Theoretical Computer Science 13 (1981) 239–270
North-Holland Publishing Company

NONDETERMINISTIC FLOWCHART PROGRAMS WITH RECURSIVE PROCEDURES: SEMANTICS AND CORRECTNESS II*

Jean H. GALLIER

Department of Computer and Information Sciences, The Moore School of Electrical Engineering D2, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.

Communicated by M. Nivat
Received March 1979
Revised December 1979

Abstract. In the second part of this work, we formulate a new inductive assertion method applying to the class of nondeterministic flowchart programs with recursive procedures studied in part 1. Using results on unfolding proved in part 1, we prove that this method is sound and complete with a *finite* number of assertions. We study four notions of correctness: two notions of partial correctness (existential and universal) and the corresponding notions of total correctness. We also formalize two notions of extension and equivalence (existential and universal) in the second-order predicate calculus.

6. Introduction

In part 1 of this work[1], we have studied the semantics of a class of nondeterministic flowchart programs with recursive procedures. The main result established in part 1 states that every nondeterministic flowchart program with recursive procedures can be unfolded into a semantically equivalent infinite flowchart without procedures. This result is used in part 2 of this work to prove the soundness of an inductive assertion method applying to the class of programs defined and studied in part 1. We also prove that this verification method is complete with a finite number of assertions (contrary to De Bakker and Meertens's method [7]).

6.1. The inductive assertion method

On the surface, there appears to be some disagreement in the literature on the possibility of a complete inductive assertion method for Algol-like programs with recursive procedures. On the one hand, in their fundamental paper [7], De Bakker and Meertens formulate an inductive assertion method for a class of monadic

* This research was supported by the National Science Foundation under Grant No. DCR-74-15901 and MCS-77-11360.

[1] See Theoretical Computer Science 13(2) (1981) 193–223.

parameterless recursive programs and prove its soundness and completeness, provided an infinite pattern of assertions is used. They also exhibit an example showing that an infinite number of assertions is actually necessary. On the other hand, using a different model Greibach [14] suggests a method with a finite number of assertions which is claimed, without proof, to be sound and complete. In this paper, expanding Greibach's idea [14], we formulate an extension of the inductive assertion method with a finite number of assertions and prove its soundness and completeness. In the model of De Bakker and Meertens, programs may use only one variable and assertions or predicates too are required to be monadic. Hence, in their system, one cannot use assertions relating the different values of the program variable at different stages of the computation. A model which allows any finite number of variables in the program and in the assertions allows one to store intermediate values and to write assertions relating the values of variables at different points in the computation, and this is why a finite number of assertions can be sufficient in such a model. To the best of our knowledge, this is the first time that such a method is formulated and justified rigorously for nondeterministic flowchart programs with recursive procedures (see the paragraph below for further comments).

Our generalization of the inductive assertion method to nondeterministic programs has been greatly influenced by the work of Ashcroft and Manna [3] and Manna [28, 30, 31]. In particular, our derivation of the verification conditions follows the backward substitution method presented in [28, 31]. The influence of De Bakker and Meertens [7] and Greibach [14] is also evident and indeed, the results of this paper could not have been obtained without their prior work.

This work has benefited from the following studies of the inductive assertion method: [3, 9, 10, 29, 30, 31] on flowchart programs without procedures but possibly nondeterminism; [6, 7] on monadic recursive programs; [14] on deterministic flowchart programs with procedures.

Related investigations on formal systems for proving properties of programs have been made by Apt and De Bakker [1], Apt and Meertens [2], Clarke [4, 5], Gorelick [13], Harel [17, 18, 19, 20], Harel and Pratt [21], Harel, Meyer and Pratt [22], Harel, Pnueli and Stavi [23, 24], Lipton [27] and Pratt [35]. However, except for [24] where an inductive assertion method is studied and [20] which is discussed below, the above papers differ in many respects from our work and we now clarify some of these differences.

First, the papers listed above are concerned with the soundness and relative completeness of deductive systems and do not study an inductive assertion method in the sense of [28] or [7]. Second, they deal with structured programs where no GOTO instructions are allowed, and none of these papers deals with a class of programs as general as ours where *both* recursion and unrestricted nondeterminism are allowed. In fact, only Harel, Pnueli and Stavi [23, 24] allow nondeterminism via relational assignments, but they do not allow nondeterministic choice points. Our work is also more general since we study two forms of partial correctness due to Manna [31], existential correctness and universal correctness, which allows us to give a thorough

treatment of total correctness as well. Third and perhaps more essential, relative completeness results obtained for deductive systems and our completeness result obtained for an inductive assertion method are not equivalent, since the concept of completeness is not the same in the two cases.

Harel [20] contains a thorough and very interesting study of nondeterministic recursive programs in the framework of Dynamic Logic. Many concepts of correctness are studied and existential and universal correctness are expressible in dynamic logic. Since Harel [20] does not study explicitly an inductive assertion method, it is difficult to say whether the soundness and completeness results found in his work have a direct bearing on the results presented here. Matters are further complicated by the fact that in Harel's model, procedures are *parameterless* and do not appear to have local variables, which in our opinion might cause substantial differences. In any case, we think that it is premature to make any judgement until a thorough comparison is made.

We believe that the method presented here is a rather natural and simple extension of Floyd's method to programs with recursive procedures, the main new ingredients being the introduction of input and output assertions characterizing each procedure and of special assertions around procedure calls similar to the usual loop invariants. Rules such as the rule of recursion and the rule of adaptation [25, 26] are not necessary (although implicit in the proofs) and, in our opinion, this is a significant advantage of our method over the axiomatic approach.

Resulting from discussions with A. Pnueli and D. Harel, it appears that the suggestion that additional variables are necessary to guarantee the completeness of a verification method in the presence of recursion has first been made by Greibach [14] (at least in writing). Variants of this idea involving the notions of auxiliary variables or freezing of variables are crucial to most of the papers listed above.

We now describe briefly the contents of part 2. In Section 6.2 we review the familiar concepts of partial correctness, total correctness and termination. Following [30], we note that in the case of nondeterministic programs, the concept of partial correctness can be refined since, for any input, the execution tree may contain more than one success path. For simplicity, we only consider the two extreme cases of existential correctness and universal correctness. Existential correctness consists in requiring that some path in the execution tree be correct while universal correctness requires all paths in the execution tree to be correct for all inputs.

In Section 6.3 we present the algorithm to construct what we call the correctness formulae. Intuitively, given an input predicate ϕ and an output predicate ψ, the correctness formulae are first-order predicate formulae constructed from the function and predicate symbols occurring in the underlying scheme of the program and also containing a finite collection of unknown predicate symbols called inductive assertions. The fundamental properties of the correctness formulae are the following:

(1) If there exists an assignment of predicates to the unknown predicate symbols making the formula true for all inputs in the domain of the interpretation $\mathscr{I}$, then the program is partially correct with respect to ϕ and ψ.

We usually call this property the *soundness* or *consistency* of the method.

(2) If the program is partially correct with respect to ϕ and ψ, then there exists an assignment of predicates to the unknown predicate symbols making the formula true for all inputs in the domain of the interpretation $\mathscr{I}$.

This is usually referred to as the *completeness* of the method.

In Section 7 we prove the soundness of our method. In Section 8 we establish the completeness of the method by verifying that a particular choice of assertions makes the formulae true. In Section 9 we derive a number of other results on total correctness, termination, extension and equivalence from the main theorem of the previous section.

6.2. *Existential and universal correctness*

Let $(\alpha, \mathscr{I})$ be a program, where $\alpha = (\alpha_0, \ldots, \alpha_N)$ is a scheme with main procedure α_0 and $\mathscr{I}$ is an interpretation. Let $\bar{x}$ be the input variables of α_0 and $\bar{z}$ be the output variables. Let $\phi(\bar{x})$ be a predicate called an input predicate and $\psi(\bar{x}, \bar{z})$ be a predicate called an output predicate. Recall that for a SUCCESS execution path π and an input $\bar{\xi}$ we let $\pi(\bar{\xi})$ denote the output associated with path π for input $\bar{\xi}$. Following [30], we define partial existential correctness and partial universal correctness as follows.

Definition 6.2.1. (i) A program $(\alpha, \mathscr{I})$ is *partially* $\exists$*-correct* for input $\bar{\xi}$ with respect to ϕ and ψ if the truth of $\phi(\bar{\xi})$ implies that either there exists an infinite execution path for input $\bar{\xi}$, or a FAILURE execution path for input $\bar{\xi}$, or a SUCCESS execution path π such that $\psi(\bar{\xi}, \bar{\eta})$ holds for the output state $\bar{\eta} = \pi(\bar{\xi})$. This may be abbreviated as $pc^{\exists}(\phi, (\alpha, \mathscr{I}), \psi)(\bar{\xi})$.

(ii) A program $(\alpha, \mathscr{I})$ is *partially* $\forall$*-correct* for input $\bar{\xi}$ with respect to ϕ and ψ if the truth of $\phi(\bar{\xi})$ implies that, for all SUCCESS execution paths π for input $\bar{\xi}$, $\psi(\bar{\xi}, \bar{\eta})$ holds for the output state $\bar{\eta} = \pi(\bar{\xi})$. This may be abbreviated as $pc^{\forall}(\phi, (\alpha, \mathscr{I}), \psi)(\bar{\xi})$.

We also define total existential correctness and total universal correctness in the following way.

Definition 6.2.2. (i) A program $(\alpha, \mathscr{I})$ is *totally* $\exists$*-correct* for input $\bar{\xi}$ with respect to ϕ and ψ if the truth of $\phi(\bar{\xi})$ implies that there exists a SUCCESS execution path for input $\bar{\xi}$ and $\psi(\bar{\xi}, \bar{\eta})$ holds for the output state $\bar{\eta} = \pi(\bar{\xi})$.

