continuous, a fixedpoint theorem characterizing the partial correctness relations of recursive procedures may be obtained.

1.3 Outline of Paper

Section 2 describes a simple recursive programming language. Sections 3 and 4 contain the regular fixedpoint theorem and some of its applications. A general fixedpoint theorem for program invariants applying to nonregular recursions as well as regular recursions is given in Section 5. Applications of the general fixedpoint theorem, including a discussion of the relationship between the completeness results of deBakker [5] and Gorelick [13], are given in Section 6. The paper concludes with a discussion of the results and some remaining open problems.

2. A Simple Recursive Programming Language (RPL)

Any formal treatment of program correctness must include a discussion of the underlying logical system in which the predicates describing a program's behavior are expressed. In this paper the underlying *assertion language* is a first order language with equality which we denote by AL. An interpretation I for AL consists of a set D (the domain of the interpretation), an assignment of functions on D to the function symbols of AL, and an assignment of predicates on D to the predicate symbols of AL. Once an interpretation I has been specified, meanings may be assigned to the variable-free terms and closed formulas of AL.

Let ID be the set of identifiers (i.e. variables) of AL, and let I be an interpretation for AL with domain D. A *program state* is a mapping from ID to D giving the "value" associated with each identifier. We denote the set of all *program states* by S. If t is a term of AL with variables $x_1, x_2, \ldots, x_n$ and s is a program state, then we use the notation $t(s)$ to mean

$$t\frac{s(x_1), \ldots, s(x_n)}{x_1, \ldots, x_n}$$

i.e. the term obtained from t by simultaneously substituting $s(x_1)$ for $x_1, \ldots, s(x_n)$ for x_n. Likewise we may define $P(s)$ where P is a formula of AL.

Frequently it will be convenient to identify a predicate P with the set $\{s \mid I[P(s)] = \text{true}\}$ of program states which make P true. If this identification is made, then the power set 2^S will be the set of all possible predicates, *false* will correspond to the empty state set, and *true* will correspond to the set S of all program states. Also logical operations on predicates can be interpreted as set theoretic operations on subsets of S, i.e. "or" becomes "union", "and" becomes "intersection", "not" becomes "complement", and "implies" becomes "is a subset of". In general there will be many sets of states which are not expressible by formulas of the assertion language AL.

An *RPL program* is a pair $\langle E, A \rangle$ where E is a list of *procedure declarations* and A is a *statement*. Procedure declarations have the form: "PROC $X = \tau$" where X is the *procedure name* and τ is the *body of the procedure* (also a statement). A statement is either a *compound statement* "$(A_1; A_2)$", a *conditional statement* "$(b \rightarrow A_1, A_2)$", an *assignment statement* "$x := e$", a *procedure call* "X" where X is a procedure name, or a *null statement* "NULL". To simplify the treatment of recursion we also allow the undefined statement "Ω". We will require that the boolean expressions of *RPL* conditionals be *quantifier-free* formulas of AL, and that the right-hand sides of *RPL* assignment statements be *terms* in AL. Thus, *side-effects* are not permitted as a consequence of arithmetic or boolean expression evaluation.

The meaning of an *RPL* statement A can only be described once an interpretation I has been specified. Since the statement A may contain procedure calls, the meaning of A also depends on the set E of procedure declarations in the program in which A occurs. Relative to a particular interpretation I and set of procedure declarations E, the meaning of A is a function $M_{I,E}[A]: S' \rightarrow S'$ ($S' = S \cup \{\perp\}$ where $\perp$ represents undefined) which gives the effect of the execution of A on the values of the identifiers occurring in A. There are many ways that $M = M_{I,E}$ can be defined — in terms of computation sequences as in [3] or as the least fixedpoint of a continuous functional as in [6]. We will simply list a number of the properties that M can be shown to satisfy regardless of the definition used.

2.1

Proposition:

(a) $M[\Omega](s) = \perp$

(b) $M[\text{NULL}](s) = s$

(c) $M[x := e](s) = s'$ where $s'[v] = \begin{cases} s[v], & x \neq v \\ I[e(s)] & \text{o. w.} \end{cases}$

(d) $M[(A_1; A_2)](s) = M[A_2](M[A_1](s))$

(e) $M[(b \rightarrow A_1, A_2)](s) = \begin{cases} M[A_1](s), & s \in b \\ M[A_2](s), & \text{o. w.} \end{cases}$

(f) $M[X](s) = M[\tau](s)$ where X is a procedure name and "PROC $X = \tau$" is a declaration in E.

Let PF be the set of partial functions from S to S, i.e. PF is the set of functions $f: S' \rightarrow S'$ such that $f(\perp) = \perp$. Let $f_1, f_2 \in PF$; we write $f_1 \sqsubseteq f_2$ iff for all x, $f_1(x) = \perp$ or $f_1(x) = f_2(x)$. A sequence $\{f_i\}_{i \geq 0}$ of functions in PF is a *chain* iff for

all i, $f_i \sqsubseteq f_{i+1}$. If $\{f_i\}_{i\geq 0}$ is a chain of partial functions, then $\bigsqcup_{i\geq 0} f_i$ is the partial function defined by

$$(\bigsqcup_{i\geq 0} f_i)(x) = \begin{cases} y \text{ if there exists a } j\geq 0 \text{ such that } f_i(x)=y \text{ for } i\geq j \\ \perp \text{ o.w.} \end{cases}$$

2.2

Proposition: Let X be a procedure name occurring in a program which contains the procedure declaration "PROC $X=\tau$". Let the statement sequence $\{X^i\}_{i\geq 0}$ be defined inductively by $X^0=\Omega$, $X^{i+1}=\tau\frac{X^i}{X}$ then

(a) the sequence $\{M[X^i]\}_{i\geq 0}$ is a chain in PF

(b) $M[X]=\bigsqcup_{i\geq 1} M[X^i]$.

Partial correctness formulas will have the form $\{P\}\ A\ \{Q\}$ where A is a program statement and P and Q are formulas of the assertion language AL.

2.3

Definition: $\{P\}\ A\ \{Q\}$ is *true with respect to* I ($\models_I \{P\}\ A\ \{Q\}$) iff

$$\forall s, s' \in S\,[s \in P \wedge M[A](s)=s' \Rightarrow s' \in Q].$$

If the interpretation I is clear from context we will simply write $\models \{P\}\ A\ \{Q\}$. $WP[A](Q)$ will denote the *weakest precondition for partial correctness* corresponding to the statement A and the post-condition Q. $WP[A](Q)$ is characterized by (i) $\models \{WP[A](Q)\}\ A\ \{Q\}$ and (ii) $\models \{P\}\ A\ \{Q\} \Rightarrow \models P \rightarrow WP[A](Q)$. In terms (i) $\models \{WP[A](Q)\}\ A\ \{Q\}$ and (ii) $\models \{P\}\ A\ \{Q\} \Rightarrow \models P \rightarrow WP[A](Q)$. In terms of the meaning function M, $WP[A](Q)$ may be defined by $WP[A](Q)=\{s\in S \mid M[A](s)=\perp$ or $M[A](s)\in Q\}$. The weakest precondition $WP[A](Q)$ may not be expressible by a formula of the assertion language AL. Cook [3] demonstrates this in the case in which the assertion language is Presburger arithmetic. Wand [25] gives another example of the same phenomenon.

2.4

Definition: The assertion language AL is *expressive* with respect to the interpretation I iff for all statements A and post-conditions Q in AL, there is a formula of AL which expresses $WP[A](Q)$.

Subsequently we will see that expressibility insures the existence of invariants for while loops and recursive procedures. Although some choices of AL and I do not give expressibility, many realistic choices do give expressibility; if AL is the full language of number theory and I is an interpretation in which the symbols of number theory receive their usual meanings, then AL will be expressive with respect to I. Also if the domain of I is finite, expressibility is assured.

