SIAM J. COMPUT.
Vol. 9, No. 4, November 1980

0097-5397/80/0904-0001 $01.00/0

COMPLETENESS WITH FINITE SYSTEMS OF INTERMEDIATE ASSERTIONS FOR RECURSIVE PROGRAM SCHEMES*

KRZYSZTOF R. APT† AND LAMBERT G. L. T. MEERTENS‡

Abstract. It is proved that in the general case of arbitrary context-free schemes a program is (partially) correct with respect to given initial and final assertions if and only if a suitable *finite* system of intermediate assertions can be found. Assertions are allowed from the extended state space $\mathcal{V} \times \mathcal{V}$. This result contrasts with the results of [2], where it is proved that if assertions are taken from the original state space $\mathcal{V}$, then in the general case an *infinite* system of intermediate assertions is needed. The extension of the state space allows a unification in the relational framework of [2], of the (essence of the) results of [2], and of [4], [5] and [6], and provides a semantic counterpart of the use of auxiliary variables.

Key words. partial correctness, intermediate assertions, relational framework, extended state space, recursive program schemes

1. Introduction. De Bakker and Meertens proved in [2] that an *infinite* system of intermediate assertions is needed to prove the completeness of the inductive assertion method in the case of an arbitrary system of (mutually) recursive parameterless procedures. On the other hand, Gorelick in [5] extended the results of [3] and obtained a completeness result for a Hoare-like axiomatic system (see [7]) for a fragment of ALGOL 60 in which (deterministic) systems of recursive procedures are allowed. Thus any true asserted statement is provable. (Observe, however, that the axiomatic system uses an oracle determining the truth of formulas from the underlying assertion language.) From the proof we can extract all intermediate assertions about atomic substatements of the original program. Since proofs are finite, we obtain a *finite* system of intermediate assertions, thus apparently contradicting the result of [2]. Also [4] and [6] avoid the necessity of an infinite number of assertions by using an extension of the inductive assertion method.

The purpose of this paper is to investigate this issue in the relational framework of [2] and to obtain, within that framework, a unification of the (essence of the) results of [2] and of [4], [5] and [6]. The solution of the apparent contradiction lies in the fact that in [4], [5] and [6] auxiliary variables are used (to store the initial values of variables). These auxiliary variables have no semantic counterpart in the relational framework of [2]. Semantically, the use of auxiliary variables corresponds to the use of states which have an additional coordinate (from a space $\mathcal{W}$) inaccessible to a program. We shall call the domain $\mathcal{V} \times \mathcal{W}$ of such states an extended state space.

We prove that if one allows intermediate assertions from the extended state space $\mathcal{V} \times \mathcal{V}$, then one can always find a *finite* system of intermediate assertions. More precisely, a program is partially correct with respect to given initial and final assertions if and only if a suitable finite system of assertions from the extended state space can be found. Thus for the space $\mathcal{W}$ one can take the original state space $\mathcal{V}$. Theorem 4.4 of [2] shows that for $\mathcal{W}$ one could also take the set of all so-called index-triple sequences, so that these two completeness results differ only in the choice of the extended state space. Our choice is both more economical and easier to use in the concrete proofs.

* Received by the editors December 27, 1978, and in revised form October 19, 1979. This publication is a revised form of the Mathematical Centre Report IW 84/77.

† Faculty of Economics, Erasmus University Rotterdam, P.O. Box 1738, Rotterdam, The Netherlands.
‡ Mathematical Centre, Kruislaan 413, Amsterdam, The Netherlands.

In [2] it is proved that in the case of regular declaration schemes (corresponding to flow-chart programs) one can always find a finite system of intermediate assertions taken from the original state space. In more syntactical terms this can be interpreted as a statement that auxiliary variables are not needed for correctness proofs in the case of flow-chart programs. They are needed in the general case of arbitrary systems of (parameterless) procedure declarations.

In the relational framework any subset of the state space can be taken as an assertion. This is not the case with a more syntactical approach in which assertions are formulas from an assertion language. These two different approaches lead to different types of completeness results. Thus one should be cautious in translating results from one framework into the other because there can exist subsets of the state space which do not correspond to (are not defined by) any formula from the assertion language. This problem within the relational framework could be resolved by defining a language over the state space in which assertions could be expressed. However, a natural question then arises as to which formulas (subsets) should be accepted as assertions. This problem has been studied in [1].

