AFDELING INFORMATICA IW 84/77 SEPTEMBER
(DEPARTMENT OF COMPUTER SCIENCE)

K.R. APT & L.G.L.T. MEERTENS

COMPLETENESS WITH FINITE SYSTEMS OF INTERMEDIATE ASSERTIONS FOR RECURSIVE PROGRAM SCHEMES

Preprint

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 68A05

ACM-Computing Reviews-categories: 5.24

Completeness with finite systems of intermediate assertions for recursive program schemes *)

by

K.R. Apt & L.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 an extended state space. This result contrasts with the results of DE BAKKER & MEERTENS [1], where it is proved that if assertions are taken from the original state space V, then in the general case an *infinite* system of intermediate assertions is needed. In the case of functional schemes (where any deterministic scheme is a functional one) one can take $V \times V$ for the extended state space, thus obtaining a semantical counterpart of the use of auxiliary variables.

KEY WORDS & PHRASES: *partial correctness, intermediate assertions, relational framework, extended state space, recursive program schemes.*

*) This report will be submitted for publication elsewhere.

1. INTRODUCTION

In DE BAKKER & MEERTENS [1] it was shown 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 [4] extended the results of COOK [2] and obtained a completeness result for a Hoare-like axiomatic system (see HOARE [5]) 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. The purpose of this paper is to investigate this apparent contradiction.

The solution lies in the fact that Gorelick uses auxiliary variables (to store the initial values of variables) which have no semantical counterpart in the relational framework of [1]. Semantically, the use of auxiliary variables corresponds to the use of states which have an additional coordinate (from a space W) inaccessible to a program. We shall call the domain of such states an extended state space.

We prove that if one allows intermediate assertions from an extended state space, 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 an extended state space can be found. For the space W one can take the powerset of the original state space V. (Theorem 4.4 of [1] shows that for W one could also take the set of all so called index-triple sequences.) In the case of "functional" declaration schemes one can take for W the original state space V. Informally speaking, in the general case programs are viewed as predicate transformers, whereas in the case of functional schemes they are viewed as state transformers.

In [1] 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. 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 clearly not the case with Hoare-like axiomatic systems, where assertions are formulas from an assertion language. 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 which formulas (subsets) should be accepted as assertions. We shall not pursue here this matter further, leaving it as a subject of another paper.

2. PRELIMINARIES

As in DE BAKKER & MEERTENS [1] we shall use binary relations over the state space to provide an interpretation for systems of mutually recursive procedures. More precisely, given a set $\mathcal{P} = \{P_1,\dots,P_n\}$ of procedure symbols, we define a language of "statements" $\mathcal{S}(\mathcal{P})$ as follows: let $\mathcal{A} = \{A_1,A_2,\dots\}$ be a set of "elementary action" symbols, $\mathcal{B} = \{t_1,t_2,\dots\}$ a set of "Boolean expressions". $\mathcal{S}(\mathcal{P})$ is then the least set containing $\mathcal{A} \cup \mathcal{B} \cup \mathcal{P}$ that is closed under the operations ; (sequencing) and $\cup$ (nondeterministic choice).

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

In [1] 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 [1] which are used below.

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

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

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

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

$I = \{(x,x): x \in \mathcal{V}\}$

Ω denotes the empty set.

Throughout the paper we use the convention from [1] 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 [1].

<u>LEMMA 1</u>. (i) $(R_1;R_2);R_3 = R_1;(R_2;R_3)$ (= $R_1;R_2;R_3$, 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 $X_1,\ldots,X_n$, $Y_1,\ldots,Y_n$ are subsets of $\mathcal{V} \times \mathcal{V}$ then by definition $(X_1,\ldots,X_n) \leq (Y_1,\ldots,Y_n)$ iff $X_i \subseteq Y_i$ for $i = 1,\ldots,n$.
$\leq$ is a partial ordering.