(ii) A program $(\alpha, \mathscr{I})$ is *totally* $\forall$*-correct* for input $\bar{\xi}$ with respect to ϕ and ψ if the truth of $\phi(\bar{\xi})$ implies that every execution path π for input $\bar{\xi}$ is a SUCCESS path and for every such path π, $\psi(\bar{\xi}, \bar{\eta})$ holds for the output $\bar{\eta} = \pi(\bar{\xi})$.

Finally, we define weak termination and strong termination with respect to input predicate ϕ.

Definition 6.2.3. (i) A program $(\alpha, \mathcal{I})$ *weakly terminates* for input $\bar{\xi}$ with respect to ϕ if the truth of $\phi(\bar{\xi})$ implies the existence of a SUCCESS execution path for input $\bar{\xi}$.

(ii) A program $(\alpha, \mathcal{I})$ *strongly terminates* for input $\bar{\xi}$ with respect to ϕ if the truth of $\phi(\bar{\xi})$ implies that all execution paths for input $\bar{\xi}$ are SUCCESS paths.

The previous definitions were stated relative to a fixed input $\bar{\xi}$. We also have the following definitions of partial correctness and total correctness for all input in the domain of an interpretation $\mathcal{I}$.

Definition 6.2.4. Let $(\alpha, \mathcal{I})$ be a program, $\phi(\bar{x})$ be an input predicate and $\psi(\bar{x}, \bar{z})$ be an output predicate. We have the following definitions:

(1) $(\alpha, \mathcal{I})$ is *partially* $\exists$*-correct* with respect to ϕ and ψ if for all input $\bar{\xi}$ in the domain of $\mathcal{I}$, $(\alpha, \mathcal{I})$ is partially $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ.

(2) $(\alpha, \mathcal{I})$ is *partially* $\forall$*-correct* with respect to ϕ and ψ if for all input $\bar{\xi}$ in the domain of $\mathcal{I}$, $(\alpha, \mathcal{I})$ is partially $\forall$-correct for input $\bar{\xi}$ with respect to ϕ and ψ.

(3) $(\alpha, \mathcal{I})$ is *totally* $\exists$*-correct* with respect to ϕ and ψ if for all input $\bar{\xi}$ in the domain of $\mathcal{I}$, $(\alpha, \mathcal{I})$ is totally $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ.

(4) $(\alpha, \mathcal{I})$ is *totally* $\forall$*-correct* with respect to ϕ and ψ if for all input $\bar{\xi}$ in the domain of $\mathcal{I}$, $(\alpha, \mathcal{I})$ is totally $\forall$-correct for input $\bar{\xi}$ with respect to ϕ and ψ.

The following lemma relating partial correctness, total correctness and termination has been shown in [30].

Lemma 6.2.5. *Let $(\alpha, \mathcal{I})$ be a program, $\phi(\bar{x})$ be an input predicate and $\psi(\bar{x}, \bar{z})$ be an output predicate. The following holds:*

(1) $(\alpha, \mathcal{I})$ is totally $\forall$-correct for input $\bar{\xi}$ with respect to ϕ and ψ if and only if the truth of $\phi(\bar{\xi})$ implies that $(\alpha, \mathcal{I})$ is not partially $\exists$-correct for input $\bar{\xi}$ with respect to **true** *and $\sim\psi$.*

(2) $(\alpha, \mathcal{I})$ is totally $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ if and only if the truth of $\phi(\bar{\xi})$ implies that $(\alpha, \mathcal{I})$ is not partially $\forall$-correct with respect to **true** *and $\sim\psi$.*

(3) $(\alpha, \mathcal{I})$ weakly terminates for input $\bar{\xi}$ with respect to ϕ if and only if $(\alpha, \mathcal{I})$ is totally $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and **true**.

(4) $(\alpha, \mathcal{I})$ strongly terminates for input $\bar{\xi}$ with respect to ϕ if and only if $(\alpha, \mathcal{I})$ is totally $\forall$-correct for input $\bar{\xi}$ with respect to ϕ and **true**.

The previous lemma shows that the notions of total correctness and of termination reduce to the two notions of partial $\exists$-correctness and partial $\forall$-correctness and therefore, we now concentrate on the problem of formulating a method for proving partial $\exists$-correctness and partial $\forall$-correctness.

6.3. Construction of the correctness formulae

Given a program $(\alpha, \mathscr{I})$, an input assertion ϕ and an output assertion ψ, we show how to construct formulae $W_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})$ and $W_\alpha^\forall(\bar{\Phi}, \bar{\Psi}, \bar{Q})$ containing unknown predicate symbols $\bar{\Phi}$, $\bar{\Psi}$ and $\bar{Q}$, formalizing the partial $\exists$-correctness and the partial $\forall$-correctness of the program $(\alpha, \mathscr{I})$ with respect to ϕ and ψ in the following sense.

A program $(\alpha, \mathscr{I})$ is partially $\exists$-correct with respect to ϕ and ψ if and only if there exists an assignment of predicates to the unknown predicate symbols $\bar{\Phi}$, $\bar{\Psi}$ and $\bar{Q}$ making the formula $W_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})$ true in the domain of the interpretation $\mathscr{I}$.

Similarly, $(\alpha, \mathscr{I})$ is partially $\forall$-correct with respect to ϕ and ψ if and only if there exists an assignment of predicates to the unknown predicate symbols $\bar{\Phi}$, $\bar{\Psi}$ and $\bar{Q}$, making the formula $W_\alpha^\forall(\bar{\Phi}, \bar{\Psi}, \bar{Q})$ true in the domain of the interpretation $\mathscr{I}$.

The algorithm is the extension of the method presented in [28, 31] to nondeterministic flowchart programs with recursive procedures.

We will follow the usual convention for denoting formulae with free variables, whereby for any arbitrary relation symbol Q, the expression $Q(\bar{x}, \bar{y})$ denotes a formula whose free variables are among the variables $\bar{x}$ and $\bar{y}$. If $\bar{z} = (z_1, \ldots, z_t)$, we let $\exists \bar{z}$ abbreviate $\exists z_1 \cdots \exists z_t$ and $\forall \bar{z}$ abbreviate $\forall z_1 \cdots \forall z_t$.

We now describe in detail the algorithm to construct the correctness formulae. In this algorithm we assume that the vectors of input variables $\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_N$ are all distinct.

Algorithm 6.3.1 (Construction of the correctness formulae)

Step 1: *Choice of cutpoints.* For every procedure α_i, $0 \leq i \leq N$, including the main procedure α_0, choose a set of cutpoints C_i satisfying the following conditions:

(i) The entry **in** and the exit **out** of procedure α_i are in C_i.

(ii) Every loop in procedure α_i contains at least one point in C_i.

(iii) Every node just before or just after a procedure call in procedure α_i belongs to C_i.

Step 2: *Assignment of inductive assertions.* For every procedure α_i, $0 \leq i \leq N$, including the main procedure α_0, atomic formulae are attached to all cutpoints in C_i with the following restrictions:

(i) To the entry of α_i is assigned an atomic formula $\Phi_i(\bar{x}_i)$ and to the exit of α_i is assigned an atomic formula $\Psi_i(\bar{x}_i, \bar{z}_i)$, where Φ_i and Ψ_i are new input and output predicate symbols.

(ii) To every cutpoint c distinct from the entry or the exit of α_i and not just before or just after a procedure call is assigned an atomic formula $Q_c(\bar{x}_i, \bar{y}_i)$ where Q_c is a new predicate symbol.

(iii) For every pair c_1, c_2, where c_1 is the source and c_2 the target of an edge e labeled with a procedure call of the form $P_j(\bar{u}; \bar{v})$ with $1 \leq j \leq N$ (since the main procedure α_0 is never called), we associate formulae as follows.

Let $\bar{w}$ be the set of variables $(\bar{x}_i \cup \bar{y}_i) - (\bar{u} \cup \bar{v})$, that is, the set of input and local variables which are not actual parameters of the procedure call. Recall that $\bar{u}$ and $\bar{v}$

are disjoint sets of variables and that the actual input parameters $\bar{u}$ are unchanged during execution of the procedure call. Therefore, the variables $\bar{u} \cup \bar{w}$ are unchanged during execution of the procedure call. Then, to c_1 is attached a formula $Q_e(\bar{u}, \bar{w}) \wedge \Phi_j(\bar{u})$ and to c_2 is attached a formula $Q_e(\bar{u}, \bar{w}) \wedge \Psi_j(\bar{u}, \bar{v})$, where $Q_e(\bar{u}, \bar{w})$ is a formula called a procedure invariant, Q_e being a new predicate symbol and Φ_j and Ψ_j are the input and output predicate symbols associated with procedure α_j at Step 2(i).

The purpose of the formula $Q_e(\bar{u}, \bar{w})$ is to transmit information across the procedure call, namely the relationship holding among the variables of $\bar{u} \cup \bar{w}$ just before the call. Since the variables in $\bar{u} \cup \bar{w}$ are unchanged during execution of the procedure call, the same relationship holds in c_2 and we will see that it is crucial to have this information in c_2 to guarantee the completeness of the method. Indeed, we will exhibit an example showing that the method is incomplete if the predicate Q_e is omitted (that is, taken to be the constant predicate T). The purpose of the predicates $\Phi_j(\bar{u})$ and $\Psi_j(\bar{u}, \bar{v})$ is to make sure that the procedure α_j is partially correct with respect to the input and output predicates assigned to Φ_j and Ψ_j for input the value of the variables $\bar{u}$ just before the call. These are also indispensable to the completeness of the method. In the formalism used by De Bakker and Meertens [7], programs have a single variable and one can only write assertions about the current value of this variable. Consequently, it is not possible in their model to write assertions similar to our procedure invariants or similar to the output predicates $\Psi_j(\bar{u}, \bar{v})$. This explains why an infinite number of assertions is needed to guarantee the completeness of their method.