2. Preliminaries. As in [2] we shall use binary relations over the state space to provide an interpretation for systems of mutually recursive procedures. More precisely, given a set $\mathscr{P} = \{P_1, \cdots, P_n\}$ of procedure symbols, we define a language of "statements" $\mathscr{S}(\mathscr{P})$ as follows: let $\mathscr{A} = \{I, A_1, A_2, \cdots\}$ be a set of "elementary action" symbols, $\mathscr{B} = \{t_1, t_2, \cdots\}$ a set of "Boolean expressions." $\mathscr{S}(\mathscr{P})$ is then the least set containing $\mathscr{A} \cup \mathscr{B} \cup \mathscr{P}$ that is closed under the operations ";" (sequencing) and "$\cup$" (nondeterministic choice).

By a declaration scheme we mean a set $\mathscr{D} = \{P_1 \Leftarrow S_1, \cdots, P_n \Leftarrow S_n\}$, where for $i = 1, \cdots, n$, $P_i \in \mathscr{P}$, $S_i \in \mathscr{S}(\mathscr{P})$.

In [2] a theory of partial correctness and inductive assertions has been worked out in a relational framework. The meaning of a program is viewed as a *binary relation* over the state space, i.e., a set of pairs of initial and final states, whereas an assertion is viewed as a *subset* of the state space, i.e., the set of states satisfying the assertion. We recall some definitions from [2] which are used below.

Let $\mathscr{V}$ be the domain of states. Letters $R, R_1, \cdots$ denote binary relations over $\mathscr{V}$; p, q, r subsets of $\mathscr{V}$; x, y, z elements of $\mathscr{V}$.

$R_1; R_2 = \{(x, y): \exists z[xR_1z \wedge zR_2y]\}$,

$p_+ = \{(x, x): x \in p\}$,

$p \circ R = \{y: \exists x[x \in p \wedge xRy]\}$,

$\breve{R} = \{(x, y): yRx\}$,

Ω denotes the empty set.

Throughout the paper we use the convention from [2] that in any expression involving programs and assertions built up by using ;, $\cup$ or $\subseteq$ we suppress the subscript "$_+$".

So, for example, if we write $p; R \subseteq R; q$ we actually mean $p_+; R \subseteq R; q_+$, i.e., $\forall x, y[(x \in p \wedge xRy) \to y \in q]$, or (informally speaking) that the program R is partially correct with respect to p and q. We shall need the following results proved in [2].

LEMMA 1.

(i) $(R_1; R_2); R_3 = R_1; (R_2; R_3)$ $\quad (= R_1; R_2; R_3$, from now on),

(ii) $R_1; (R_2 \cup R_3) = R_1; R_2 \cup R_1; R_3$,

(iii) $(R_1 \cup R_2); R_3 = R_1; R_3 \cup R_2; R_3$,

(iv) $p \circ (R_1; R_2) = (p \circ R_1) \circ R_2$.

If $X_1, \cdots, X_n, Y_1, \cdots, Y_n$ are subsets of $\mathscr{V} \times \mathscr{V}$, then by definition $(X_1, \cdots, X_n) \leqq (Y_1, \cdots, Y_n)$ iff $X_i \subseteq Y_i$, for $i = 1, \cdots, n$ ($\leqq$ is a partial ordering).

Let $\mathscr{D} = \{P_1 \Leftarrow S_1, \cdots, P_n \Leftarrow S_n\}$ be a declaration scheme. By an *interpretation* $i_{\mathscr{D}}$ into a state space $\mathscr{V}$ we mean a mapping from $\mathscr{S}(\mathscr{P})$ into relations over $\mathscr{V}$ such that:

(a) for each $A \in \mathscr{A}$, $i_{\mathscr{D}}(A)$ is a binary relation over $\mathscr{V}$;
(b) $i_{\mathscr{D}}(I) = \{(x, x): x \in \mathscr{V}\}$;
(c) for each $t \in \mathscr{B}$, $i_{\mathscr{D}}(t)$ is a subset of $\mathscr{V}$;
(d) for each $P \in \mathscr{P}$, $i_{\mathscr{D}}(P)$ is a binary relation over $\mathscr{V}$;
(e) $i_{\mathscr{D}}(S_1; S_2) = i_{\mathscr{D}}(S_1); i_{\mathscr{D}}(S_2)$;
(f) $i_{\mathscr{D}}(S_1 \cup S_2) = i_{\mathscr{D}}(S_1) \cup i_{\mathscr{D}}(S_2)$;
(g) $(i_{\mathscr{D}}(P_1), \cdots, i_{\mathscr{D}}(P_n))$ is the $\leqq$-least n-tuple such that $(i_{\mathscr{D}}(P_1), \cdots, i_{\mathscr{D}}(P_n)) = (i_{\mathscr{D}}(S_1), \cdots, i_{\mathscr{D}}(S_n))$ holds.

The above definition is the usual denotational semantics of recursive program schemes. Its justification and equivalence with operational semantics is an immediate consequence of the results proved in [2].

Observe, for example, that if $\mathscr{D} = \{P \Leftarrow t_1; t_2\}$, then due to the convention mentioned above $i_{\mathscr{D}}(P) = i_{\mathscr{D}}(t_1)_+; i_{\mathscr{D}}(t_2)_+$.

In the sequel we shall always consider programs with respect to a given declaration scheme. We shall freely identify statements and their interpretations, hoping that no confusion will result from this.

3. Extending the state space. We now want to use the assertions from the extended space $\mathscr{V} \times \mathscr{V}$. In order to do this we have to extend (in an obvious way) several operations from $\mathscr{V}$ into $\mathscr{V} \times \mathscr{V}$. Let a, b denote subsets of $\mathscr{V} \times \mathscr{V}$ used as assertions and σ, τ elements of $\mathscr{V}$ used as a second coordinate of the extended state space. Let $R^{\uparrow} = \{((x, \sigma), (y, \sigma)): xRy \wedge \sigma \in \mathscr{V}\}$ be the extension of a program R to the space $\mathscr{V} \times \mathscr{V}$. The operations ; and $_+$ mentioned above retain their meaning when applied to subsets of $(\mathscr{V} \times \mathscr{V}) \times (\mathscr{V} \times \mathscr{V})$ and $\mathscr{V} \times \mathscr{V}$ respectively, so obviously Lemma 1 holds in the case of the extended state space $\mathscr{V} \times \mathscr{V}$. We shall use in the sequel "mixed" expressions involving assertions from $\mathscr{V} \times \mathscr{V}$ and programs from $\mathscr{V} \times \mathscr{V}$. While doing so we shall *always* mean their "extensions" to $(\mathscr{V} \times \mathscr{V}) \times (\mathscr{V} \times \mathscr{V})$, which can be obtained by attaching the subscript $_+$ to assertions and the superscript $^{\uparrow}$ to programs. For example, if we write $R_1; a; R_2$, we actually mean $R_1^{\uparrow}; a_+; R_2^{\uparrow}$. The reader should convince himself that the convention of omitting brackets (as indicated in Lemma 1) does not lead now to any ambiguities, since $(R_1; R_2)^{\uparrow} = R_1^{\uparrow}; R_2^{\uparrow}$.

Observe that $a; R \subseteq R; b$ means that $a_+; R^{\uparrow} \subseteq R^{\uparrow}; b_+$, i.e., that

$$\forall x, y, \sigma[((x, \sigma) \in a \wedge xRy) \rightarrow (y, \sigma) \in b],$$

or that the program R is partially correct with respect to a and b.

We shall need the following definition:

$$a(R) = \{(x, \sigma): \exists \tau[\sigma R \tau \wedge (x, \tau) \in a]\}.$$

In the proofs below we shall use Scott induction to prove inclusions between relations on $\mathscr{V} \times \mathscr{V}$.