Likewise, $SP[A](P)$ will denote the *strongest post-condition* corresponding to statement A and precondition P. $SP[A](P)$ is characterized by (i) $\models \{P\}\ A\ \{SP[A](P)\}$ and (ii) $\models \{P\}\ A\ \{Q\} \Rightarrow \models SP[A](P) \rightarrow Q$. $SP[A](P)$ may also be defined in terms of the meaning function M: $SP[A](P) = \{M[A](s) \mid s \in P\}$. It is not difficult to show that weakest preconditions are expressible by formulas of AL iff strongest post-conditions are also expressible. Thus an equivalent definition of "expressive" may be given in terms of $SP[A](P)$.

If H is a Floyd-Hoare Axiom System and T is a complete proof system for the language AL with respect to I (i.e. for the formulas of AL which are true under interpretation I), then a proof in the system (H, T) will consist of a sequence of partial correctness formulas $\{P\}\ A\ \{Q\}$ and formulas of AL each of which is either an axiom (of H or T) or follows from previous formulas by a rule of inference (of H or T). If $\{P\}\ A\ \{Q\}$ occurs as a line in such a proof, then we write $\vdash_{H,T} \{P\}\ A\ \{Q\}$. In a similar manner, we may define $\pi_1 \vdash_{H,T} \pi_2$ where π_1 and π_2 are sets of partial correctness assertions. If H and T are clear from context, we will write "$\vdash$" instead of "$\vdash_{H,T}$".

2.5

Definition (Cook): A Floyd-Hoare Axiom System H for a programming language PL is *sound and complete* iff for all AL, T, and I such that (a) AL is expressive relative to I and (b) T is a complete proof system for AL with respect to I, $\models_I \{P\}\ A\ \{Q\} \Leftrightarrow \vdash_{H,T} \{P\}\ A\ \{Q\}$.

3. The Regular Fixedpoint Theorem

This section contains the first of the two fixedpont theorems which are the main technical results of the paper. These theorems characterize the invariants of a recursive procedure as (pre-) fixedpoints of a functional which may be obtained in a natural manner from the procedure's declaration. Subsequently, we shall argue that these fixedpoint theorems are logically equivalent to the soundness and relative completeness theorems of Cook *et al.* The first fixedpoint theorem is for regular recursive procedures. In this case the associated functionals are similar to the *predicate transformers* of Dijkstra [8].

3.1

Definition [5]: Let X be a procedure name, τ_1 and τ_2 statements. Then

(a) X is *regular in* X.

(b) If τ_1 does not contain X and τ_2 is regular in X, then $(\tau_1; \tau_2)$ is *regular in* X.

(c) If τ_1 and τ_2 are both regular in X, then $(b \rightarrow \tau_1, \tau_2)$ is *regular in* X.

A procedure declaration "PROC $X = \tau$" is *regular* if τ is regular in X.

Thus, for example, PROC $X = (A_1; (b \rightarrow (A_2\ X), \text{NULL}))$ is regular in X, but PROC $X = (b \rightarrow (A_1; X; A_2), A_3)$ is not.

Let "PROC $X=\tau$" be a regular procedure declaration and let $Q \subseteq S$ be a predicate. The *partial correctness functional* $\Gamma: 2^S \to 2^S$ associated with X and Q is defined as follows: $\Gamma(U)=G(U, \tau, Q)$ where

(A) $G(U, A, R)=WP[A](R)$ if A does not contain any calls on procedure X;

(B) $G(U,(A_1; A_2), R)=G(U, A_1, G(U, A_2, R))$;

(C) $G(U,(b \to A_1, A_2), R)=(b \wedge G(U, A_1, R)) \vee (\sim b \wedge G(U, A_2, R))$;

(D) $G(U, X, R)=U$.

Note that $G(U, \tau, Q)$ will be well-defined as long as τ is regular in X. Thus, for example, the partial correctness functional Γ associated with

$$\text{PROC } X=(A_1;(b \to (A_2; X), \text{NULL}))$$

and predicate Q is

$$\Gamma(U)=WP[A_1]((b \wedge WP[A_2](U)) \vee (\sim b \wedge WP[\text{NULL}](Q)).$$

3.2

Theorem (Regular Fixedpoint Theorem): *Let "PROC $X=\tau$" be a regular procedure declaration, $Q \subseteq S$ be a predicate, and Γ be the partial correctness functional associated with X and Q. Then $WP[X](Q)$ is the unique maximal pre-fixedpoint of Γ, i.e. the unique maximal solution of $U \subseteq \Gamma(U)$.*

Example: If Γ is the partial correctness functional associated with "PROC $X=(A_1;(b \to (A_2; X), \text{NULL}))$", then (i) $WP[X](Q) \subseteq \Gamma(WP[X](Q))$ and (ii) $U \subseteq \Gamma(U)$ implies that $U \subseteq WP[X](Q)$.

$WP[X](Q)$ is also the unique maximal fixedpoint[1] of Γ. For our purposes, however, it will be easier to work with pre-fixedpoints rather then fixedpoints. Before giving the proof of Theorem 3.2, we must prove a number of lemmas which give key properties of weakest preconditions and the partial correctness functional Γ.

3.3

Lemma: *Let A be an RPL statement.*

(A) (*Monotonicity*): *If P and Q are predicates (subsets of S) and $P \subseteq Q$, then $WP[A](P) \subseteq WP[A](Q)$.*

(B) (*Additivity*): *If $\{P_i\}$, $i \geq 0$ is a family of predicates, then*
$WP[A](\bigcup_i P_i)=\bigcup_i WP[A](P_i)$, $WP[A](\bigcap_i P_i)=\bigcap_i WP[A](P_i)$.

Proof of (A): The proof is a direct application of the definition of $WP[A](Q)$ in terms of the meaning function M. Assume that $P \subseteq Q$. If $s \in WP[A](P)$, then either $M[A](s)$ diverges or $M[A](s) \in P$. Since $P \subseteq Q$, it follows that either $M[A](s)$ diverges or $M[A](s) \in Q$. Thus $s \in WP[A](Q)$. Proof of (B) is similar and will be left to the reader.

[1] In fact, if the declaration of X does not contain calls on other procedures besides X, we may prove that the fixedpoints of Γ form a complete lattice under the natural partial ordering on 2^S. The top element of this lattice is the *weakest precondition* for *partial correctness* $WP[X](Q)$ and the bottom element is the *weakest precondition* for *total correctness* $WT[X](Q)$.

3.4

Lemma: *Let X be a procedure name occurring in a program which contains the procedure declaration "PROC $X=\tau$". $Q\subseteq S$ be a predicate. Define the statement sequence $\{X^i\}_{i\geq 0}$ by induction $X^0=\Omega$, $X^{i+1}=\tau\frac{X^i}{X}$ then*

(*A*) *the sequence $\{WP[X^i](Q)\}_{i\geq 0}$ is a descending chain in 2^S, i.e. for all $i\geq 0$ $WP[X^{i+1}](Q)\subseteq WP[X^i](Q)$.*

(*B*) $WP[X](Q)=\bigcap_{i\geq 0} WP[X^i](Q)$.

Proof: (Given in Appendix)

3.5

Lemma: *Let Γ be the partial correctness functional corresponding to the regular procedure X and predicate Q. Then*

(*A*) *If $\{V_i\}$, $i\geq 0$ is a family of predicates in S, then $\Gamma(\bigcap_i V_i)=\bigcap_i \Gamma(V_i)$ and $\Gamma(\bigcup_i V_i)=\bigcup_i \Gamma(V_i)$.*

(*B*) *$U\subseteq V$ implies $\Gamma(U)\subseteq\Gamma(V)$.*

(*C*) *Let the statement sequence X^i be defined as in Lemma 3.4. Then $\Gamma(WP[X^i](Q))=WP[X^{i+1}](Q)$.*

Proof: (*A*) follows directly from 3.3 (B). (B) follows immediately from Lemma 3.5 (A). Since $U\cup(V-U)=V$, we get $\Gamma(V)=\Gamma(U)\cup\Gamma(V-U)$. It follows that $\Gamma(U)\subseteq\Gamma(V)$. To prove (C) we suppose that X has declaration "PROC $X=\tau$" and that τ is regular in X. Since $X^{i+1}=\tau\frac{X^i}{X}$, we see that $G(WP[X^i](Q),\tau,Q)=WP[X^{i+1}](Q)$. By definition of Γ, we obtain $\Gamma(WP[X^i](Q))=WP[X^{i+1}](Q)$.