Step 3: *Partition every procedure into a set of finite subtrees.* For every procedure α_i, $0 \leq i \leq N$, including the main procedure α_0, form a set T_i of finite subtrees of α_i, where the root and the leaves of every tree are cutpoints in C_i, no interior node of a tree is a cutpoint and no edge in a tree is labeled with a procedure call. The sets T_i are constructed in such a way that every edge not labeled with a procedure call in α_i belongs to at least one tree in the set T_i.

Step 4: *Construct the verification conditions.* For every procedure α_i, $0 \leq i \leq N$ and for every tree T in the set T_i of trees associated with α_i, we construct two formulae denoted $W_T^{\exists}(\Psi_i)(\bar{x}_i)$ and $W_T^{\forall}(\Psi_i)(\bar{x}_i)$ which are used in the construction of the final correctness formulae. The verification conditions $W_T^{\exists}(\Psi_i)(\bar{x}_i)$ and $W_T^{\forall}(\Psi_i)(\bar{x}_i)$ are constructed inductively, starting from the leaves of the tree T and proceeding bottom up to the root. With every node n of the tree T are associated two formulae denoted $R_n^{\exists}(\bar{x}, \bar{y})$ and $R_n^{\forall}(\bar{x}, \bar{y})$ which are computed according to the following algorithm.

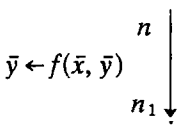
(A) *Initialization of the algorithm.* For every leaf n of the tree T, the formulae $R_n^{\exists}(\bar{x}, \bar{y})$ and $R_n^{\forall}(\bar{x}, \bar{y})$ are defined in the following way.

Case 1: Node n is distinct from the exit of procedure α_i. Then, $R_n^{\exists}(\bar{x}, \bar{y}) \equiv R_n^{\forall}(\bar{x}, \bar{y}) \equiv Q_n(\bar{x}, \bar{y})$, where $Q_n(\bar{x}, \bar{y})$ is the inductive assertion attached to the cutpoint n.

Case 2: Node n is the exit of procedure α_i. Then, $R_n^{\exists}(\bar{x}, \bar{z}) \equiv R_n^{\forall}(\bar{x}, \bar{z}) \equiv \Psi_i(\bar{x}, \bar{z})$,where $\Psi_i(\bar{x}, \bar{z})$ is the output assertion attached to the exit of procedure α_i.

(B) *Inductive step.* For every node n in the tree T which is not a leaf and has exactly k successors $n_1, \ldots, n_k$ $(k \geqslant 1)$, $R_n^{\exists}(\bar{x}, \bar{y})$ is constructed from the formulae $R_{n_1}^{\exists}(\bar{x}, \bar{y}), \ldots, R_{n_k}^{\exists}(\bar{x}, \bar{y})$ and $R_n^{\forall}(\bar{x}, \bar{y})$ is constructed from the formulae $R_{n_1}^{\forall}(\bar{x}, \bar{y}), \ldots, R_{n_k}^{\forall}(\bar{x}, \bar{y})$ as explained below, depending on the nature of n.

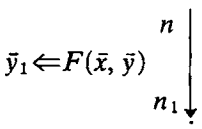
(i) Node n is the source of a functional assignment:



Then, we have

$$R_n^{\exists}(\bar{x}, \bar{y}) \equiv R_{n_1}^{\exists}(\bar{x}, f(\bar{x}, \bar{y})) \quad \text{and} \quad R_n^{\forall}(\bar{x}, \bar{y}) \equiv R_{n_1}^{\forall}(\bar{x}, f(\bar{x}, \bar{y})).$$

(ii) Node n is the source of a relational assignment:



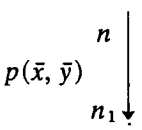
Then, we have

$$R_n^{\exists}(\bar{x}, \bar{y}) \equiv \exists \bar{y}_1 (F(\bar{x}, \bar{y}, \bar{y}_1) \wedge R_{n_1}^{\exists}(\bar{x}, \bar{y}_1))$$

and

$$R_n^{\forall}(\bar{x}, \bar{y}) \equiv \forall \bar{y}_1 (F(\bar{x}, \bar{y}, \bar{y}_1) \supset R_{n_1}^{\forall}(\bar{x}, \bar{y}_1)).$$

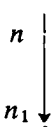
(iii) Node n is the source of a filter test:



Then, we have

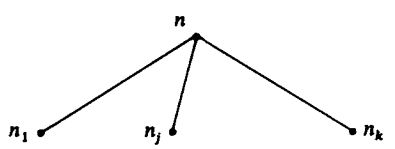
$$R_n^{\exists}(\bar{x}, \bar{y}) \equiv (p(\bar{x}, \bar{y}) \supset R_{n_1}^{\exists}(\bar{x}, \bar{y})) \quad \text{and} \quad R_n^{\forall}(\bar{x}, \bar{y}) \equiv (p(\bar{x}, \bar{y}) \supset R_{n_1}^{\forall}(\bar{x}, \bar{y})).$$

(iv) Node n is the source of a null statement:

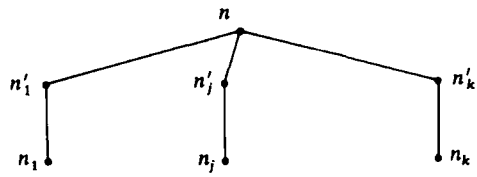


Then, we have $R_n^{\exists}(\bar{x}, \bar{y}) \equiv R_{n_1}^{\exists}(\bar{x}, \bar{y})$ and $R_n^{\forall}(\bar{x}, \bar{y}) \equiv R_{n_1}^{\forall}(\bar{x}, \bar{y})$.

(v) Node n is the source of a choice point $(k \geqslant 2)$:



It is convenient to introduce fictitious nodes $n'_1, \ldots, n'_k$ and represent the above tree in the following way:



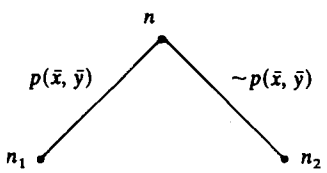
Each edge from n'_j to n_j has the same label as the edge from n to n_j.

With every fictitious node n'_j are associated two formulae $R_j^{\exists}(\bar{x}, \bar{y})$ and $R_j^{\forall}(\bar{x}, \bar{y})$ which are obtained as specified in (B)(i), (B)(ii) and (B)(iv). (Recall that the edges originating from n must be labeled either with functional assignments or relational assignments or the null statement.)

Then, we have

$$R_n^{\exists}(\bar{x}, \bar{y}) \equiv \bigvee_{j=1}^{k} R_j^{\exists}(\bar{x}, \bar{y}) \quad \text{and} \quad R_n^{\forall}(\bar{x}, \bar{y}) \equiv \bigwedge_{j=1}^{k} R_j^{\forall}(\bar{x}, \bar{y}).$$

(vi) Node n is the source of a binary test:



Then, we have

$$R_n^{\exists}(\bar{x}, \bar{y}) \equiv \textbf{if } p(\bar{x}, \bar{y}) \textbf{ then } R_{n_1}^{\exists}(\bar{x}, \bar{y}) \textbf{ else } R_{n_2}^{\exists}(\bar{x}, \bar{y})$$

and

$$R_n^{\forall}(\bar{x}, \bar{y}) \equiv \textbf{if } p(\bar{x}, \bar{y}) \textbf{ then } R_{n_1}^{\forall}(\bar{x}, \bar{y}) \textbf{ else } R_{n_2}^{\forall}(\bar{x}, \bar{y}).$$

(C) *End of the construction.* If n is the root of the tree T, $W_T^{\exists}(\Psi_i)(\bar{x}_i)$ and $W_T^{\forall}(\Psi_i)(\bar{x}_i)$ are defined as follows.

Case *1*: Node n is not the entry of procedure α_i. Then, we have

$$W_T^{\exists}(\Psi_i)(\bar{x}_i) \equiv \forall \bar{y}(Q_n(\bar{x}_i, \bar{y}) \supset R_n^{\exists}(\bar{x}_i, \bar{y}))$$

and

$$W_T^{\forall}(\Psi_i)(\bar{x}_i) \equiv \forall \bar{y}(Q_n(\bar{x}_i, \bar{y}) \supset R_n^{\forall}(\bar{x}_i, \bar{y})),$$

where $Q_n(\bar{x}_i, \bar{y})$ is the inductive assertion attached to the cutpoint n.