Let $\mathcal{D} = \{P_1 \Leftarrow S_1,\ldots,P_n \Leftarrow S_n\}$ be a declaration scheme. By an *interpretation* $i_{\mathcal{D}}$ into a state space $\mathcal{V}$ we mean a mapping from $\mathcal{S}$ into relations over $\mathcal{V}$ such that for each $A \in \mathcal{A}$ $i_{\mathcal{D}}(A)$ is a binary relation over $\mathcal{V}$, for each $t \in \mathcal{B}$ $i_{\mathcal{D}}(t)$ is a subset of $\mathcal{V}$, for each $P \in \mathcal{P}$ $i_{\mathcal{D}}(P)$ is a binary relation over $\mathcal{V}$ and

(a) $$i_{\mathcal{D}}(S_1;S_2) = i_{\mathcal{D}}(S_1);i_{\mathcal{D}}(S_2)$$

(b) $$i_{\mathcal{D}}(S_1 \cup S_2) = i_{\mathcal{D}}(S_1) \cup i_{\mathcal{D}}(S_2)$$

(c) $(i_{\mathcal{D}}(P_1),\ldots,i_{\mathcal{D}}(P_n))$ is the $\leq$-least n-tuple such that

$$(i_{\mathcal{D}}(P_1),\ldots,i_{\mathcal{D}}(P_n)) = (i_{\mathcal{D}}(S_1),\ldots,i_{\mathcal{D}}(S_n)) \text{ holds}$$

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 [1].

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

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 want now to use the assertions from an extended space $\mathcal{V}\times\mathcal{W}$. In order to do this we have to extend (in an obvious way) several operations from $\mathcal{V}$ into $\mathcal{V}\times\mathcal{W}$. Let U,V denote subsets of $(\mathcal{V}\times\mathcal{W}) \times (\mathcal{V}\times\mathcal{W})$, a,b,c subsets of $\mathcal{V}\times\mathcal{W}$, $\sigma,\sigma',\ldots$ elements of $\mathcal{W}$, f,g,h partial functions from $\mathcal{W}$ into $\mathcal{W}$. Let $R^{\uparrow} = \{((x,\sigma),(y,\sigma)) \colon xRy \wedge \sigma \in \mathcal{W}\}$ be the extension of a program R to the space $\mathcal{V}\times\mathcal{W}$. The operations ; and $_+$ mentioned above retain their meaning when applied to subsets of $(\mathcal{V}\times\mathcal{W}) \times (\mathcal{V}\times\mathcal{W})$ and $\mathcal{V}\times\mathcal{W}$, so obviously Lemma 1 holds in the case of the extended state space $\mathcal{V}\times\mathcal{W}$. We shall use in the sequel "mixed" expressions involving assertions from $\mathcal{V}\times\mathcal{W}$ and programs from $\mathcal{V}\times\mathcal{V}$. While doing so we shall always mean their "extensions" to $(\mathcal{V}\times\mathcal{W}) \times (\mathcal{V}\times\mathcal{W})$, 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 (from Lemma 1) does not lead now to any ambiguities, since $(R_1;R_2)^{\uparrow} = R_1^{\uparrow};R_2^{\uparrow}$.

For any two sets A and B a subset f of A×B is called a function if

$$\forall x,y,z[((x,y) \in f \wedge (x,z) \in f) \rightarrow y=z].$$

If f is a function then $\mathrm{dom}(f) = \{x\colon \exists y((x,y) \in f)\}$. For $x \in \mathrm{dom}(f)$ we denote the unique y such that $(x,y) \in f$ by f(x).

We write $f\colon A \xrightarrow[\text{part}]{} B$ to denote the fact that f is a subset of A×B which is a function. If $f\colon A \xrightarrow[\text{part}]{} B$ and $\mathrm{dom}(f) = A$ we write $f\colon A \rightarrow B$.