We are now ready to complete the proof of the regular fixedpoint theorem (Theorem 3.2).

(i) $WP[X](Q)$ is a fixedpoint of Γ.

$$\begin{aligned}\Gamma(WP[X](Q))&=\Gamma(\bigcap_i WP[X^i](Q)) && \text{by Lemma 3.4}\\ &=\bigcap_i \Gamma(WP[X^i](Q)) && \text{by Lemma 3.5}\\ &=\bigcap_i WP[X^{i+1}](Q) && \text{by Lemma 3.5}\\ &=WP[X](Q) && \text{by Lemma 3.4}\end{aligned}$$

(ii) $V\subseteq\Gamma(V)$ implies $V\subseteq WP[X](Q)$. Suppose that $V\subseteq\Gamma(V)$ then $V\subseteq WP[X^0](Q)$, since $WP[X^0](Q)=$ true. Assume that $V\subseteq WP[X^i](Q)$ then by Lemma 3.5 $\Gamma(V)\subseteq\Gamma(WP[X^i](Q))$. Since $V\subseteq\Gamma(V)$, we get $V\subseteq WP[X^{i+1}](Q)$. It follows that $V\subseteq WP[X^i](Q)$ for all $i\geq 0$. Thus $V\subseteq\bigcap_i WP[X^i](Q)$ or $V\subseteq WP[X](Q)$.

This completes the proof of the regular fixedpoint theorem. We end this section by stating some additional properties of weakest preconditions which we will need in later sections. Proofs of these properties may be found in [6] or [19].

3.6

Proposition:

(a) $WP[x:=e](Q)=Q\frac{e}{x}$

(b) $WP[\text{NULL}](Q)=Q$

(c) $WP[\Omega](Q)=\text{true}$

(d) $WP[(A_1;A_2)](Q)=WP[A_1](WP[A_2](Q))$

(e) $WP[(b\to A_1, A_2)](Q)=[b \wedge WP[A_1](Q)] \vee [\sim b \wedge WP[A_2](Q)]$

(f) $WP[X](Q)=WP[\tau](Q)$ provided X has declaration "PROC $X=\tau$".

4. Applications of the Regular Fixedpoint Theorem

To demonstrate the power of the regular fixedpoint theorem, we prove the soundness and completeness of a Floyd-Hoare Axiom System H_0 for a subset RPL_0 of the programming language *RPL*. A statement in the language RPL_0 will be either: (1) an *assignment statement*, "$x:=e$"; (2) a *conditional statement*, "$(b\to A_1, A_2)$"; (3) an *iteration statement*, "LOOP A_1 WHILE b DO A_2 REPEAT"; or (4) a *composition statement*, "$(A_1; A_2)$". The *iteration* statement will be represented within the language *RPL* by a call on the procedure PROC $X=(A_1;(b\to(A_2; X), \text{NULL}))$.

For each basic statement type in the language RPL_0 there is a corresponding axiom or rule of inference in the deduction system H_0. These axioms and rules of inference are listed below:

(1) $\{Q\frac{e}{x}\}\ x:=e\ \{Q\}$ (assignment)

(2) $$\frac{\{P \wedge b\}\ A_1\ \{Q\},\ \{P \wedge \sim b\}\ A_2\ \{Q\}}{\{P\}\ (b\to A_1, A_2)\ \{Q\}}$$ (conditional)

(3) $$\frac{\{P\}\ A_1\ \{R\},\ \{R \wedge b\}\ A_2\ \{P\},\ R \wedge \sim b \to Q}{\{P\}\ \text{LOOP}\ A_1\ \text{WHILE}\ b\ \text{DO}\ A_2\ \text{REPEAT}\ \{Q\}}$$ (iteration)

(4) $$\frac{\{P\}\ A_1\ \{R\},\ \{R\}\ A_2\ \{Q\}}{\{P\}\ (A_1; A_2)\ \{Q\}}$$ (composition)

(5) $$\frac{P\to R_1,\ \{R_1\}\ A\ \{R_2\},\ R_2\to Q}{\{P\}\ A\ \{Q\}}$$ (consequence)

We will show that soundness and completeness of the deduction system H_0 are direct corollaries of the regular fixedpoint theorem. The question of soundness will be considered first. Let T be a sound proof system for the true formulas of the assertion language. We will prove that $\vdash_{H_0, T}\{P\}\ A\ \{Q\} \Rightarrow \models_I \{P\}\ A\ \{Q\}$. Since T is sound, it is sufficient to prove that each instance of an axiom is true; and

that if all of the hypotheses of a rule of inference are true, then the conclusion will be true also. We will only consider the case of the iteration statement. We assume that $\models P \rightarrow WP[A_1](R)$, $\models R \wedge b \rightarrow WP[A_2](P)$, and $\models R \wedge \sim b \rightarrow Q$ hold, and prove that $\models P \rightarrow WP[X](Q)$ must hold also.

From the regular fixedpoint theorem, we know that $WP[X](Q)$ is the unique maximal pre-fixedpoint of $\Gamma(U) = WP[A_1]([b \wedge WP[A_2](U)] \vee [\sim b \wedge Q])$. From $\models R \wedge b \rightarrow WP[A_2](P)$ and $\models R \wedge \sim b \rightarrow Q$, we derive $\models R \rightarrow [b \wedge WP[A_2](P)] \vee [\sim b \wedge Q]$. Since $\models P \rightarrow WP[A_1](R)$ holds, we conclude that $P \rightarrow WP[A_1]([b \wedge WP[A_2](P)] \vee [\sim b \wedge Q])$ and consequently $P \rightarrow \Gamma(P)$ are both true. Making use of the correspondence between "$\subseteq$" and "$\rightarrow$" and the maximality of $WP[X](Q)$, we see that $P \subseteq WP[X](Q)$. Since $P \subseteq WP[X](Q)$ is equivalent to $\{P\}\ X\ \{Q\}$, the proof of soundness for the iteration statement is completed. We see that the soundness of the axiom for the iteration construct follows immediately from the characterization of the weakest precondition as a *maximal* fixedpoint.

We next show that the proof system H_0 is complete. Let T be a complete proof system for the true formulas of the assertion language. Assume also that the assertion language AL is expressive with respect to the interpretation I. We prove that $\models \{P\}\ A\ \{Q\} \Rightarrow \vdash_{H_0, T} \{P\}\ A\ \{Q\}$. The proof uses induction on the structure of the statement A. We only consider the case of the iteration statement. Assume that $\models \{P\}\ X\ \{Q\}$ where X has declaration PROC $X = (A_1; (b \rightarrow (A_2; X), \text{NULL}))$. By the regular fixedpoint theorem,

$$WP[X](Q) = WP[A_1]([b \wedge WP[A_2](WP[X](Q))] \vee [\sim b \wedge Q]).$$

Let $R = [b \wedge WP[A_2](WP[X](Q))] \vee [\sim b \wedge Q]$. R and $WP[X](Q)$ are representable by formulas of AL because of the expressibility condition. Thus each of the following three formulas is true:

(a) $WP[X](Q) = WP[A_1](R)$; (b) $R \wedge b \rightarrow WP[A_2](WP[X](Q))$; (c) $R \wedge \sim b \rightarrow Q$.

Converting to precondition-postcondition notation and using the inductive hypothesis we obtain:

(a) $\vdash \{WP[X](Q)\}\ A_1\ \{R\}$; (b) $\vdash \{R \wedge b\}\ A_2\ \{WP[X](Q)\}$; (c) $\vdash R \wedge \sim b \rightarrow Q$.

By the rule of inference for the iteration statement, it follows that $\{WP[X](Q)\}$ LOOP A_1 WHILE b DO A_2 REPEAT $\{Q\}$ is provable. Since $P \rightarrow WP[X](Q)$ is true and T is a complete proof system for the assertion language, $P \rightarrow WP[X](Q)$ must be provable. By the rule of consequence it follows that $\{P\}$ LOOP A_1 WHILE b DO A_2 REPEAT $\{Q\}$ is provable also. Note that key to the completeness argument is the fact that $WP[X](Q)$ is a fixedpoint of the partial correctness functional associated with the recursive procedure PROC $X = (A_1; (b \rightarrow (A_2; X), \text{NULL}))$.