Case 2: Node n is the entry of prodecure α_i. Then we have

$$W_T^{\exists}(\Psi_i)(\bar{x}_i) \equiv R_{\mathbf{in}}^{\exists}(\bar{x}_i) \quad \text{and} \quad W_T^{\forall}(\Psi_i) \equiv R_{\mathbf{in}}^{\forall}(\bar{x}_i).$$

Finally, to every procedure α_i, $0 \leq i \leq N$, we assign two formulae $W^{\exists}_{\alpha_i}(\Phi_i, \Psi_i)(\bar{x}_i)$ and $W^{\forall}_{\alpha_i}(\Phi_i, \Psi_i)(\bar{x}_i)$ defined by:

$$W^{\exists}_{\alpha_i}(\Phi_i, \Psi_i)(\bar{x}_i) \equiv \left[\Phi_i(\bar{x}_i) \supset \bigwedge_{T \in T_i} W^{\exists}_T(\Psi_i)(\bar{x}_i) \right]$$

and

$$W^{\forall}_{\alpha_i}(\Phi_i, \Psi_i)(\bar{x}_i) \equiv \left[\Phi_i(\bar{x}_i) \supset \bigwedge_{T \in T_i} W^{\forall}_T(\Psi_i)(\bar{x}_i) \right].$$

The *correctness formulae* $W^{\exists}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x}_0)$ and $W^{\forall}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x}_0)$ associated with the scheme α are obtained as follows.

$$W^{\exists}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x}_0) \equiv \forall \bar{x}_1 \cdots \forall \bar{x}_N \left(\bigwedge_{i=0}^{N} W^{\exists}_{\alpha_i}(\Phi_i, \Psi_i)(\bar{x}_i) \right)$$

and

$$W^{\forall}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x}_0) \equiv \forall \bar{x}_1 \cdots \forall \bar{x}_N \left(\bigwedge_{i=0}^{N} W^{\forall}_{\alpha_i}(\Phi_i, \Psi_i)(\bar{x}_i) \right).$$

The free variables of the above formulae are the unknown input predicate symbols $\bar{\Phi} = (\Phi_0, \Phi_1, \ldots, \Phi_N)$, the unknown output predicate symbols $\bar{\Psi} = (\Psi_0, \Psi_1, \ldots, \Psi_N)$, the vector $\bar{Q}$ of all unknown predicate symbols Q_n and Q_e and the input variables $\bar{x}_0$ of the main procedure α_0.

It should be noted that the predicate symbols Φ_0 and Ψ_0 play a special role and that they are not unknown assertions as the other Φ_i and Ψ_i are, for $1 \leq i \leq N$. Indeed, for every interpretation $\mathcal{I}$, if we want to prove the partial $\exists$-correctness or $\forall$-correctness of the program $(\alpha, \mathcal{I})$ with respect to a given input predicate ϕ and a given predicate ψ for input $\bar{x}_0 = \bar{\xi}$, we claim that it is necessary and sufficient to show that the formulae

$$\exists \Phi_1 \cdots \exists \Phi_N \exists \Psi_1 \cdots \exists \Psi_N \exists \bar{Q} W^{\exists}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x}_0)$$

and

$$\exists \Phi_1 \cdots \exists \Phi_N \exists \Psi_1 \cdots \exists \Psi_N \exists \bar{Q} W^{\forall}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x}_0)$$

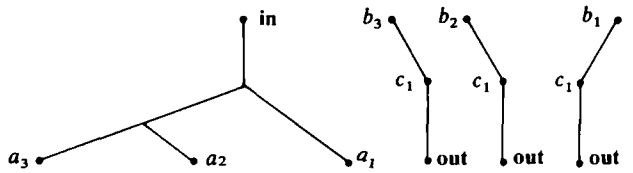
are true under the interpretation $\mathcal{I}^+$ obtained from $\mathcal{I}$ by assigning ϕ to Φ_0, ψ to Ψ_0 and $\bar{\xi}$ to $\bar{x}_0$. Equivalently, the program $(\alpha, \mathcal{I})$ is partially $\exists$-correct (partially $\forall$-correct) with respect to input assertion ϕ and ouput assertion ψ for input $\bar{\xi}$, if and only if there exists an assignment of predicates to the input predicate symbols $\Phi_1, \ldots, \Phi_N$, to the output predicate symbols $\Psi_1, \ldots, \Psi_N$ and to the rest of the predicate symbols Q_c and Q_e, making the above formulae true in the interpretation $\mathcal{I}^+$. Hence, it is necessary to guess not only the usual inductive assertions Q_c, but also the procedure invariants Q_e, and the input predicate Φ_i and the output predicate Ψ_i for all procedures α_i excluding the main procedure.

We observe that for programs without relational assignments or choice-points, the notions of $\exists$-correctness and of $\forall$-correctness coincide. In this case, we will simply use the terminology partial correctness.

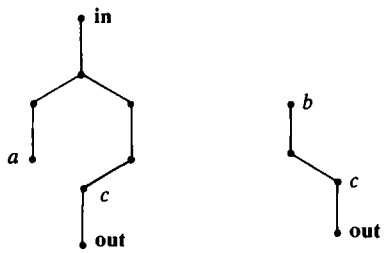
Before proving the main results of this paper, we demonstrate the use of our method by means of an example. In the following, a program $(\alpha, \mathscr{I})$ will be represented by its scheme, with every function, relation or predicate symbol replaced by its interpretation under $\mathscr{I}$ (see Fig. 6).

The set of cutpoints for the main procedure P_0 consists of: **in**, a_1, a_2, a_3, b_1, b_2, b_3, **out**. The role of c_1 is to make the description of the trees easier. Similarly, the set of cutpoints for procedure P_1 consists of: **in**, a, b, **out**. Again, c is not a cutpoint and its role is to simplify the task of describing the trees.

The set of trees associated with the main procedure P_0 is composed of the four trees:



The set of trees associated with procedure P_1 is composed of the two trees:



The input assertion attached to the entry of the main procedure is $\phi(x_1, x_2) \equiv x_1$, $x_2 \in \omega$, where ω denotes the set of nonnegative integers. The output assertion attached to the exit of the main procedure is

$$\psi(x_1, x_2, z) \equiv \textbf{if } x_1 = x_2 \textbf{ then } z = x_2! \textbf{ else}$$
$$\textbf{if } x_1 > x_2 \textbf{ then } z = (x_2 + 1)! \textbf{ else } z = x_1!,$$

where $x_1!$ denotes the function factorial.

To the entry of procedure P_1 is attached the formula $A(x)$ and to the exit of procedure P_1 is attached the formula $B(x, z)$.

The formulae attached to the cutpoints a_1, a_2, a_3, b_1, b_2, b_3 are:

$$Q_{a_1}(x_1, x_2, y_1) \equiv C_1(x_1, x_2, y_1) \wedge A(y_1),$$

$$Q_{b_1}(x_1, x_2, y_1, y_2) \equiv C_1(x_1, x_2, y_2) \wedge B(y_1, y_2),$$

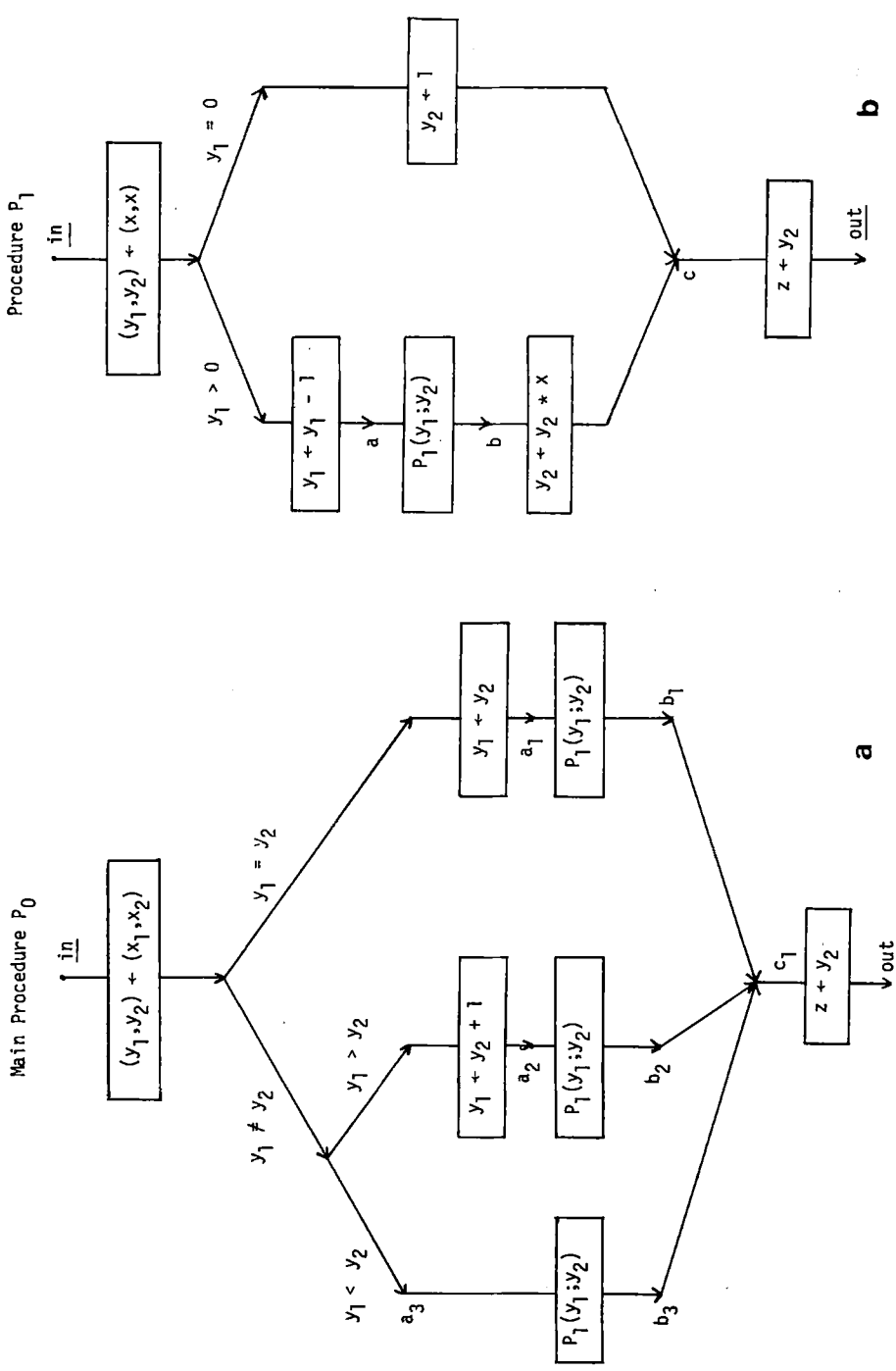


Fig. 6. A program (P_0, P_1) composed of a main procedure P_0 and a recursive procedure P_1.