Scott induction. Let $\mathscr{D} = \{P_1 \Leftarrow S_1(P_1, \cdots, P_n), \cdots, P_n \Leftarrow S_n(P_1, \cdots, P_n)\}$ be a declaration scheme. Let $\mathscr{E}_l(X_1, \cdots, X_n)$ and $\mathscr{E}_r(X_1, \cdots, X_n)$ be two expressions built up from assertions from $\mathscr{V} \times \mathscr{V}$ and programs from $\mathscr{V} \times \mathscr{V}$ and formal (place-holding) variables $X_1, \cdots, X_n$ using ; and $\cup$ and let the following two conditions be satisfied:

(i) $\mathscr{E}_l(\Omega, \cdots, \Omega) \subseteq \mathscr{E}_r(\Omega, \cdots, \Omega)$, and
(ii) for each $R_1, \cdots, R_n \subseteq \mathscr{V} \times \mathscr{V}$,
if $\mathscr{E}_l(R_1, \cdots, R_n) \subseteq \mathscr{E}_r(R_1, \cdots, R_n)$
then $\mathscr{E}_l(S_1(R_1, \cdots, R_n), \cdots, S_n(R_1, \cdots, R_n))$
$\subseteq \mathscr{E}_r(S_1(R_1, \cdots, R_n), \cdots, S_n(R_1, \cdots, R_n))$.

Then $\mathscr{E}_l(P_1, \cdots, P_n) \subseteq \mathscr{E}_r(P_1, \cdots, P_n)$.

The proof is analogous to the proof of the version formulated in [2].

4. Completeness result. The general context-free declaration scheme is

$$\{P_i \Leftarrow S_{i,1} \cup S_{i,2} \cup \cdots \cup S_{i,M_i}\}_{i=1}^{n}, \tag{1}$$

with M_i some integer $\geqq 1$, and each $S_{i,j}$, $j = 1, \cdots, M_i$, of the form

$$S_{i,j} = A(i, j, 0); P(i, j, 1); \cdots; A(i, j, K_{i,j}-1); P(i, j, K_{i,j}); A(i, j, K_{i,j}),$$

where $A(i, j, k) \in \mathscr{A} \cup \mathscr{B}$, $P(i, j, k) \in \{P_1, \cdots, P_n\}$, and $K_{i,j}$ is an integer $\geqq 0$ (if $K_{i,j} = 0$, then $S_{i,j}$ is simply $A(i, j, 0)$).

In the above declaration scheme each $P(i, j, k)$ is some element of $\{P_1, \cdots, P_n\}$. Define a function h by: $h(i, j, k) = l$ iff $P(i, j, k) = P_l$.

The general inductive assertion method calls for suitable intermediate assertions preceding and succeeding each statement in the program. The theorem presented below states soundness and completeness of a particular version of the method in which intermediate assertions from the extended state space are used. The theorem shows that the global correctness property $p; P_1 \subseteq P_1; q$ can always be established by finding intermediate assertions of the special form a^i, $a(i, j, k)$, b^i and $b(i, j, k)$.

THEOREM. *Assume the declaration scheme* (1). *For any two assertions* $p, q \subseteq \mathscr{V}$,

$$p; P_1 \subseteq P_1; q$$

iff there exist assertions $a^i, b^i \subseteq \mathscr{V} \times \mathscr{V}$ $(i \in \{1, \cdots, n\})$ *and relations* $R_{i,j,k} \subseteq \mathscr{V} \times \mathscr{V}$ $(i \in \{1, \cdots, n\}$, $j \in \{1, \cdots, M_i\}$ *and* $k \in \{1, \cdots, K_{i,j}\})$ *such that for all* $i \in \{1, \cdots, n\}$ *and* $j \in \{1, \cdots, M_i\}$,

$$\begin{cases} a^i; A(i, j, 0) \subseteq A(i, j, 0); b^i & \text{if } K_{i,j} = 0, \\ a^i; A(i, j, 0) \subseteq A(i, j, 0); a(i, j, 1), & \\ b(i, j, k); A(i, j, k) \subseteq A(i, j, k); a(i, j, k+1), \quad k = 1, \cdots, K_{i,j} - 1 & \\ b(i, j, K_{i,j}); A(i, j, K_{i,j}) \subseteq A(i, j, K_{i,j}); b^i, & \end{cases} \quad \text{if } K_{i,j} > 0, \tag{2}$$

and

$$\begin{aligned} I \cap (p \times p) &\subseteq a^1, \\ b^1 \cap (\mathscr{V} \times p) &\subseteq q \times p. \end{aligned} \tag{3}$$

Here by definition $a(i, j, k) = a^{h(i,j,k)}(R_{i,j,k})$ and $b(i, j, k) = b^{h(i,j,k)}(R_{i,j,k})$.