Conversely, it is possible to prove that the soundness and completeness of the axiom system H_0 imply that $WP[X](Q)$ is the unique maximal pre-fixedpoint of the partial correctness functional Γ associated with procedure X and postcondition Q. Let H_0, T, and I be as previously described. Assume that for all statements A and predicates P, Q in AL, $\vdash_{H_0, T} \{P\}\ A\ \{Q\} \Leftrightarrow \models_I \{P\}\ A\ \{Q\}$.

(i): $WP[X](Q)$ is a pre-fixedpoint of the partial correctness functional Γ. Since $\models \{WP[X](Q)\}\ X\ \{Q\}$, we may use completeness to deduce the existence of predicates U_1, R_1, and V_1 in AL with the properties:

(a) $\vdash WP[X](Q) \rightarrow U_1$; (b) $\vdash \{U_1\}\ X\ \{V_1\}$; (c) $\vdash \{U_1\}\ A_1\ \{R_1\}$;

(d) $\vdash \{R_1 \wedge b\}\ A_2\ \{U_1\}$; (e) $\vdash R_1 \wedge \sim b \rightarrow V_1$; (f) $\vdash V_1 \rightarrow Q$.

By soundness and (f) we obtain:

(a′) $\models WP[X](Q) \rightarrow U_1$; (b′) $\models \{U_1\}\ X\ \{Q\}$; (c′) $\models \{U_1\}\ A_1\ \{R_1\}$;

(d′) $\models \{R_1 \wedge b\}\ A_2\ \{U_1\}$; (e′) $\models R_1 \wedge \sim b \rightarrow Q$.

Using the characterization of weakest preconditions in Section 2, we may rewrite (b′)—(e′) as follows:

(b″) $\models U_1 \rightarrow WP[X](Q)$; (c″) $\models U_1 \rightarrow WP[A_1](R_1)$;

(d″) $\models R_1 \wedge b \rightarrow WP[A_2](U_1)$; (e″) $\models R_1 \wedge \sim b \rightarrow Q$.

From (d″) and (e″) we get, $\models R_1 \rightarrow (b \wedge WP[A_2](U_1))\ V\ (\sim b \wedge Q)$.

From (c″) and monotonicity, $\models U_1 \rightarrow WP[A_1]((b \wedge WP[A_2](U_1))\ V\ (\sim b \wedge Q))$ or $\models U_1 \rightarrow \Gamma(U_1)$.

From (a′) and (b″) we see that U_1 and $WP[X](Q)$ correspond to the same set of program states. Thus $WP[X](Q) \subseteq \Gamma(WP[X](Q))$ and $WP[X](Q)$ is a pre-fixedpoint of Γ.

(ii): If U is a pre-fixedpoint of Γ, then $U \subseteq WP[X](Q)$. Suppose that U is a pre-fixedpoint of Γ, then $\models U \rightarrow WP[A_1]((b \wedge WP[A_2](U))\ V\ (\sim b \wedge Q))$.

If we choose $R = (b \wedge WP[A_2](U))\ V\ (\sim b \wedge Q)$, then we may show:

(a) $\models \{U\}\ A_1\ \{R\}$ (b) $\models \{R \wedge b\}\ A_2\ \{U\}$; (c) $\models R \wedge \sim b \rightarrow Q$.

By soundness of the axiom for the iteration statement, we get $\models \{U\}\ X\ \{Q\}$. Using the characteriziation of weakest preconditions, $\models U \rightarrow WP[X](Q)$ which is equivalent to $U \subseteq WP[X](Q)$.

5. The General Fixedpoint Theorem

The argument used to establish the regular fixedpoint theorem does not generalize to arbitrary recursive procedures. In fact, the partial correctness functional Γ will not in general be defined for *nonregular recursive procedures.* In Section 6 we shall see that the additional complexity of nonregular recursions is also reflected in the proof rules for recursive procedures; it is impossible to adequately specify the meaning of a recursive procedure by means of a single predicate which is left invariant by the execution of a procedure call. In order to treat nonregular recursions we must generalize the notion of a program invariant from a single predicate to a *binary relation on predicates* which expresses the input-output behavior of the recursive procedure.

To express the invariants of arbitrary recursive procedures we introduce a generalization of the weakest precondition concept which we call the *general correctness relation.* The general correctness relation GR maps a programming language statement into the partial correctness relation determined by that statement.

5.1

Definition: Let REL be the universal relation on 2^S, i.e. $\mathrm{REL}=2^S \times 2^S$. The general correctness relation $GR: ST \to 2^{\mathrm{REL}}$ may be defined by

$$GR[A]=\{(P,Q) \mid P, Q \text{ are expressible in } AL \text{ and } P \subseteq WP[A](Q)\}.$$

Since $(P,Q) \in GR[A]$ iff $\models \{P\}\ A\ \{Q\}$, the general correctness relation may be regarded as a notational device for emphasizing the functional relationship between a programming language statement and the partial correctness relation determined by the statement.

To obtain the general correctness relation for a composite program from the general correctness relations of its component parts we will need two operations on binary relations in addition to the standard set theoretic operations (union, intersection, etc.). The first operation is the standard composition operation for relations:

$$R_1 \circ R_2=\{(U,W) \mid \exists V[(U,V) \in R_1 \wedge (V,W) \in R_2]\}.$$

The second operation is the "scalar multiplication" of a relation by a predicate:

$$b * R=\{(P \wedge b, Q) \mid (P,Q) \in R\}.$$

Each of the properties of weakest preconditions listed in Propositions 3.4 and 3.5 of Section 3 has an exact analogue for the general correctness relation GR.

5.2

Proposition:

(a) $GR[\mathrm{NULL}]=\{(P,Q) \mid P, Q \text{ are expressible in } AL \text{ and } \models P \to Q\}$

(b) $GR[x:=e]=\{(P_x^e, P) \mid P \text{ is expressible in } AL\}$

(c) $GR[\Omega]=\mathrm{REL}$

(d) $GR[(A_1; A_2)]=GR[A_1] \circ GR[A_2]$

(e) $GR[(b \to A_1, A_2)]=(b * GR[A_1]) \cap (\sim b * GR[A_2])$

(f) $GR[X]=GR[\tau]$ where X is a procedure with declaration "PROC $X=\tau$".

Proof of (e): $(P,Q) \in GR[(b \to A_1, A_2)]$ implies that $P \subseteq WP[(b \to A_1, A_2)](Q)$. By Proposition 3.6 $P \subseteq (b \wedge WP[A_1](Q)) \vee (\sim b \wedge WP[A_2](Q))$. Thus, $P \wedge b \subseteq WP[A_1](Q)$ and $P \wedge \sim b \subseteq WP[A_2](Q)$. It follows that

$$(P,Q) \in (b * GR[A_1]) \cap (\sim b * GR[A_2]).$$

Since each of the above steps is reversible, we conclude that

$$GR[(b \to A_1, A_2)]=(b * GR[A_1]) \cap (\sim b * GR[A_2]).$$

5.3

Proposition: Let X be a procedure name in a program which contains the procedure declaration "PROC $X=\tau$". Define the statement sequence $\{X^i\}$, $i\geq 0$ by induction $X^0=\Omega$, $X^{i+1}=\tau\frac{X^i}{X}$ then

(A) the sequence $\{GR[X^i]\}$ $i\geq 0$ is a descending chain in 2^{REL}; i.e. for all $i\geq 0$ $GR[X^{i+1}]\subseteq GR[X^i]$

(B) $GR[X]=\bigcap_i GR[X^i]$

Proof of (*A*): Let $(P, Q)\in GR[X^{i+1}]$, then $P\subseteq WP[X^{i+1}](Q)$. But

$$WP[X^{i+1}](Q)\subseteq WP[X^i](Q).$$

Thus $P\subseteq WP[X^i](Q)$ and $(P, Q)\in GR[X^i]$. The proof of (**B**) is also simple and will be left to the reader.