$$Q_{a_2}(x_1, x_2, y_1) \equiv C_2(x_1, x_2, y_1) \wedge A(y_1),$$

$$Q_{b_2}(x_1, x_2, y_1, y_2) \equiv C_2(x_1, x_2, y_1) \wedge B(y_1, y_2),$$

$$Q_{a_3}(x_1, x_2, y_1) \equiv C_3(x_1, x_2, y_1) \wedge A(y_1),$$

$$Q_{b_3}(x_1, x_2, y_1, y_2) \equiv C_3(x_1, x_2, y_1) \wedge B(y_1, y_2).$$

The formulae attached to the cutpoints a and b are:

$$Q_a(x, y_1) \equiv C(x, y_1) \wedge A(y_1), \qquad Q_b(x, y_1, y_2) \equiv C(x, y_1) \wedge B(y_1, y_2).$$

C_1, C_2, C_3 and C are procedure invariants.

We have six verification conditions corresponding to the six trees listed above:

$$W_1(x_1, x_2) \equiv \textbf{if } x_1 = x_2 \textbf{ then } C_1(x_1, x_2, x_2) \wedge A(x_2) \textbf{ else if } x_1 > x_2$$
$$\textbf{then } C_2(x_1, x_2, x_2 + 1) \wedge A(x_2 + 1) \textbf{ else}$$
$$C_3(x_1, x_2, x_1) \wedge A(x_1),$$

$$W_2(x_1, x_2) \equiv \forall y_1 \forall y_2 [C_3(x_1, x_2, y_1) \wedge B(y_1, y_2) \supset \Psi_0(x_1, x_2, y_2)],$$

$$W_3(x_1, x_2) \equiv \forall y_1 \forall y_2 [C_2(x_1, x_2, y_1) \wedge B(y_1, y_2) \supset \Psi_0(x_1, x_2, y_2)],$$

$$W_4(x_1, x_2) \equiv \forall y_1 \forall y_2 [C_1(x_1, x_2, y_1) \wedge B(y_1, y_2) \supset \Psi_0(x_1, x_2, y_2)],$$

$$W_5(x) \equiv \textbf{if } x > 0 \textbf{ then } C(x, x - 1) \wedge A(x - 1) \textbf{ else } B(x, 1),$$

$$W_6(x) \equiv \forall y_1 \forall y_2 [C(x, y_1) \wedge B(y_1, y_2) \supset B(x, y_2 * x)].$$

The correctness formula associated with the program is:

$$W(A, B, C_1, C_2, C_3, C, x_1, x_2)$$
$$\equiv [\Phi_0(x_1, x_2) \supset W_1(x_1, x_2) \wedge W_2(x_1, x_2) \wedge W_3(x_1, x_2) \wedge W_4(x_1, x_2)]$$
$$\wedge \forall x [A(x) \supset W_5(x) \wedge W_6(x)].$$

We claim that the program (P_0, P_1) is partially correct with respect to the input predicate ϕ and the output predicate ψ for all input, if and only if the formula $\exists A \exists B \exists C_1 \exists C_2 \exists C_3 \exists C \forall x_1 \forall x_2 W(A, B, C_1, C_2, C_3, x_1, x_2)$ is true in the interpretation $\mathcal{I}^+$ obtained from $\mathcal{I}$ by assigning ϕ to Φ_0 and ψ to Ψ_0. Let us consider the following assignment of predicates:

$$A(x) \equiv x \geqslant 0, \qquad B(x, z) \equiv z = x!,$$

$$C(x, y_1) \equiv (y_1 = x - 1) \wedge (x > 0),$$

$$C_1(x_1, x_2, y_1) \equiv (y_1 = x_2) \wedge (x_1 = x_2),$$

$$C_2(x_1, x_2, y_1) \equiv (y_1 = x_2 + 1) \wedge (x_1 > x_2),$$

$$C_3(x_1, x_2, y_1) \equiv (y_1 = x_1) \wedge (x_1 < x_2).$$

The verification that the above assignment makes the correctness formula true is straightforward. Let us check, for instance, that $\forall x_1 \forall x_2 W_3(x_1, x_2)$ and $\forall x W_6(x)$ are true in $\mathcal{I}^+$. We have to prove that the formulae written below are true in the domain of the integers:

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 [(y_1 = x_2 + 1) \wedge (x_1 > x_2) \wedge (y_2 = y_1!) \supset \textbf{if } x_1 \doteq x_2 \textbf{ then}$$

$$y_2 = x_2! \textbf{ else if } x_1 > x_2 \textbf{ then } y_2 = (x_2 + 1)! \textbf{ else } y_2 = x_1!]$$

and

$$\forall x \forall y_1 \forall y_2 [(y_1 = x - 1) \wedge (x > 0) \wedge (y_2 = y_1!) \supset (y_2 * x = x!)].$$

The first formula is obviously true and the second formula follows from the identity $x! = x * (x-1)!$, which holds when x is a positive integer. The last identity actually amounts to the recursive definition of the function factorial.

Let us now assume that we ignore the procedure invariants, that is, we assign the predicate **true** to C, C_1, C_2, C_3, and let us try to find an assignment of predicates for $A(x)$ and $B(x, z)$ making the correctness formula true. We find that there is no such choice. Indeed, if we consider, say,

$$W_2(x_1, x_2) \equiv \forall y_1 \forall y_2 [C_3(x_1, x_2, y_1) \wedge B(y_1, y_2) \supset \Psi_0(x_1, x_2, y_2)],$$

since **true** is assigned to C_3 and $B(y_1, y_2)$ is independent of x_1 and x_2, the only way to make $\forall x_1 \forall x_2 \forall y_1 \forall y_2 W_2(x_1, x_2)$ true is to choose $B \equiv$ **false**, but then,

$$\forall x W_5(x) \equiv \forall x [\textbf{if } x > 0 \textbf{ then } C(x, x-1) \wedge A(x-1) \textbf{ else } B(x, 1)]$$

is false. Therefore, even though the program is partially correct with respect to the predicates ϕ and ψ,, we are unable to prove it by our method, if we ignore the procedure invariants. Similarly, if we ignore the input and output formulae $A(x)$ and $B(x, z)$, this time, since C, C_1, C_2 and C_3 are independent of y_2, there is no choice for C, C_1, C_2, C_3 making the correctness formula true. This example shows that both the input and output assertions assigned to the procedures (excluding the main one) and the procedure invariants are necessary if we want our verification method to be complete.

We will now prove in detail that our method is both consistent and complete. Consistency will be proved in Section 7 and completeness in Section 8. The main theorem that will be established is the following.

Theorem 6.3.2. *For every program $(\alpha, \mathcal{I})$, where α is a scheme $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_N)$ with main procedure α_0, for every input predicate $\phi(\bar{x})$ and output predicate $\psi(\bar{x}, \bar{z})$, we have the following*:

(1) *$(\alpha, \mathcal{I})$ is partially $\exists$-correct with respect to $\phi(\bar{x})$ and $\psi(\bar{x}, \bar{z})$ for all input, if and only if there exists an extension $\mathcal{I}^+$ of the interpretation $\mathcal{I}$, obtained by assigning ϕ to Φ_0, ψ to Ψ_0, and predicates to the predicate symbols $\Phi_1, \ldots, \Phi_N, \Psi_1, \ldots, \Psi_N$ and $\bar{Q}$, making the formula $\forall \bar{x} W_\alpha^{\exists}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ true in $\mathcal{I}^+$.*

(2) $(\alpha, \mathscr{I})$ *is partially* $\forall$*-correct with respect to* $\phi(\bar{x})$ *and* $\psi(\bar{x}, \bar{z})$ *for all input, if and only if there exists an extension* $\mathscr{I}^+$ *of the interpretation* $\mathscr{I}$ *obtained as in* (1) *making the formula* $\forall \bar{x} W_\alpha^\forall(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ *true in* $\mathscr{I}^+$.

We now turn to the proof of the if part of the theorem.

7. Soundness of the inductive assertion method

Let $(\alpha, \mathscr{I})$ be a program with underlying scheme $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_N)$, with main procedure α_0 and let $\phi(\bar{x})$ and $\psi(\bar{x}, \bar{z})$ be the input and output predicates respectively. In order to prove the if part of Theorem 6.3.2, we have to establish a number of lemmas. Before stating our results, it is convenient to introduce the following terminology.

Definition 7.1. Given a program $(\alpha, \mathscr{I})$, an input predicate ϕ and an output predicate ψ, we define the *extension* $\mathscr{I}^+$ of the interpretation $\mathscr{I}$ as the interpretation obtained by assigning ϕ to Φ_0, ψ to Ψ_0, predicates ϕ_i to the input predicate symbols Φ_i $(1 \leqslant i \leqslant N)$, predicates ψ_i to the output predicate symbols Ψ_i $(1 \leqslant i \leqslant N)$, predicates q_c to the predicate symbols Q_c and predicates q_e to the predicate symbols Q_e.