We define

$$a(f) = \{(x,\sigma)\colon \sigma \in \mathrm{dom}(f) \wedge (x,f(\sigma)) \in a\},$$
$$a[\sigma] = \{x\colon (x,\sigma) \in a\}.$$

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

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

Obviously we have

$$a;R \subseteq R;b \quad \text{iff for all } \sigma\epsilon\mathcal{W} \quad a[\sigma];R \subseteq R;b[\sigma], \tag{*}$$

so correctness of a program with respect to assertions from the extended state space can be proved by a simple reduction to the original state space.

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

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

(i) $\mathcal{E}_\ell(\Omega,\ldots,\Omega) \subseteq \mathcal{E}_r(\Omega,\ldots,\Omega)$, and

(ii) for each $R_1,\ldots,R_n \subseteq \mathcal{V}\times\mathcal{V}$
if $\mathcal{E}_\ell(R_1,\ldots,R_n) \subseteq \mathcal{E}_r(R_1,\ldots,R_n)$
then $\mathcal{E}_\ell(S_1(R_1,\ldots,R_n),\ldots,S_n(R_1,\ldots,R_n)) \subseteq$
$\mathcal{E}_r(S_1(R_1,\ldots,R_n),\ldots,S_n(R_1,\ldots,R_n))$.

Then $\mathcal{E}_\ell(P_1,\ldots,P_n) \subseteq \mathcal{E}_r(P_1,\ldots,P_n)$.

The proof is analogous to the proof of the version formulated in [1].

4. COMPLETENESS RESULTS

(i) *The general case.*

The general form of the theorem we want to prove threatens to obscure its basic simplicity because of the heavy use of indices that is needed. We shall therefore state our theorem by means of an example. It should be clear how to extend it to the general case of an arbitrary context-free declaration scheme.

THEOREM 1. *Assume the declaration* $P \Leftarrow A_1;P;A_2;P;A_3 \cup A_4$. *For any two assertions* $p,q \subseteq V$

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

iff there exists a set W, *assertions* $a,b \subseteq V\times W$, *a set* $u \subseteq W$ *and functions* $f\colon W \xrightarrow[\text{part}]{} W$, $g\colon W \xrightarrow[\text{part}]{} W$ *and* $\alpha\colon\ p \to u$ *such that*

$$\begin{aligned} &a;A_1 \subseteq A_1;a(f),\\ &b(f);A_2 \subseteq A_2;a(g),\\ &b(g);A_3 \subseteq A_3;b,\\ &a;A_4 \subseteq A_4;b \end{aligned} \tag{1}$$

and

$$\begin{aligned} &\alpha \subseteq a,\\ &b \cap (V\times u) \subseteq q\times u. \end{aligned} \tag{2}$$

PROOF. *If part.*
We first prove by Scott induction that

$$a;P \subseteq P;b. \tag{3}$$

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

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

Thus for any function $h\colon W \xrightarrow[\text{part}]{} W$

$$\forall x,y,\sigma[(\sigma \in \mathrm{dom}(h) \wedge (x,h(\sigma))\in a \wedge xXy) \to (y,h(\sigma))\in b],$$

i.e. according to our notation,

$$a(h);X \subseteq X;b(h). \tag{4}$$

Now, due to the assumptions, Lemma 1 and (4),

$$a;(A_1;X;A_2;X;A_3) = (a;A_1);X;A_2;X;A_3 \subseteq A_1;a(f);X;A_2;X;A_3 \subseteq$$
$$A_1;X;b(f);A_2;X;A_3 \subseteq A_1;X;A_2;a(g);X;A_3 \subseteq A_1;X;A_2;b(g);A_3 \subseteq$$
$$(A_1;X;A_2;X;A_3);b.$$

, by Lemma 1 and the assumptions,

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

obviously $a;\Omega \subseteq \Omega;b$, by Scott induction (3) holds.