To state the general fixedpoint theorem we will need an analogue of the partial correctness functional Γ defined in Section 3. Let "PROC $X=\tau$" be an arbitrary *RPL* procedure declaration. Since we are now dealing with binary relations on predicates, the appropriate partial correctness functional Δ is a mapping from 2^{REL} to 2^{REL} which may be defined as follows: $\Delta(R)=H(R, \tau)$ where

(A) $H(R, A)=GR[A]$ if the statement A does not contain any calls on procedure X;

(B) $H(R, (A_1; A_2))=H(R, A_1)\circ H(R, A_2)$;

(C) $H(R, (b\rightarrow A_1, A_2))=(b * H(R, A_1))\cap(\sim b * H(R, A_2))$;

(D) $H(R, X)=R$.

Note that $H(R, A)$ will be defined for both regular and nonregular statements A. For example, if X has the nonregular declaration "PROC $X=(b\rightarrow A_1; X; A_2, \text{NULL})$" then the associated partial correctness function is

$$\Delta(R)=(b * GR[A_1]\circ R\circ GR[A_2])\cap(\sim b * GR[\text{NULL}]).$$

5.4

Theorem (General Fixedpoint Theorem): *Let* "PROC $X=\tau$" *be an arbitrary RPL procedure declaration, and let* Δ *be the associated partial correctness functional. Then* $GR[X]$ *is the unique maximal pre-fixedpoint of* Δ, *i.e. the unique maximal solution to* $R\subseteq\Delta(R)$.

The proof of the general fixedpoint theorem uses essentially the same idea as the proof of the regular fixedpoint theorem and will not be given in full here. To illustrate how the proof works we consider the special case in which the procedure X has declaration "PROC $X=(b\rightarrow(A_1; X; A_2), \text{NULL})$". We prove that $GR[X]$ is the unique maximal pre-fixedpoint of

$$\Delta(R)=(b * GR[A_1]\circ R\circ GR[A_2])\cap(\sim b * GR[\text{NULL}]).$$

To see that $GR[X]$ is a fixedpoint of Δ note that:

$$\Delta(GR[X]) = (b * GR[A_1] \circ GR[X] \circ GR[A_2]) \cap (\sim b * GR[\text{NULL}])$$
$$= GR[(b \rightarrow (A_1; X; A_2), \text{NULL})] = GR[X].$$

To see that $GR[X]$ is maximal, assume that $R \subseteq \Delta(R)$. Then

(1) $R \subseteq \text{REL} = GR[X^0]$,

(2) if $R \subseteq GR[X^i]$ then $R \subseteq \Delta(R) \subseteq \Delta(GR[X^i]) = GR[X^{i+1}]$.

Thus, by induction $R \subseteq GR[X^i]$ for $i \geq 0$. It follows that $R \subseteq \bigcap_i GR[X^i]$ or $R \subseteq GR[X]$.

6. Applications of the General Fixedpoint Theorem

6.1 Nonregular Recursions and the Completeness Theorem of deBakker and Meertens

Suppose that we interpret the RPL_0 iteration statement of Section 4 as a call on the nonregular procedure PROC $X = (b \rightarrow (A_1; X; A_2), \text{NULL})$ instead of the regular procedure PROC $X = (A_1; (b \rightarrow (A_2; X), \text{NULL}))$. What would be an appropriate rule of inference for this modified version of the iteration statement? A first choice might be

$$\frac{\{P \wedge b\}\ A_1\ \{P\},\ P \wedge \sim b \rightarrow Q,\ \{Q\}\ A_2\ \{Q\}}{\{P\}\ X\ \{Q\}}. \tag{1}$$

It is not difficult to show that (1) is sound; an argument similar to the one in Section 4 may be used. Unfortunately, neither (1) nor any similar rule of inference gives completeness when used in conjunction with the other rules of inference of H_0. To see that (1) does not give completeness, consider the procedure declaration "PROC $Y = (b \rightarrow (A_1; Y), A_2)$". A good rule of inference for Y is

$$\frac{\{P \wedge b\}\ A_1\ \{P\},\ \{P \wedge \sim b\}\ A_2\ \{Q\}}{\{P\}\ Y\ \{Q\}}. \tag{2}$$

Since Y is regular, the reader may use the techniques of Section 4 to show that (2) gives both soundness and completeness. Next, let P_0, Q_0, and I be such that $\{P_0\}\ X\ \{Q_0\}$ is true with respect to I but $\{P_0\}\ Y\ \{Q_0\}$ is false. For example, choose

$$b \equiv n > 0 \qquad P_0 \equiv n = n_0$$
$$A_1 \equiv n := n - 1 \qquad Q_0 \equiv n = n_0$$
$$A_2 \equiv n := n + 1$$

If (1) gave completeness, there would be formulas U_0 and V_0 expressible in the assertion language AL such that:

(a) $\models P_0 \rightarrow U_0$, (b) $\models \{U_0 \wedge b\}\ A_1\ \{U_0\}$, (c) $\models U_0 \wedge \sim b \rightarrow V_0$;

(d) $\models \{V_0\}\ A_2\ \{V_0\}$, (e) $\models V_0 \rightarrow Q_0$.

Combining (c) and (d), we get $\models \{U_0 \wedge b\}\ A_1\ \{U_0\}$ and $\models \{U_0 \wedge \sim b\}\ A_2\ \{V_0\}$. By axiom (2) it follows that $\models \{U_0\}\ Y\ \{V_0\}$. From the rule of consequence, $\models \{P_0\}\ Y\ \{Q_0\}$ which is a contradiction.

A general version of the above argument is used by Fokkinga [11] to argue that an *infinite pattern of assertions is necessary* in order to obtain completeness for nonregular recursions. In [5], deBakker and Meertens attach a pair of assertions to each node in the infinite tree obtained by unwinding the recursive declaration (" the tree of incarnations"). In the case of the linear recursion

$$\text{"PROC } X = (b \to A_1; X; A_2, \text{NULL})\text{"},$$

they obtain proof scheme:

$$\left.\begin{array}{l} P \to P_0 \\ \{P_i \wedge b\}\ A_1\ \{P_{i+1}\} \\ P_i \wedge \sim b \to Q_i \\ \{Q_{i+1}\}\ A_2\ \{Q_i\} \\ Q_0 \to Q \end{array}\right\} \quad i \in N \qquad (3)$$

$$\{P\}\ X\ \{Q\}$$

Similar proof schemes may be obtained for other nonregular recursions; however, for many of these recursions, it is necessary to index the predicates by strings in $(0+1)^*$ rather than integers.

The general fixedpoint theorem may be used to prove the soundness and completeness of this extended version of the inductive assertion method. To illustrate how such a proof may be constructed we prove the soundness and completeness of (3) above.

To prove soundness we assume that $\{P_i \wedge b\}\ A_1\ \{P_{i+1}\}$, $P_i \wedge \sim b \to Q_i$, and $\{Q_{i+1}\}\ A_2\ \{Q_i\}$ are true for all $i \geq 0$ and show that $\{P_0\}\ X\ \{Q_0\}$ must be true also. Let $K = \{(P_i, Q_i) \mid i \geq 0\}$ then $K \subseteq (b * GR[A_1] \circ K \circ GR[A_2]) \cap (\sim b * GR[\text{NULL}])$. By the general fixedpoint theorem $GR[X]$ is the unique maximal pre-fixedpoint of $\Delta(R) = (b * GR[A_1] \circ R \circ GR[A_2]) \cap (\sim b * GR[\text{NULL}])$. Hence $K \subseteq GR[X]$ and consequently $\models \{P_0\}\ X\ \{Q_0\}$.