Lemma 7.2. *For every procedure* α_i, $0 \leqslant i \leqslant N$, *for every tree* T *with root* $\bar{c}$ *belonging to the set of trees associated with procedure* α_i, *for every state* $(\bar{\xi}, \bar{\eta})$, *if* $q_{\bar{c}}(\bar{\xi}, \bar{\eta})$ *holds, then we have the following property*:

If $\forall \bar{x} W_T^\exists(\Psi_i)(\bar{x})$ *holds in* $\mathscr{I}^+$, *then either all execution paths with root* $\bar{c}$ *are* FAILURE *paths, or there exists a* SUCCESS *path* π *from the root* $\bar{c}$ *to a leaf* c *of the tree* T *such that* $q_c(\bar{\xi}, \bar{\eta}_1)$ *holds at* c *for the output state* $\bar{\eta}_1$ *associated with* π.

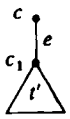
Proof. If the root $\bar{c}$ of the tree T is the entry of procedure α_i, we have $W_T^\exists(\Psi_i)(\bar{x}) \equiv R_{\text{in}}^\exists(\bar{x})$ and otherwise we have $W_T^\exists(\Psi_i)(\bar{x}) \equiv \forall \bar{y}(Q_{\bar{c}}(\bar{x}, \bar{y}) \supset R_{\bar{c}}^\exists(\bar{x}, \bar{y}))$. In either case, since $q_{\bar{c}}(\bar{\xi}, \bar{\eta})$ is assumed to hold as well as $\forall \bar{x} W_T^\exists(\Psi_i)(\bar{x})$, $r_{\bar{c}}^\exists(\bar{\xi}, \bar{\eta})$ holds in $\mathscr{I}^+$, where $r_{\bar{c}}^\exists$ denotes the interpretation of $R_{\bar{c}}^\exists$ in $\mathscr{I}^+$. Consequently, we prove the following claim by induction on the depth of subtrees in T.

Claim. For every node c in the tree T, if $r_c^\exists(\bar{\xi}, \bar{\eta})$ holds at c, either all execution paths from c are FAILURE paths or there exists a SUCCESS path π from c to a leaf c' of the tree T, such that $q_{c'}(\bar{\xi}, \bar{\eta}_1)$ holds at c' for the output $\bar{\eta}_1$ associated with π.

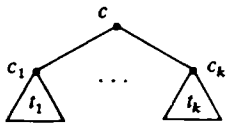
The claim is proved by induction on the depth of the subtree t rooted at c. The proof proceeds from bottom up.

Basis of the induction: Let c be a leaf of the tree T. Since $r_c^\exists(\bar{\xi}, \bar{\eta})$ holds by hypothesis and $q_c(\bar{\xi}, \bar{\eta}) = r_c^\exists(\bar{\xi}, \bar{\eta})$, the claim holds trivially.

Induction step: The subtree rooted at c can be either of the form



or of the form



where $k \geqslant 2$ and $t', t_1, \ldots, t_k$ are subtrees, the cases $t' = c_1$, $t_1 = c_1, \ldots, t_k = c_k$ included.

In the first case, e can be labeled with either a functional assignment, a relational assignment, a filter test or a null statement. We consider the subcases:

(i) Edge e is labeled with a functional assignment of the form $\bar{y} \leftarrow f(\bar{x}, \bar{y})$. Then, we know that $R_c^{\exists}(\bar{x}, \bar{y})$ is given by the identity $R_c^{\exists}(\bar{x}, \bar{y}) \equiv R_{c_1}^{\exists}(\bar{x}, f(\bar{x}, \bar{y}))$. Since the interpretation $\delta(f)$ of f in $\mathscr{S}^+$ is a total function, there is a unique execution path from c to c_1 with output $\bar{y} = \bar{\eta}_2 = \delta(f)(\bar{\xi}, \bar{\eta})$ and since $r_c^{\exists}(\bar{\xi}, \bar{\eta})$ holds by hypothesis, $r_{c_1}^{\exists}(\bar{\xi}, \bar{\eta}_2) = r_c^{\exists}(\bar{\xi}, \bar{\eta})$ holds at c_1. We conclude the proof by applying the inductive hypothesis to the subtree t' whose depth is less than that of t.

(ii) Edge e is labeled with a relational assignment of the form $y_1 \Leftarrow F(\bar{x}, \bar{y})$. As we are in the $\exists$-case, $R_c^{\exists}(\bar{x}, \bar{y}) \equiv \exists \bar{y}_1 (F(\bar{x}, \bar{y}, \bar{y}_1) \wedge R_{c_1}^{\exists}(\bar{x}, \bar{y}_1))$. Since by hypothesis $r_c^{\exists}(\bar{\xi}, \bar{\eta})$ holds at c, there exists $\bar{\eta}_1$ such that $\delta(F)(\bar{\xi}, \bar{\eta}, \bar{\eta}_1)$ holds and $r_{c_1}^{\exists}(\bar{\xi}, \bar{\eta}_1)$ holds at c_1. But then, there exists an execution path π_1 from c to c_1 with $y_1 = \bar{\eta}_1$ and such that $r_{c_1}^{\exists}(\bar{\xi}, \bar{\eta}_1)$ holds at c_1. By the inductive hypothesis, either all execution paths with root c_1 are FAILURE paths, or there exists a SUCCESS path π_2 from c_1 to a leaf c' of the tree such that $q_{c'}(\bar{\xi}, \bar{\eta}_2)$ holds at c' for the output state $\bar{\eta}_2$. Therefore, either all execution paths with root c are FAILURE paths, or the path $\pi_1 \pi_2$ has the desired properties, which concludes the inductive step.

(iii) Edge e is labeled with a filter test of the form $p(\bar{x}, \bar{y})$. Then, $R_c^{\exists}(\bar{x}, \bar{y}) \equiv (p(\bar{x}, \bar{y}) \supset R_{c_1}^{\exists}(\bar{x}, \bar{y}))$. If $\delta(p)(\bar{\xi}, \bar{\eta})$ is false, c is a FAILURE node. Otherwise, there is an execution path from c to c_1 and since $r_c^{\exists}(\bar{\xi}, \bar{\eta})$ holds and $\delta(p)(\bar{\xi}, \bar{\eta})$ holds, $r_{c_1}^{\exists}(\bar{\xi}, \bar{\eta})$ also holds at c_1. The proof is concluded by applying the inductive hypothesis to the subtree t'.

(iv) Edge e is labeled with a null statement. This case is trivial since $R_c^{\exists}(\bar{x}, \bar{y}) \equiv R_{c_1}^{\exists}(\bar{x}, \bar{y})$.

In the second case, either c corresponds to a binary test, or it is a choice-point. We examine the two subcases.

(v) Node c is a choice-point with exactly $k \geqslant 2$ successors. As we are in the $\exists$-case, $R_c^{\exists}(\bar{x}, \bar{y}) \equiv \bigvee_{j=1}^{k} R_j^{\exists}(\bar{x}, \bar{y})$. Since all edges with source c are either labeled with functional assignments or relational assignments and since by hypothesis $r_c^{\exists}(\bar{\xi}, \bar{\eta})$ holds, there exists some j, $1 \leqslant j \leqslant k$, such that $r_j^{\exists}(\bar{\xi}, \bar{\eta})$ holds and there is an execution path from c to c_j. But then, applying the reasoning of (i) and (ii), we have

that $r^{\exists}_{c_j}(\bar{\xi}, \bar{\eta}_j)$ holds at c_j for the output η_j of an execution path from c to c_j. We conclude the proof by applying the inductive hypothesis to the subtree t_j exactly as in case (ii).

(vi) Node c corresponds to a binary test. Then, $R^{\exists}_c(\bar{x}, \bar{y}) \equiv$ **if** $p(\bar{x}, \bar{y})$ **then** $R^{\exists}_{c_1}(\bar{x}, \bar{y})$ **else** $R^{\exists}_{c_2}(\bar{x}, \bar{y})$. If $\delta(p)(\bar{\xi}, \bar{\eta})$ is true, there is an execution path from c to c_1 and $r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta})$ holds at c_1 and otherwise, there is an execution path from c to c_2 and $r^{\exists}_{c_2}(\bar{\xi}, \bar{\eta})$ holds at c_2. The rest of the proof follows by applying the inductive hypothesis to the subtree rooted at c_1 or to the subtree rooted at c_2.

This completes the proof of Lemma 7.2.

We also have a similar lemma applying to partial $\forall$-correctness.

Lemma 7.3. *For every procedure α_i, $0 \leq i \leq N$, for every tree T with root $\bar{c}$ belonging to the set T_i of trees associated with procedure α_i, for every state $(\bar{\xi}, \bar{\eta})$, if $q_{\bar{c}}(\bar{\xi}, \bar{\eta})$ holds, then we have the following property:*

If $\forall \bar{x} W^{\forall}_T(\Psi_i)(\bar{x})$ holds in $\mathscr{I}^+$, then, for every execution path π from the root $\bar{c}$ to a leaf c in the tree T, $q_c(\bar{\xi}, \bar{\eta}_1)$ holds for the output state $\bar{\eta}_1$ at c.

Proof. The proof is similar to that of Lemma 7.2. We prove the following claim.

Claim. For every node c in the tree T, if $r^{\forall}_c(\bar{\xi}, \bar{\eta})$ holds at c, then for every SUCCESS execution path π from c to a leaf c', $q_{c'}(\bar{\xi}, \bar{\eta}_1)$ holds at c' for the output $\bar{\eta}_1$ of π.

The proof of the claim only differs from the previous proof in cases (ii) and (v) and the easy modifications are left to the reader.

We note that the proof showed that, if the tree does not contain filter tests, then all execution paths are SUCCESS paths, which is not surprising since T being a tree, there are no loops, and as our interpretations are total, the execution of every statement is defined.