We are now ready to prove $p;P \subseteq P;q$. Suppose that $x \epsilon p$ and xPy for $x,y \in \mathcal{V}$. We have to show: $y \epsilon q$. By the assumptions $x \in \mathrm{dom}(\alpha)$, so (x) is defined and $(x,\sigma) \in a$. By (3), $(y,\sigma) \in b$. Since $\sigma \epsilon u$, by the as-tions $(y,\sigma) \in q \times u$, so $y \epsilon q$.

Only if part.

We have to establish the existence of $\mathcal{W}$,a,b,u,f,g and α satisfying (1) (2).

$\mathcal{W} = \{r\colon r \subseteq \mathcal{V}\}$, the powerset of $\mathcal{V}$. Define

$$a = \{(x,r)\colon x \in r,\ r \subseteq \mathcal{V}\},$$
$$b = \{(x,r)\colon x \in r \circ P,\ r \subseteq \mathcal{V}\},$$
$$u = \{p\}$$

let

$$f = \{(r, r \circ A_1)\colon r \subseteq \mathcal{V}\},$$
$$g = \{(r, r \circ (A_1;P;A_2))\colon r \subseteq \mathcal{V}\},$$
$$\alpha = p \times \{p\}.$$

rve that for any $r \subseteq \mathcal{V}$ and $h\colon \mathcal{W} \to \mathcal{W}$ $a[r]=r$, $b[r]=r \circ P$, $[r] = h(r)$ and $b(h)[r] = h(r) \circ P$.

Hence, for any $r \subseteq \mathcal{V}$:

$$a[r];A_1 = r;A_1 \subseteq A_1;r \circ A_1 = A_1;f(r) = A_1;a(f)[r],$$

$$b(f)[r];A_2 = f(r) \circ P;A_2 = (r \circ A_1) \circ P;A_2 = r \circ (A_1;P);A_2 \subseteq$$
$$A_2;(r \circ (A_1;P) \circ A_2) = A_2;r \circ (A_1;P;A_2) = A_2;g(r) =$$
$$A_2;a(g)[r],$$

$$\begin{aligned} b(g)[r];A_3 &= g(r)\circ P;A_3 \subseteq A_3;(g(r)\circ P)\circ A_3 = \\ &A_3;(r\circ(A_1;P;A_2)\circ P)\circ A_3 = A_3;r\circ(A_1;P;A_2;P;A_3) \subseteq \\ &A_3;r\circ P = A_3;b[r], \end{aligned}$$

$$a[r];A_4 = r;A_4 \subseteq A_4;r\circ A_4 \subseteq A_4;r\circ P = A_4;b[r].$$

So by (*) (1) holds.

By the definition of a, $\alpha = p\times\{p\} \subseteq a$. By the definition of b, $b \cap (V\times\{p\}) = \{(x,p): x \in p\circ P\}$. But by the assumption $p;P \subseteq P;q$, so $p\circ P \subseteq q$. So $\{(x,p): x \in p\circ P\} \subseteq q\times\{\dot{p}\}$.

This concludes the proof of Theorem 1.

The above theorem should not be too surprising to the diligent reader of DE BAKKER & MEERTENS [1]. A similar fact is there stated as 5.3 on page 354. In [1], however, assertions are always subsets of the original state space, thus the above system {a,a(f), a(g), b, b(f), b(g)} from the extended state space corresponds to the infinite system $\{a[\sigma], b[\sigma]\}_{\sigma \in W}$ from the original state space. Observe also that the proof of the "only if part" of 5.3 is much more difficult--the required space W is the space T of the index-triple sequences. On the other hand, T is always a countable set, which is not the case if we take for D the set $\{r: r \subseteq V\}$ (unless V is finite), as we did in the proof of the "only if part". In some situations, however, we can simply take for W the state space V.

(ii) *The case of functional declaration schemes.*

The reader should be warned that if $W=V$, the construct a;R becomes ambiguous, its meaning being either $\{(x,y): \exists z[(x,z) \in a \wedge zRy]\}$ or $a_+;R^\dagger$. In the considerations below we always mean the latter by imposing the convention that letters a,b (with possible subscripts) *always* mean assertions.