Proof. To make the argument more readable we shall prove the theorem in the case of the declaration $P \Leftarrow A_1; P; A_2; P; A_3 \cup A_4$. The proof for the case of the general context-free declaration scheme is analogous and we leave it to the reader. We thus prove the following.

Assume the declaration $P \Leftarrow A_1; P; A_2; P; A_3 \cup A_4$. *For any two assertions* $p, q \subseteq \mathscr{V}$,

$$p; P \subseteq P; q,$$

iff there exist assertions $a, b \subseteq \mathscr{V} \times \mathscr{V}$ *and relations* $R_1, R_2 \subseteq \mathscr{V} \times \mathscr{V}$ *such that*

$$\begin{aligned} &a; A_1 \subseteq A_1; a(R_1), \\ &b(R_1); A_2 \subseteq A_2; a(R_2), \\ &b(R_2); A_3 \subseteq A_3; b, \\ &a; A_4 \subseteq A_4; b, \end{aligned} \tag{4}$$

and

$$I \cap (p \times p) \subseteq a, \qquad b \cap (\mathcal{V} \times p) \subseteq q \times p. \tag{5}$$

If part. We first prove by Scott induction that

$$a; P \subseteq P; b. \tag{6}$$

Assume that $a; X \subseteq X; b$ for some $X \subseteq \mathcal{V} \times \mathcal{V}$, i.e., that

$$\forall x, y, \sigma[((x, \sigma) \in a \wedge xXy) \rightarrow (y, \sigma) \in b].$$

Thus for any relation R,

$$\forall x, y, \sigma, \tau[\sigma R \tau \wedge (x, \tau) \in a \wedge xXy \rightarrow (y, \tau) \in b],$$

i.e., according to our notation,

$$a(R); X \subseteq X; b(R). \tag{7}$$

Now, due to the assumptions, Lemma 1 and (7),

$$\begin{aligned} a; (A_1; X; A_2; X; A_3) &= (a; A_1); X; A_2; X; A_3 \subseteq A_1; a(R_1); X; A_2; X; A_3 \\ &\subseteq A_1; X; b(R_1); A_2; X; A_3 \subseteq A_1; X; A_2; a(R_2); X; A_3 \subseteq A_1; X; b(R_2); A_3 \\ &\subseteq (A_1; X; A_2; X; A_3); b. \end{aligned}$$

Hence, by Lemma 1 and the assumptions,

$$a; (A_1; X; A_2; X; A_3 \cup A_4) \subseteq (A_1; X; A_2; X; A_3 \cup A_4); b.$$

Since obviously $a; \Omega \subseteq \Omega; b$, by Scott induction, (6) holds.

We are now ready to prove $p; P \subseteq P; q$. Suppose that $x \in p$ and xPy for some $x, y \in \mathcal{V}$. We have to show: $y \in q$. By the assumptions $(x, x) \in a$. By (6), $(y, x) \in b$. Since $x \in p$, by the assumptions $(y, x) \in q \times p$, so $y \in q$.

Only if part. Put $a = I$, $b = \check{P}$ and let $R_1 = A_1$ and $R_2 = A_1; P; A_2$. We are to prove that (4) and (5) hold.

Let x, y, σ be arbitrary elements of $\mathcal{V}$.

(i) We have to show: $a; A_1 \subseteq A_1; a(R_1)$, i.e., $(x, \sigma) \in a$ and xA_1y implies $(y, \sigma) \in a(R_1)$, which is equivalent to $\exists \tau[\sigma R_1 \tau \wedge (y, \tau) \in a]$. Suppose $(x, \sigma) \in a$ and xA_1y. By the definition of a, $\sigma = x$, so by the definition of R_1, $\sigma R_1 y$. Hence, since $(y, y) \in a$, we get $\exists \tau[\sigma R_1 \tau \wedge (y, \tau) \in a]$ by putting $\tau = y$.