We prove the soundness of our method by using the unfoldment graph $\alpha^{\forall}_0$ of the main procedure of a scheme α. First, it is necessary to examine more closely the structure of execution paths in the unfoldment graph $\alpha^{\forall}_0$, which is the object of the next two lemmas.

Lemma 7.4. *Given a scheme $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_N)$, for every i, $0 \leq i \leq N$, every path π in the unfoldment graph $\alpha^{\forall}_i$ can be uniquely written as the concatenation $\pi = \pi_1 \cdots \pi_k$ of subpaths π_j, where each subpath π_j in $\alpha^{\forall}_i$ is isomorphic to a subpath π'_j existing in a tree T associated with some procedure α_{m_j}, as defined in Step 3 of Algorithm 6.3.1.*

Proof. We know that for every path π in the unfoldment graph $\alpha^{\forall}_i$, there exists some $n \geq 1$ such that π exists in an isomorphic copy of the graph $\alpha^{(n)}_i$. We prove the lemma by induction on the smallest integer $n \geq 1$ such that π belongs to (a copy) of $\alpha^{(n)}_i$. If

$n = 1$, the lemma is obvious by definition of the set T_i of trees associated with α_i. If $n \geqslant 1$, since $\alpha_i^{(n)} = \alpha^{(n-1)} \circ \alpha_i$ and n is the least integer such that π belongs to a copy of $\alpha_i^{(n)}$, π must be of the form $\pi = \pi_1 \bar{\pi}_1 \pi_2 \cdots \pi_k \bar{\pi}_k \pi_{k+1}$, where each π_j is a (possibly null) path in the graph of α_i in $\alpha_i^{(n)}$ and each $\bar{\pi}_j$ is a path in a copy of some $\alpha_{m_j}^{(n-1)}$. But then, the inductive hypothesis applies to each $\bar{\pi}_j$ and to each π_j, which establishes the induction step.

Let $\alpha_i^1 = \alpha_i$ and $\alpha_i^{n+1} = \alpha^n \circ \alpha_i (n \geqslant 1)$. We also have the following lemma.

Lemma 7.5. *For every path π in the unfoldment graph α_i^{∇}, let n be the smallest integer such that π belongs to an isomorphic copy of $\alpha_i^{(n)}$. Every subpath π_j of π as defined in Lemma 7.4 either belongs to the copy of α_i in $\alpha_i^{(n)} = \alpha^{(n-1)} \circ \alpha_i$, or $n \geqslant 2$ and there exists a unique integer p with $1 \leqslant p \leqslant n-1$, such that $\alpha_i^{(n)} = (\alpha_0^{(n-p)}, \ldots, \alpha_N^{(n-p)}) \circ \alpha_i^p$ and π_j belongs to the copy of some α_{m_j} in $\alpha_{m_j}^{(n-p)} = \alpha^{(n-p-1)} \circ \alpha_{m_j}$, with $0 \leqslant m_j \leqslant N$.*

Proof. If $n = 1$, the lemma is obvious. Otherwise, we have $\alpha_i^{(n)} = (\alpha_0^{(n-1)}, \ldots, \alpha_N^{(n-1)}) \circ \alpha_i$ and either π_j belongs to the copy of α_i or π_j belongs to the copy of $\alpha_{m_j}^{(n-1)}$ for some m_j, $0 \leqslant m_j \leqslant N$. In the first case, the lemma holds. In the second case, by the inductive hypothesis, either π_j belongs to the copy of α_{m_j} in $\alpha_{m_j}^{(n-1)} = \alpha^{(n-2)} \circ \alpha_{m_j}$ and the lemma holds with $p = 1$ or we have the following. There exists some (unique) q, $1 \leqslant q \leqslant n-2$, such that $\alpha_{m_j}^{(n-1)} = \alpha^{(n-q-1)} \circ \alpha_{m_j}^q$ and for some α_{n_j}, π_j belongs to the copy of α_{n_j} in $\alpha^{(n-q-2)} \circ \alpha_{n_j}$. But we have

$$\alpha_i^{(n)} = (\alpha^{(n-q-1)} \circ \alpha^q) \circ \alpha_i = \alpha^{(n-q-1)} \circ (\alpha^q \circ \alpha_i) = \alpha^{(n-q-1)} \circ \alpha_i^{q+1}$$

and the lemma holds with $p = q + 1$. Therefore, the proof is complete.

Since every node n in the unfoldment graph α_i^{∇} is a copy of a node n' in some procedure α_j, we associate to every node n in α_i^{∇} corresponding to a cutpoint an assertion Q_n defined as follows:

- If n corresponds to a cutpoint n' which is neither just before nor just after a procedure call, Q_n is the assertion $Q_{n'}$, attached to n'.
- If n corresponds to the source of an edge e in some α_j and e is labeled with a procedure call $P_k(\bar{u}; \bar{v})$, the assertion Q_n attached to n is $Q_e(\bar{u}, \bar{w}) \wedge \Phi_k(\bar{u})$.
- If n corresponds to the target of an edge e in some α_j and e is labeled with a procedure call $P_k(\bar{u}; \bar{v})$, the assertion Q_n attached to n is $Q_e(\bar{u}, \bar{v}) \wedge \Psi_k(\bar{u}, \bar{v})$.

We are now ready for the proof of the soundness of our method. We prove two lemmas, one for partial $\exists$-correctness and the other for $\forall$-correctness.

Lemma 7.6. *Let $(\alpha, \mathcal{I})$ be a program and ϕ and ψ be the input and output predicates respectively. For every input $\bar{\xi}$, if there exists an extension $\mathcal{I}^+$ of the interpretation $\mathcal{I}$ and the formula $W_\alpha^{\exists}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{\xi})$ is true in $\mathcal{I}^+$, then $(\alpha, \mathcal{I})$ is partially $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ.*

Proof. Assume that $\phi(\bar{\xi})$ holds, since otherwise partial $\exists$-correctness is trivial. If the execution tree $E(\alpha_0^{\nabla}, \mathscr{I})(\bar{\xi})$ contains an infinite execution path or a FAILURE path, then $(\alpha, \mathscr{I})$ is partially $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ. Otherwise, all execution paths for input $\bar{\xi}$ are SUCCESS paths. We prove that among these SUCCESS paths, there exists a path for wich $\psi(\bar{\xi}, \bar{\eta})$ holds for the output state $\bar{\eta}$ associated with that path. Such a path π will be defined inductively using Lemma 7.2. This path π will have the following property: for every prefix π' of π of the form $\pi' = \pi_1 \cdots \pi_k$, with the π_j defined as in Lemma 7.4, if the source of π_j is denoted c_{j-1} and its target c_j, then, for all i, $0 \leq i \leq k$, the assertion attached to c_i in the unfoldment graph α_0^{∇} holds for the current values of the variables at c_i.

For every subpath π_j, the subpath π_j' isomorphic to π_j defined in Lemma 7.4 belongs to a tree denoted T_j. We now define π inductively. Since every execution path for input $\bar{\xi}$ is a SUCCESS path, considering the tree T_0 with root the entry of the main procedure and applying Lemma 7.2, there exists a SUCCESS path π_1' in T_0 such that $q_{c_1'}(\bar{\xi}, \bar{\eta}_1)$ holds at a leaf c_1' of T_0 for the output state $\bar{\eta}_1$. Then, the corresponding path π_1 from c_0 to c_1 in the unfoldment graph α_0^{∇} can be taken as a prefix of π. Assume that a prefix $\pi_1 \cdots \pi_k$ of π has been defined up to this point. By Lemma 7.5, π_k belongs to the copy of a procedure α_h whose entry will be denoted c_h $(0 \leq h \leq k)$. There are three cases.

Case 1: Node c_k' is the source of an edge e in α_h labeled with a procedure call $P_m(\bar{u}; \bar{v})$.

Since the assertion attached to c_k' is of the form $Q_e(\bar{u}, \bar{w}) \wedge \Phi_m(\bar{u})$, by inductive hypothesis $\phi_m(\bar{a})$ holds for the current values $\bar{a}$ of the variables $\bar{u}$ at c_k. The cutpoint c_k' corresponding to c_k is also the root of a tree T_{k+1} in procedure α_m, and since $\phi_m(\bar{a})$ and $W_{\alpha}^{\exists}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{\xi})$ hold in $\mathscr{I}^+$, we can infer that $W_{T_{k+1}}^{\exists}(\Psi_m)(\bar{a})$ holds in $\mathscr{I}^+$. Since $\pi_1 \cdots \pi_k$ is the prefix of a SUCCESS execution path, by Lemma 7.2, there exists a path π_{k+1}' from c_k' to a leaf c_{k+1}' in the tree T_{k+1} and the assertion attached to c_{k+1}' and thus to c_{k+1} in the unfoldment graph α_0^{∇} holds for the current values of the variables.

Case 2: Node c_k' is neither the source nor the target of an edge labeled with a procedure call.

Then, the cutpoint c_k' corresponding to c_k is the root of a tree T_{k+1} in procedure α_h. The input predicate ϕ_h attached to the entry c_h of procedure α_h holds for the values $\bar{a}$ of the input variables at c_h. As in Case 1, we can infer that $W_{T_{k+1}}^{\exists}(\Psi_h)(\bar{a})$ holds, which implies that there exists a path π_{k+1}' from c_k' to a leaf c_{k+1}' of the tree T_{k+1} and the assertion attached to c_{k+1}' holds for the current values of the variables. We have two subcases:

(i) Node c_{k+1}' is not the exit of procedure α_h. Then, the assertions attached to c_{k+1} and c_{k+1}' are identical and the proof of this case is complete.