Assume an arbitrary context-free declaration scheme

$$\{P_i \Leftarrow S_{i,1} \cup S_{i,2} \cup \ldots \cup S_{i,M_i}\}_{i=1}^{n}, \tag{5}$$

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

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

with elementary actions $A(i,j,k)$, $P(i,j,k) \in \{P_1,\dots,P_n\}$, and $K_{i,j}$ an integer ≥ 0 (if $K_{i,j}=0$ then $S_{i,j}$ is simply $A(i,j,0)$).

DEFINITION 1. The scheme (5) is called *functional* if
i) for all $i=1,\dots,n$, $j=1,\dots,M_i$ and $k=0,\dots,K_{i,j}$ $A(i,j,k)$ is a function.
ii) each P_i $(i=1,\dots,n)$ is a function.

Observe that the condition ii) is clearly fulfilled if i) holds and for each $i=1,\dots,n$ and j_1,j_2 such that $1 \leq j_1 < j_2 \leq M_i$

$$\mathrm{dom}(A(i,j_1,0)) \cap \mathrm{dom}(A(i,j_2,0)) = \emptyset$$

We shall now prove that in the case of functional declaration schemes we can take $V\times V$ as an extended state space. Again we prove it by means of an example, leaving the general case to the reader.

THEOREM 2. *Suppose that* $P \Leftarrow A_2;P;A_2;P;A_3 \cup A_4$ *is a functional declaration scheme. For any two assertions* $p,q \subseteq V$

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

iff there exist assertions $a,b \subseteq V\times V$, *a set* $u \subseteq V$ *and functions* $f\colon V \xrightarrow[\text{part}]{} V$, $g\colon V \xrightarrow[\text{part}]{} V$ *and* $\alpha\colon p \to u$ *such that* (1) *and* (2) *hold.*

PROOF. The "if part" is an immediate consequence of Theorem 1. To prove the "only if part", put $a = \{(x,x)\colon x \in V\}$, $b = \{(y,x)\colon xPy\}$, $u=p$ and let $f=A_1$, $g = A_1;P;A_2$ and $\alpha = \{(x,x)\colon x \in p\}$.

Let x,y,σ be arbitrary elements of V.

(i) We have to show: $a;A_1 \subseteq A_1;a(f)$, i.e., $(x,\sigma) \in a$ and xA_1y implies $(y,\sigma) \in a(f)$, which is equivalent to $\sigma \in \mathrm{dom}(f)$ and $(y,f(\sigma)) \in a$. Suppose $(x,\sigma) \in a$ and xA_1y. By the definition of a $x=\sigma$, so, since $f = A_1$, $\sigma \in \mathrm{dom}(f)$, $y = f(x) = f(\sigma)$ and $(y,f(\sigma)) \in a$.

(ii) We have to show: $b(f);A_2 \subseteq A_2;a(g)$, i.e., $\sigma \in \mathrm{dom}(f)$, $(x,f(\sigma)) \in b$ and xA_2y implies $\sigma \in \mathrm{dom}(g)$ and $(y,g(\sigma)) \in a$.

Suppose $\sigma \in \mathrm{dom}(f)$, $(x,f(\sigma)) \in b$ and xA_2y. Then $f(\sigma)Px$. Since $\sigma A_1 f(\sigma)$

we find $\sigma(A_1;P;A_2)y$, so $\sigma \in \mathrm{dom}(g)$ and $y = g(\sigma)$, i.e., $(y,g(\sigma)) \in a$.