To prove completeness, we assume that $\models \{P_0\}\ X\ \{Q_0\}$, i.e. $(P_0, Q_0) \in GR[X]$. By the general fixedpoint theorem

$$GR[X] = (b * GR[A_1] \circ GR[X] \circ GR[A_2]) \cap (\sim b * GR[\text{NULL}]).$$

By expressibility, there exists a pair of predicates P_1, Q_1 such that

$$(Q_1, Q_0) \in GR[A_2] \qquad (4)$$

$$(P_1, Q_1) \in GR[X] \qquad (5)$$

$$(P_0 \wedge b, P_1) \in GR[A_1] \qquad (6)$$

$$(P_0 \wedge \sim b, Q_0) \in GR[\text{NULL}] \qquad (7)$$

In precondition-postcondition notation (4), (5), and (7) may be rewritten as

$$\models \{P_1 \wedge b\}\ A_1\ \{P_1\} \qquad (4')$$

$$\models \{Q_1\}\, A_2\, \{Q_0\} \tag{5'}$$

$$\models P_0\, A \sim b \rightarrow Q_0 \tag{7'}$$

This process may be continued to obtain $P_2, Q_2, P_3, Q_3, \ldots$

Note that the general fixedpoint theorem was used in exactly the same way that the regular fixedpoint theorem was used in Section 4. Completeness of the rule scheme for procedure X follows from the existence of a fixedpoint to the partial correctness functional Δ associated with X. Soundness is obtained from the maximality of the fixedpoint.

The argument used by deBakker and Meertens for the necessity of rules of inference involving infinitively many hypothesis is based on a treatment of predicates as sets of program states. If predicates are formulas in the first order predicate calculus, and assertions are allowed to contain variables which are not changed by the program, then simpler rules of inference may be obtained. Such a proof system based on the work of Cook and Gorelick is given in the next section. Alternatively the proof system of Cook and Gorelick may be viewed as a way of organizing or indexing the predicates in the system of deBakker and Meertens.

6.2 The Completeness Theorem of Cook and Gorelick

The deduction system which we consider in this chapter is based on "induction on depth of recursion" and is a modification of the deduction system described by S. Cook [3] and G. Gorelick [13]. Before stating the axioms and rules of inference of this system, we must introduce some additional terminology.

6.2.1

Definition: Let A be a statement in a *RPL* program J. A variable u is *active* in A iff either:

(i) A is an assignment statement and u occurs in the right or left side of A.

(ii) A is a conditional statement "$b \rightarrow A_1, A_2$" and u occurs in b, u is active in A_1, or u is active in A_2.

(iii) A is a composite statement "$(A_1 ; A_2)$" and u is active in A_1 or u is active in A_2.

(iv) A is a call on a procedure X with declaration "PROC $X \equiv \tau$" occurring in the declaration part of J and u is active in τ.

If u is not active in A, then u is said to be *inactive*.

Intuitively, a variable u is inactive in the statement A if u is not changed by the execution of A and if the value of u does not influence the results of the execution of A. The definition of inactive may be extended to terms and to substitutions of terms for variables.

6.2.2

Definition: A term t is *inactive* if the only variables occurring in t are inactive variables. A substitution σ is an inactive substitution if σ is a substitution of inactive terms for inactive variables.

The deduction system *CG* for arbitrary recursive procedures consists of seven axioms. The first axioms are called the *basic partial correctness axioms*; they are the assignment axiom, the axiom for the null statement, the rule of composition, the rule of the conditional, and the rule of consequence. We refer the reader to Section 4 for statements of these axioms. The two remaining axioms are needed to handle recursive procedures. The first axiom is an induction axiom which allows proofs to be constructed by induction on depth of recursion.

$$CG1 \qquad \frac{\{P\}\ X\ \{Q\} \vdash \{P\}\ \tau\ \{Q\}}{\{P\}\ X\ \{Q\}} \qquad \text{(recursion)}$$

where X has declaration "PROC $X=\tau$".

The second axiom is used to adapt a general induction hypothesis about what a recursive procedure does to a specific call on the procedure.

$$CG2 \qquad \frac{\{P\}\ A\ \{Q\}}{\{\exists\, \bar{c}\,[P\,\sigma \wedge T]\}\ A\ \{\exists\, \bar{c}\,[Q\,\sigma \wedge T]\}} \qquad \text{(adaption)}$$

provided that

(a) T does not contain any free variables which are active in A.

(b) σ is an inactive substitution.

(c) the variables $\bar{c}$ are inactive with respect to A.

We illustrate the use of these axioms by proving $\{n=n_0\}\ X\ \{n=\lfloor n_0/2\rfloor\}$ where X is the procedure with declaration PROC $X=(\mathrm{n}>0\rightarrow(n:=n-2; X; n:=n+1), \mathrm{NULL})$. To simplify notation, let P be the predicate $\{n=n_0\}$, Q be the predicate $\{n=\lfloor n_0/2\rfloor\}$, and T the predicate $\{n_1=n_0-2\}$. We begin by assuming that $\{P\}\ X\ \{Q\}$ is true. If σ is the inactive substitution $\sigma=\frac{n_1}{n_0}$, then

(a) $\{P \wedge n>0\}\ n:=n-2\ \{\exists\, n_1\,[P\,\sigma \wedge T]\}$ and (b) $\{\exists\, n_1\,[Q\,\sigma \wedge T]\}\ n:=n+1\ \{Q\}$

are both provable. By the inductive hypothesis and *CG*2,

$$\text{(c)}\ \{\exists\, n_1\,[P\,\sigma \wedge T]\}\ X\ \{\exists\, n_1\,[Q\,\sigma \wedge T]\}$$

is also provable. Using the rule of consequence to combine (a), (b), and (c), we obtain

$$\vdash \{P \wedge n>0\}\ n:=n-2; X; n:=n+1\ \{Q\}.$$

Since $\{P \wedge \sim b\}$ NULL $\{Q\}$ is also provable, the *conditional* rule may be used to show

$$\vdash \{P\}\ (n>0\rightarrow(n:=n-2; X; n:=n+1), \mathrm{NULL})\ \{Q\}.$$

By the rule of recursion *CG*1, we obtain $\vdash \{P\}\ X\ \{Q\}$.

The deduction system *CG* may be shown to be sound and complete for proving partial correctness of *RPL* programs provided that such programs do not involve *recursive cycles* of procedure calls [13], i.e. provided that procedures are always declared before they are used. The completeness proof may be extended to general *RPL* programs by using a more general version of the rule of recursion

$CG1$. We will outline how soundness and completeness proofs for GR may be obtained as a by-product of the general fixedpoint theorem. To simplify the proof, we will only consider the case of a single recursive procedure X.

Recall that $SP[A](P)$ is the *strongest postcondition* associated with statement A and precondition P. Since the meaning of a procedure is usually given by an assertion about how the procedure transforms its input variables, it is convenient to use strongest postconditions rather than weakest preconditions in the proof of soundness and completeness for CG.

6.3

Lemma: *Let A be a statement in an RPL program J and let P be a predicate in AL.*

(a) *If Q is a formula of AL which does not contain any free variables which are active in A, then $SP[A](P \wedge Q) = SP[A](P) \wedge Q$.*

(b) *If the substitution σ is inactive with respect to A, then $SP[A](P\sigma) = SP[A](P)\sigma$.*

(c) *If the variables $\bar{c}$ are inactive with respect to A, then $SP[A](\exists \bar{c}[P]) = \exists \bar{c}[SP[A](P)]$.*

Proofs of (a), (b), and (c) follow directly from the definition of SP and will be left to the reader.

6.4

Definition: Let P, Q be formulas of the assertion language AL and let X be a procedure. The *partial correctness relation determined by P and Q* is the set $R[P, Q]$ of all ordered pairs of formulas (U, V) such that for some choice of variable list $\bar{c}$, substitution σ, and predicate T the following three conditions are satisfied: (1) $\bar{c}$, σ, and T are all inactive with respect to X; (2) $U \rightarrow \exists \bar{c}[P\sigma \wedge T]$; (3) $\exists \bar{c}[Q\sigma \wedge T] \rightarrow V$.

Intuitively, $R[P, Q]$ is the set of precondition/postcondition pairs which may be deduced from $\{P\}\ X\ \{Q\}$ by means of the rule of consequence and the rule of adaptation. By choosing T to be the predicate *true*, σ to be the *identity substitution*, and $\bar{c}$ to be a list of variables not appearing in P or Q, we see that the pair (P, Q) is itself an element of $R[P, Q]$.