(ii) We have to show: $b(R_1); A_2 \subseteq A_2; a(R_2)$, i.e., $\exists \tau[\sigma R_1 \tau \wedge (x, \tau) \in b]$ and xA_2y implies $\exists \tau_1[\sigma R_2 \tau_1 \wedge (y, \tau_1) \in a]$. Suppose that for some $\tau, \sigma R_1 \tau$, $(x, \tau) \in b$ and xA_2y. By the definition of R_1 and $b, \sigma A_1 \tau$ and τPx, so $\sigma(A_1; P; A_2)y$. By the definition of $R_2, \sigma R_2 y$, so, since $(y, y) \in a$, we get $\exists \tau_1[\sigma R_2 \tau_1 \wedge (y, \tau_1) \in a]$ by putting $\tau_1 = y$.

(iii) We have to show: $b(R_2); A_3 \subseteq A_3; b$, i.e., $\exists \tau[\sigma R_2 \tau \wedge (x, \tau) \in b]$ and xA_3y implies $(y, \sigma) \in b$. Suppose that for some $\tau, \sigma R_2 \tau, (x, \tau) \in b$ and xA_3y. By the definition of R_2 and b, $\sigma(A_1; P; A_2)\tau$ and τPx, so $\sigma(A_1; P; A_2; P; A_3)y$. Thus, σPy, which means $(y, \sigma) \in b$.

(iv) We have to show: $a; A_4 \subseteq A_4; b$, i.e., $(x, \sigma) \in a$ and xA_4y implies $(y, \sigma) \in b$. Suppose $(x, \sigma) \in a$ and xA_4y. Then $\sigma = x$ and xPy, *i.e.*, $(y, \sigma) \in b$.

(v) Obviously $I \cap (p \times p) \subseteq a$.

(vi) We have to show: $b \cap (\mathcal{V} \times p) \subseteq q \times p$, i.e., $(x, y) \in b$ and $y \in p$ implies $x \in q$. Suppose $(x, y) \in b$ and $y \in p$. Then yPx, and since $p; P \subseteq P; q$, we find $x \in q$. This concludes the proof.

The above proof is an analogue of the corresponding completeness proofs in [5] and [6]. However, the relational approach sheds some light on the role of the auxiliary variables used in [5] to obtain so-called "most general formulas" and in [6], analogously, to "freeze" the global variables upon entering a procedure call. It is clear from the above proof that completeness is obtained by using the *meaning* of a procedure as an assertion.

The proof also suggests an alternative, equivalent point of view at the way of introducing the extended state space. Namely, the same result can be obtained by proving first partial correctness of the program $\sigma := x; P$ using assertions from its state space. The condition $I \cap (p \times p) \subseteq a$ is then replaced by the equivalent requirement $p \times \mathscr{V}; \sigma := x \subseteq \sigma := x; a$.

In such a way the extension of the state space is caused by a change in the original program P. The desired global correctness property is then derived by deleting the assignment $\sigma := x$ to the "auxiliary variable" σ using the corresponding proof rule from [8].

5. An application. Having obtained a specific form of the completeness result we shall illustrate its usefulness by the following example.

Let the state space $\mathscr{V}$ be the set of natural numbers $\mathscr{N}$. Consider the following declaration:

$$P \Leftarrow [n \leqq 100]; [n := n+11]; P; P \cup [n > 100]; [n := n-10], \tag{8}$$

where, of course, $[n \leqq 100] = \{x : x \leqq 100\}$, $[n := n+11] = \{(x, y) : y = x+11\}$ and so on. P is of course McCarthy's well-known 91 function defined in a relational framework. We want to prove that

$$[n \leqq 100]; P \subseteq P; [n = 91]. \tag{9}$$

Observe that the above declaration is of the form $P \Leftarrow A_1; P; A_2; P; A_3 \cup A_4$, where

$$A_1 = [n \leqq 100]; [n := n+11],$$

$$A_2 = I,$$

$$A_3 = I,$$

$$A_4 = [n > 100]; [n := n-10].$$

We can now use the theorem to prove (9). The easiest way to proceed is to define the required relations and functions as in the proof of the theorem, taking for P $[n \leqq 100]; [n := 91] \cup [n > 100]; [n := n-10]$, and to check that (4) and (5) hold.

Thus we define

$$a = \{(x, x) : x \in \mathscr{N}\},$$

$$b = \{(x, y) : (x = 91 \wedge y \leqq 100) \vee (x = y - 10 \wedge y > 100)\},$$

$$R_1 = \{(x, y) : x \leqq 100 \wedge y = x + 11\},$$

$$R_2 = [n \leqq 100]; [n := n+11]; ([n \leqq 100]; [n := 91] \cup [n > 100]; [n := n-10])$$

$$= \{(x, y) : (90 \leqq x \leqq 100 \wedge y = x+1) \vee (x < 90 \wedge y = 91)\}.$$

We leave the task of checking that (4) and (5) indeed hold to the reader. Now, by the theorem, (9) holds.