(iii) We have to show: $b(g);A_3 \subseteq A_3;b$, i.e., $\sigma \in \mathrm{dom}(g)$, $(x,g(\sigma)) \in b$ and xA_3y implies $(y,\sigma) \in b$. Suppose $\sigma \in \mathrm{dom}(g)$, $(x,g(\sigma)) \in b$ and xA_3y. Then $g(\sigma)Px$, i.e., by the definition of g $\sigma(A_1;P;A_2;P)x$ and so $\sigma(A_1;P;A_2;P;A_3)y$. Thus, σPy, which means that $(y,\sigma) \in b$.

(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 $x=\sigma$ and xPy, i.e., $(y,\sigma) \in b$.

(v) Obviously $\alpha \subseteq a$.

(vi) We have to show: $b \cap (V \times u) \subseteq q \times u$, i.e., $(x,y) \in b$ and $y \in p$ implies $x \in q$. Suppose $(x,y) \in b$ and $y \in p$. Then, $(y,x) \in p$, and since $p;P \subseteq p;q$, we find $x \in q$.

This concludes the proof.

REMARK. In most cases one deals with deterministic programs which can be represented by means of functional declaration schemes. A notable exception is Dijkstra's guarded command construct (see DIJKSTRA [3]), which gives rise to programs of bounded determinacy. It is easy to see that in the case of declaration schemes of bounded determinacy we can take for W the set of all finite subsets of V. The proof is analogous to the proof of the "only if part" of Theorem 1.

5. AN APPLICATION

Having obtained specific forms of completeness results we shall illustrate their usefulness by the following example.

Let the state space V be the set of natural numbers N. Consider the following declaration

$$P \Leftarrow [n \leq 100];[n := n+11];P;P \cup [n > 100];[n := n-10] \qquad (6)$$

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

$$[n \leq 100];P \subseteq P;[n=91]. \tag{7}$$

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

$$\begin{aligned} A_1 &= [n \leq 100];[n := n+11] \\ A_2 &= I \\ A_3 &= I \\ A_4 &= [n > 100];[n := n-10] \end{aligned}$$

Since (6) is a functional declaration scheme, we can use Theorem 2 to prove (7). The easiest way to proceed is to define the required relations and functions as in the proof of Theorem 2, taking for P $[n \leq 100];[n := 91] \cup [n > 100];[n := n-10]$, and to check that (1) and (2) hold.

Thus we define:

$$\begin{aligned} a &= \{(x,x)\colon x \in N\}; \\ b &= \{(x,y)\colon (x=91 \wedge y \leq 100) \vee (x=y-10 \wedge y>100)\}; \\ u &= \{x\colon x \leq 100\}; \\ f &= \{(x,y)\colon x \leq 100 \wedge y = x+11\}; \\ g &= [n \leq 100];[n := n+11];([n \leq 100];[n := 91] \cup [n > 100];[n := n-10 \\ &= \{(x,y)\colon (90 \leq x \leq 100 \wedge y=x+1) \vee (x<90 \wedge y=91)\}; \\ \alpha &= \{(x,x)\colon x \leq 100\}. \end{aligned}$$

We leave the task of checking that (1) and (2) indeed hold to the reader. Now, by Theorem 2, (7) holds.

ACKNOWLEDGEMENT

We are grateful to J.W. de Bakker for critical comments on an earlier version.

REFERENCES

[1] DE BAKKER, J.W. & L.G.L.T. MEERTENS, *On the Completeness of the Inductive Assertion Method*, Journal of Computer and System Sciences, vol. 11, No. 3, pp. 323-357 (1975).

[2] COOK, S.A., *Axiomatic and Interpretive Semantics for an Algol Fragment*, Technical Report no. 79, University of Toronto (1975).

[3] DIJKSTRA, E.W., *A Discipline of Programming*, Prentice-Hall, Inc., London (1976).

[4] GORELICK, G.A., *A Complete Axiomatic System for Proving Assertions about Recursive and Non-Recursive Programs*, Technical Report no. 75, University of Toronto (1975).

[5] HOARE, C.A.R., *An Axiomatic Basis for Programming Language Constructs*, C. ACM 12, pp. 576-580 (1969).