6.5

Lemma: *Let $P \equiv \{\bar{w} = \bar{w}_0\}$ and $Q \equiv \{SP[A](\bar{w} = \bar{w}_0)\}$ where $\bar{w}$ is the list of variables which are active in A and $\bar{w}_0$ is a list of new inactive variables having the same length as $\bar{w}$, then $GR[A] \subseteq R[P, Q]$.*

Proof: Let $(P', Q') \in GR[A]$ then $\models SP[A](P') \rightarrow Q'$. Note that

$$P' \equiv \exists \bar{w}_1 \left[\bar{w} = \bar{w}_1 \wedge P' \frac{\bar{w}_1}{\bar{w}} \right] \equiv \exists \bar{w}_1 [P\sigma \wedge T]$$

where $\bar{w}_1$ is a list of new variables which are inactive in X, σ is the inactive substitution $\frac{\bar{w}_1}{\bar{w}_0}$, and T is the inactive predicate $P' \frac{\bar{w}_1}{\bar{w}}$. Thus,

$$\models SP[A](\exists \bar{w}_1 [P\sigma \wedge T]) \rightarrow Q'.$$

By Lemma 6.3 we see that $\models \exists \bar{w}_1 [SP[A](P)\sigma \wedge T] \rightarrow Q'$. Since $SP[A](P) = Q$, it follows that $\models \exists \bar{w}_1 [Q\sigma \wedge T] \rightarrow Q'$.

6.6

Lemma: *Assume that AL is expressive with respect to I and that T is a complete proof system for the true formulas of AL. Let X be an RPL procedure declaration* "PROC $X=\tau$", *and let Δ be the partial correctness functional associated with X. Then* $\{P\}\ X\ \{Q\} \vdash \{P'\}\ \tau\ \{Q'\}$ *holds iff* $(P', Q') \in \Delta(R[P, Q])$.

Proof: Proof of Lemma 6.6 involves an induction on the structure of τ. A full proof of this lemma is given in the Appendix.

Soundness of the System CG: To prove the soundness of *CG*, we must show that each instance of an axiom is true and that if all of the hypothesis of a rule of inference are true, then the conclusion will be true also. The soundness of the five basic partial correctness axioms and the rule of adaptation *CG*2 may be proved in a straightforward manner, and will be left as exercises to the reader. To prove the soundness of the rule of recursion *CG*1 we assume that

$$\{P\}\ X\ \{Q\} \vdash \{P\}\ \tau\ \{Q\} \tag{1}$$

where X is a procedure with declaration "PROC $X=\tau$". Let Δ be the partial correctness functional associated with X. By the general fixedpoint theorem, we know that $GR[X]$ is the unique maximal pre-fixedpoint of Δ.

The partial correctness relation $R[P, Q]$ is also a pre-fixedpoint of Δ; to see this, let $(U, V) \in R[P, Q]$ with $\models U \to \exists \bar{c}\,[P\,\sigma\,\Lambda\,T]$ and $\models \exists \bar{c}\,[Q\,\sigma\,\Lambda\,T] \to V$. From (1) and the rule of adaptation we see that $\{P\}\,X\,\{Q\} \vdash \{\exists \bar{c}\,[P\sigma\,\Lambda\,T]\}\,\tau\,\{\exists \bar{c}\,[Q\sigma\,\Lambda\,T]\}$. By the rule of consequence $\{P\}\ X\ \{Q\} \vdash \{U\}\ \tau\ \{V\}$. By Lemma 6.6 it follows that $(U, V) \in \Delta(R[P, Q])$.

Since $GR[X]$ is the unique maximal pre-fixedpoint of Δ, $R[P, Q] \subseteq GR[X]$. Since $(P, Q) \in R[P, Q]$, we see that $(P, Q) \in GR[X]$ or equivalently that $\models \{P\}\ X\ \{Q\}$.

Completeness of the System CG: To prove the completeness of *CG*, we must show that if *AL* is expressive with respect to *I* and *T* is a complete proof system for the true formulas of *AL*, then $\models \{U\}\ A\ \{V\}$ implies $\vdash_{CG,T} \{U\}\ A\ \{V\}$. The proof is by induction on the structure of A. We will only consider the case where A is a call on the procedure X; other cases are simpler and will be left to the reader. Let $P \equiv \{\bar{w} = \bar{w}_0\}$ and $Q \equiv SP[X](\bar{w} = \bar{w}_0)$ where $\bar{w}$ and $\bar{w}_0$ are defined in Lemma 6.5. By Lemma 6.5, $GR[X] \subseteq R[P, Q]$. Let Δ be the partial correctness functional associated with X. By the general fixedpoint theorem $GR[X]$ is a pre-fixedpoint of Δ, i.e. $GR[X] \subseteq \Delta(GR[X])$. Since Δ is monotonic, $GR[X] \subseteq \Delta(R[P, Q])$. Since $\models \{P\}\ X\ \{Q\}$, $(P, Q) \in GR[X] \subseteq \Delta(R[P, Q])$. Thus $\{P\}\ X\ \{Q\} \vdash \{P\}\ \tau\ \{Q\}$ by Lemma 6.6. By the rule of recursion, we obtain $\vdash \{P\}\ X\ \{Q\}$. If $\models \{U\}\ X\ \{V\}$ then $(U, V) \in GR[X] \subseteq R[P, Q]$. By Lemma 6.6 it follows that $\{P\}\,X\,\{Q\} \vdash \{U\}\,X\,\{V\}$. Since $\vdash \{P\}\ X\ \{Q\}$, we conclude that $\vdash \{U\}\ X\ \{V\}$ as required.

7. Conclusion

In this paper we have demonstrated how soundness and relative completeness theorems may be viewed as fixedpoint theorems for functionals. We believe that this alternative point of view sheds light on the meaning of soundness and relative

completeness results and promises to simplify the derivation of such results. We are currently attempting to extend this fixedpoint theory to additional programming language features such as parallelism [21]. We believe that the approach outlined in this paper may also be helpful in understanding recent incompleteness theorems for Floyd-Hoare Axiom Systems ([2], [17]). These theorems use classical undecidability techniques to show that for certain programming language constructs it is impossible to obtain axiom systems which give soundness and relative completeness. We conjecture that for many programming language constructs it may be easier to prove the nonexistence of a fixedpoint than to prove an undecidability results. Finally, we are exploring the question of whether the explicit formulas for program invariants given in Section 3 may be used to automatically generate invariants of loops and recursive procedures. The work of Suzuki [24] and Cousot [4] suggests that invariants can indeed be generated in this manner.

Appendix

Proof of Lemma 3.4 (*A*): Let $s \in WP[X^{i+1}](Q)$ then $M[X^{i+1}](s) = \perp$ or there is an s' such that $M[X^{i+1}](s) = s'$ and $s' \in Q$. If $M[X^{i+1}](s) = \perp$, then $M[X^i](s) = \perp$ also. If $M[X^{i+1}](s) = s'$ then either $M[X^i](s) = \perp$ or $M[X^i](s) = s'$ and $s' \in Q$. Thus in either case $s \in WP[X^i](Q)$.

Proof of Lemma 3.4 (*B*): Let $s \in WP[X](Q)$ then either $M[X](s) = \perp$ or there is an s' such that $M[X](s) = s'$ and $s' \in Q$. Thus either for all $i \geq 0$, $M[X^i](s) = \perp$ or there is a j such that $i \geq j$ implies $M[X^i](s) = s'$. But this implies that for each i either $M[X^i](s) = \perp$ or there is an s' such that $M[X^i](s) = s'$ and $s' \in Q$. It follows that $s \in WP[X^i](Q)$ for all i. Hence $s \in \bigcap_{i \geq 0} WP[X^i](Q)$ and thus $WP[X](Q) \subseteq \bigcap_{i \geq 0} WP[X^i](Q)$.

Conversely, let $s \in \bigcap_{i \geq 0} WP[X^i](Q)$, then for all $i > 0$, $s \in WP[X^i](Q)$. So for all $i \geq 0$, $M[X^i](s) = \perp$ or there exists an $s' \in S$ such that $M[X^i](s) = s'$ and $s' \in Q$. Since the sequence $M[X^i](s)$ is a chain, we get that either for all $i \geq 0$ $M[X^i](s) = \perp$ or there exists s' and j such that $i \geq j$ implies that $M[X^i](s) = s'$ and $s' \in Q$. Hence $M[X](s) = \perp$ or there is an s' such that $M[X](s) = s'$ and $s' \in Q$. This means that $s \in WP[X](Q)$. We conclude that $\bigcap_{i \geq 0} WP[X^i](Q) \subseteq WP[X](Q)$.