The above program together with the corresponding assertions can be represented by the flow-chart (Fig. 1).

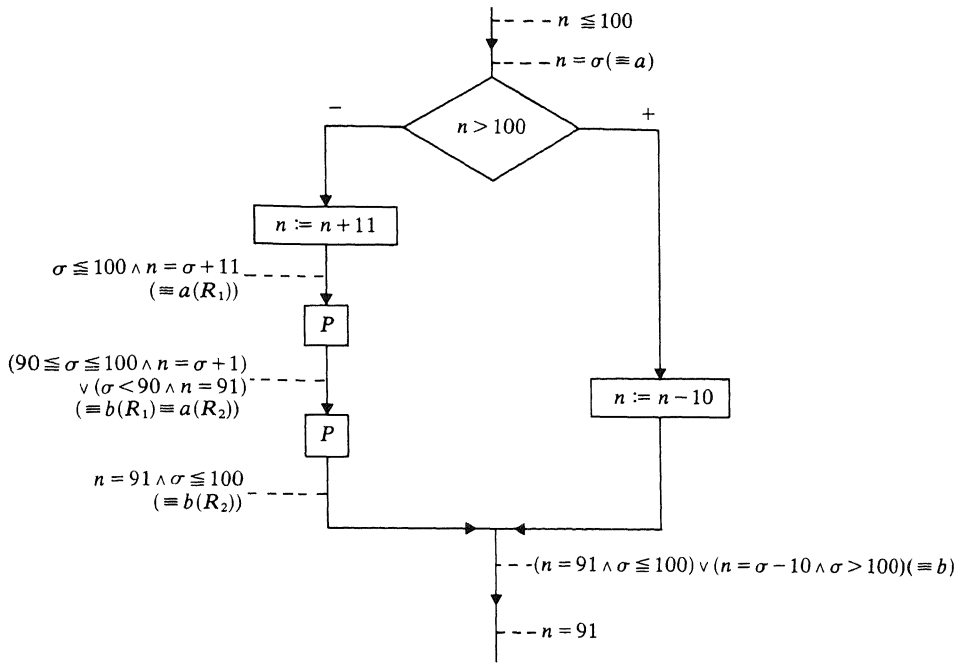

FIG. 1

Acknowledgments. We are grateful to Prof. A. Pnueli who refereed the paper and suggested various improvements, in particular the present, stronger version of the completeness result. We also thank Prof. J. W. de Bakker for critical comments on an earlier version.

REFERENCES

[1] K. R. APT, J. A. BERGSTRA AND L. G. L. T. MEERTENS, *Recursive assertions are not enough—or are they?*, Theor. Comput. Sci., 8 (1979), pp. 73–87.

[2] J. W. DE BAKKER AND L. G. L. T. MEERTENS, *On the completeness of the inductive assertion method*, J. Comput. System Sci., 11 (1975), pp. 323–357.

[3] S. A. COOK, *Soundness and completeness of an axiom system for program verification*, this Journal, 7 (1978), pp. 70–90.

[4] J. H. GALLIER, *Semantics and correctness of nondeterministic flowchart programs with recursive procedures*, Proc. 5th Coll. Automata, Languages and Programming, Lecture Notes in Computer Science, No. 62, Springer-Verlag, 1978, pp. 251–267.

[5] G. A. GORELICK, *A complete axiomatic system for proving assertions about recursive and nonrecursive programs*, Technical Report No. 75, University of Toronto (1975).

[6] D. HAREL, A. PNUELI AND J. STAVI, *Completeness issues for inductive assertions and Hoare's method*, Technical Report, Tel-Aviv University, 1976.

[7] C. A. R. HOARE, *An axiomatic basis for programming language constructs*, Comm. ACM, 12 (1969), pp. 576–580.

[8] S. OWICKI AND D. GRIES, *An axiomatic proof technique for parallel programs I*, Acta Informatica, 6 (1976), pp. 319–340.