Proof of Lemma 6.6: Assume that AL is expressive with respect to I and that T is a complete proof system for the true formulas of AL. We prove that for all statements A, $\{P\}\ X\ \{Q\} \vdash \{P'\}\ A\ \{Q'\}$ iff $(P', Q') \in H(R[P, Q], A)$ where H is the auxiliary function used in the definition of Δ. ($\Leftarrow$) Proof is by induction on the structure of A. If A does not contain a call on procedure X, then $H(R[P, Q], A) = GR[A]$. Since $(P', Q') \in GR[A]$ implies that $\models \{P'\}\ A\ \{Q'\}$ and since the basic partial correctness axioms are complete for RPL programs not involving recursion, we conclude $\vdash \{P'\}\ A\ \{Q'\}$.

If $A =$ "$(A_1; A_2)$" then $H(R[P, Q], A) = H(R[P, Q], A_1) \circ H(R[P, Q], A_2)$. If $(P', Q') \in H(R[P, Q], A)$ then by the expressibility condition there exists a formula

U of AL such that $(P', U) \in H(R[P, Q], A_1)$ and $(U, Q') \in H(R[P, Q], A_2)$. By the induction hypothesis, $\{P\}\ X\ \{Q\} \vdash \{P'\}\ A_1\ \{U\}$ and $\{P\}\ X\ \{Q\} \vdash \{U\}\ A_2\ \{Q'\}$. By the rule of composition it follows that $\{P\}\ X\ \{Q\} \vdash \{P'\}\ (A_1; A_2)\ \{Q'\}$.

The case of the conditional statement i.e. $A = "(b \to A_1, A_2)"$ is similar to the case of the composite statement and will be left to the reader. If A is a call on the procedure X, then $H(R[P, Q], A) = R[P, Q]$ and $(P', Q') \in [P, Q]$ implies that there exists $\bar{c}$, σ, and T as described in Definition 6.4 such that $\models P' \to \exists \bar{c}\,[P\,\sigma\,\Lambda\,T]$ and $\models \exists \bar{c}\,[Q\,\sigma\,\Lambda\,T] \to Q'$. By the rule of adaptation and the rule of consequence it follows that $\{P\}\ X\ \{Q\} \vdash \{P'\}\ A\ \{Q'\}$. ($\Rightarrow$) Proof is by induction on the length of the proof of $\{P'\}\ A\ \{Q'\}$ from $\{P\}\ X\ \{Q\}$. We will only consider the cases in which the last line of the proof was obtained by the use of the rule of composition or the rule of adaptation. If the last line was obtained by the rule of composition, then $\{P\}\ X\ \{Q\} \vdash \{P'\}\ A_1\ \{U\}$ and $\{P\}\ X\ \{Q\} \vdash \{U\}\ A_2\ \{Q'\}$ where $A = (A_1; A_2)$. By the inductive assumption $(P', U) \in H(R[P, Q], A_1)$ and $(U, Q') \in H(R[P, Q], A_2)$. Since $H(R[P, Q], A) = H(R[P, Q], A_1) \circ H(R[P, Q], A_2)$, it follows that $(P', Q') \in H(R[P, Q], A)$. If the last line was obtained by the rule of adaptation, then A is simply a call on procedure X. Furthermore there exist inactive $\bar{c}$, σ, and T such that $\models P' \to \exists \bar{c}\,[P\sigma\,\Lambda\,T]$ and $\models \exists \bar{c}\,[Q\sigma\,\Lambda\,T] \to Q'$. Since $H(R[P,Q], A) = R[P,Q]$ in this case, it follows that $(P', Q') \in H(R[P, Q], A)$.

References

[1] Cherniavsky, J., Kamin, S.: A complete and consistent Hoare axiomatics for a simple programming language. Proceedings of the 4th POPL, 1977.

[2] Clarke, E. M.: Programming language constructs for which it is impossible to obtain good Hoare-like axiom systems. Proceedings of the 4th POPL, 1977.

[3] Cook, S. A.: Axiomatic and interpretative semantics for an algol fragment. Technical Report 79, Department of Computer Science, University of Toronto, 1975 (to be published in SCICOMP).

[4] Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of Programs by construction or approximation of Fixpoints. Proceedings of the 4th POPL, 1977.

[5] deBakker, J. W., Meertens, L. G. L. Th.: On the completeness of the induction assertion method. Mathematical Centre, December 1973.

[6] de Bakker, J. W.: Fixed point semantics and Dijkstra's fundamental invariance theorem. Mathematical Centre, January 1975.

[7] deBakker, J. W.: Flow of control in the proof theory of structured programming. Mathematical Centre, 1975.

[8] Dijkstra, E. E.: A simple axiomatic basis for programming language constructs. Lecture notes from the International Summer School on Structured Programming and Programmed Structures, Munich, Germany, 1973.

[9] Donahue, J.: Mathematical semantics as a complementary definition for axiomatically defined programming language constructs. Technical Report CSRG-45, Computer Systems Research Group, University of Toronto, December 1974.

[10] Floyd, R. W.: Assigning meaning to programs. In: Mathematical Aspects of Computer Science Proc. Symposia in Applied Mathematics (Schwartz, J. T., ed.), Vol. 19, pp. 19—32. Amer. Math. Soc. 1967.

[11] Fokkinga, M. C.: Inductive assertion patterns for recursive procedures. Techn. University Delft Report, 1973.

[12] Gerhart, S. L.: Proof theory of partial correctness verification systems. SIAM J. Comput. 5 (1976).

[13] Gorelick, G.: A complete axiomatic system for proving assertions about recursive and non-recursive programs. Technical Report No. 75, Department of Computer Science, University of Toronto, January 1975.
[14] Hoare, C. A. R.: An axiomatic approach to computer programming. CACM *12*, 322—329 (1969).
[15] Hoare, C. A. R.: Procedures and parameters: An axiomatic approach. Symposium on Semantics of Algorithmic Languages (Engeler, E., ed.), pp. 102—116. Berlin-Heidelberg-New York: Springer 1971.
[16] Hoare, C. A. R., Lauer, P. E.: Consistent and complementary formal theories of the semantics of programming languages. Acta Informatica *3*, 135—154 (1974).
[17] Lipton, R.: A necessary and sufficient condition for the existence of Hoare Logics. 18 Annual Symposium on Foundations of Computer Science, 1977.
[18] Manna, Z., Pnueli, A.: Formalization of properties of functional programs. JACM *17*, 555—569 (1970).
[19] McGowan, C., Misra, J.: A mathematical basis for Dijkstra-Hoare semantics. Technical Report No. 73-73, Center for Computer and Information Sciences, Brown University, November 1973.
[20] Park, D.: Fixpoint induction and proofs of program properties. Machine Intelligence *5*, 59—78 (1970).
[21] Owicki, S.: A consistent and complete deductive system for the verification of parallel programs. 8th Annual Symposium on Theory of Computing, 1976.
[22] Scott, D.: Outline of a mathematical theory of computation. Proceeding of Fourth Annual Princeton Conference on Information Science and Systems. Princeton, pp. 169—176, 1970.
[23] Scott, D.: The lattice of flow diagrams. Semantics of Algorithmic Languages (Springer Notes in Mathematics, Vol. 188), (Engeler, E., ed.), pp. 311—366. Berlin-Heidelberg-New York: Springer 1971.
[24] Suzuki, N., Ishihata, K.: Implementation of an array bound checker. Proceedings of the 4th POPL, 1977.
[25] Wand, M.: A new incompleteness result for Hoare's system. 8th Annual Symposium on Theory of Computing, 1976.
[26] Yeh, R. T., Reynolds, C.: Induction as the basis for program verification. IEEE Transactions on Software Engineering, SE-2(4), 244—252 (1976).

Dr. E. M. Clarke
Assistant Professor
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138, U.S.A.

Printed in Austria
Druck: Paul Gerin, A-1021 Wien