(ii) Node c_{k+1}' is the exit of procedure α_h. If α_h is the main procedure, the path $\pi_1 \cdots \pi_{k+1}$ is the path π and the lemma is established. Otherwise, the assertion attached to the exit c_{k+1}' of α_h is the output predicate ψ_h. But the procedure invariant

attached to the entry and the exit of procedure α_h holds at c'_{k+1} since it holds at c'_h by the inductive hypothesis. Therefore, the assertion attached to c_{k+1} holds for the current values of the variables.

Case 3: Node c'_k is the target of an edge e in α_h labeled with a procedure call $P_m(\bar{u}; \bar{v})$. Then, c'_k is the root of a tree T_{k+1} in α_h and the rest of the proof is similar to that of the previous cases.

Since every execution path for input $\bar{\xi}$ is a SUCCESS path, the construction of the path π must eventually halt and the proof of Lemma 7.6 is complete.

We also have the following lemma corresponding to partial $\forall$-correctness.

Lemma 7.7. *For every input $\bar{\xi}$, if there exists an extension $\mathcal{I}^+$ of the interpretation $\mathcal{I}$ as in Lemma 7.6 and the formula $W^{\forall}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{\xi})$ is true in $\mathcal{I}^+$, then $(\alpha, \mathcal{I})$ is partially $\forall$-correct for input $\bar{\xi}$ with respect to ϕ and ψ.*

Proof. From Lemma 7.4, every SUCCESS execution path π for input $\bar{\xi}$ can be uniquely written as $\pi = \pi_1 \cdots \pi_k$. We prove by induction on k that for all i, $0 \leqslant i \leqslant k$, the assertion attached to c_i holds for the current values of the variables. The proof is similar to that of the previous lemma, using Lemma 7.3 instead of Lemma 7.2.

Theorem 7.8. *Let $(\alpha, \mathcal{I})$ be a program and ϕ and ψ be the input and output predicates respectively. Let $\mathcal{I}^+$ be an extension of the interpretation $\mathcal{I}$ obtained by assigning ϕ to Φ_0, ψ to Ψ_0, predicates ϕ_i to Φ_i, predicates ψ_i to Ψ_i, predicates q_c to the predicate symbol Q_c and predicates q_e to the predicate symbols Q_e. The following implications hold:*

(1) If $\forall \bar{x} W^{\exists}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ holds in $\mathcal{I}^+$, then $(\alpha, \mathcal{I})$ is partially $\exists$-correct with respect to ϕ and ψ.

(2) If $\forall \bar{x} W^{\forall}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ holds in $\mathcal{I}^+$, then $(\alpha, \mathcal{I})$ is partially $\forall$-correct with respect to ϕ and ψ.

Proof. We only prove (1), the proof of (2) being similar. Assume that $\forall \bar{x} W^{\exists}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ holds in $\mathcal{I}^+$. For every input $\bar{\xi}$, since $W^{\exists}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{\xi})$ holds, by Lemma 7.6 $(\alpha, \mathcal{I})$ is partially $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ. Since this is true for all input, $(\alpha, \mathcal{I})$ is partially $\exists$-correct with respect to ϕ and ψ.

The completeness of the method is carried out in the next section.

8. Completeness of the inductive assertion method

To show the completeness of our method, it suffices to produce an assignment of input predicates ϕ_i to the predicate symbols Φ_i, of output predicates ψ_i to the predicate symbols Ψ_i, of predicates q_c to the predicate symbols Q_c and of predicate q_e

to the predicate symbols Q_e such that, if the program $(\alpha, \mathscr{J})$ is partially $\exists$-correct with respect to ϕ and ψ, $\forall \bar{x} W_\alpha^{\exists}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ holds in the extension $\mathscr{J}^+$ of the interpretation $\mathscr{J}$, obtained by assigning ϕ to Φ_0, ψ to Ψ_0 and other predicates as in the assignment described above (and similarly for $\forall$-correctness). The particular choice of predicates that we present now is obtained informally speaking by pushing backward the output predicates attached to procedures. There is a dual choice obtained by pushing forward the input predicates, but we will not pursue further this possibility in this section. In this section the symbol ϕ_0 is sometimes used for ϕ' and the symbol ψ_0 for ψ in order to handle all input output predicates together.

The choice of input and output predicates is the same for partial $\exists$-correctness and partial $\forall$-correctness, and is as follows:

For all i, $1 \leqslant i \leqslant N$, let

$$\phi_i = \textbf{true}$$

and

$$\psi_i \equiv B(\alpha_i, \mathscr{J}), \text{ the input output relation computed by procedure } \alpha_i.$$

Therefore, we have to guess the relations computed by the procedures α_i, excluding the main procedure for which the input predicate ϕ and the output predicate ψ are already specified. Let R denote the N-tuple $(B(\alpha_1, \mathscr{J}), \ldots, B(\alpha_N, \mathscr{J}))$. The predicates chosen in the other cases depend on the type of partial correctness and we first give the choice for $\exists$-correctness.

Choice of inductive assertions for partial $\exists$-correctness. For every procedure α_i, $0 \leqslant 1 \leqslant N$, and for every cutpoint c distinct from the entry or the exit of procedure α_i and such that c is not just before or just after a procedure call, q_c is defined by the equivalence: For all $\bar{\xi}$, for all $\bar{\eta}$, $q_c(\bar{\xi}, \bar{\eta})$ holds if and only if $\phi_i(\bar{\xi})$ holds and either there is a FAILURE execution path or an infinite execution path starting at c with $\bar{x} = \bar{\xi}$ and $\bar{y} = \bar{\eta}$ or there exists a SUCCESS execution path such that $\psi_i(\bar{\xi}, \bar{\eta}_2)$ holds for the output $\bar{\eta}_2$ of that path.

For every edge e in procedure α_i, $0 \leqslant 1 \leqslant N$, where e is labeled with a procedure call, q_e is defined as follows.

For all $\bar{\xi}$, for all $\bar{\eta}$, $q_e(\bar{\xi}, \bar{\eta})$ holds if and only if $\phi_i(\bar{\xi})$ holds and either there is a FAILURE execution path or an infinite execution path starting at $\bar{c}$ the source of e with $\bar{u} = \bar{\xi}$ and $\bar{w} = \bar{\eta}$ or there exists a SUCCESS path such that $\psi_i(\bar{\xi}, \bar{\eta}_2)$ holds for the output $\bar{\eta}_2$ of that path.

Choice of inductive assertions for partial $\forall$-correctness. For every procedure α_i, $0 \leqslant i \leqslant N$, for every cutpoint c distinct from the entry or the exit of procedure α_i and such that c is not just before or just after a procedure call, q_c is defined as follows. For all $\bar{\xi}$, for all $\bar{\eta}$, $q_c(\bar{\xi}, \bar{\eta})$ holds if and only if $\phi_i(\bar{\xi})$ holds, and for every SUCCESS execution path starting at c with $\bar{x} = \bar{\xi}$ and $\bar{y} = \bar{\eta}$, $\psi_i(\bar{\xi}, \bar{\eta}_2)$ holds for the output $\bar{\eta}_2$ of that path.

For every edge e in procedure α_i, $0 \leq i \leq N$, where e is labeled with a procedure call, q_e is defined as follows. For all $\bar{\xi}$, for all $\bar{\eta}$, $q_e(\bar{\xi}, \bar{\eta})$ holds if and only if $\phi_i(\bar{\xi})$ holds, and for every SUCCESS execution path starting at $\bar{c}$ the source of e with $\bar{u} = \bar{\xi}$ and $\bar{w} = \bar{\eta}$, $\psi_i(\bar{\xi}, \bar{\eta}_2)$ holds for the output $\bar{\eta}_2$ of that path.

It is convenient for stating our results to give the following definition.

Definition 8.1. Let $(\alpha, \mathscr{I})$ be a program and ϕ and ψ be an input predicate and an output predicate. The *extension* $\mathscr{I}^{\exists}$ of the interpretation $\mathscr{I}$ is the interpretation obtained by assigning ϕ to Φ_0, ψ to Ψ_0, the choice of input and output predicates given above to the predicate symbols Φ_i and Ψ_i $(1 \leq i \leq N)$ and the choice of predicates given above for the $\exists$-case to the predicate symbols Q_c and Q_e. Similarly, $\mathscr{I}^{\forall}$ is the *extension* of the interpretation $\mathscr{I}$ obtained as in the $\exists$-case but using the choice of predicates given for the $\forall$-case.

The completeness of our method is proved in two lemmas.

Lemma 8.2. *Let $(\alpha, \mathscr{I})$ be a program and ϕ and ψ be an input and an output predicate. Then we have the following. If $(\alpha, \mathscr{I})$ is partially $\exists$-correct with respect to ϕ and ψ, then the formula $\forall \bar{x} W_{\alpha}^{\exists}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ is true in the extension $\mathscr{I}^{\exists}$.*

Proof. Since $\forall \bar{x}_0 W_{\alpha}^{\exists}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x}_0)$ is the conjunction of the formulae $\forall \bar{x}_i W_{\alpha_i}^{\exists}(\Phi_i, \Psi_i)(\bar{x}_i)$ for $0 \leq i \leq N$ and we have

$$\forall \bar{x}_i W_{\alpha_i}^{\exists}(\Phi_i, \Psi_i)(\bar{x}_i) \equiv \forall \bar{x}_i [\Phi_i(\bar{x}_i) \supset \bigwedge_{T \in T_i} W_T^{\exists}(\Psi_i)(\bar{x}_i)],$$

it suffices to prove that for all i, $0 \leq i \leq N$, for all trees T in the set T_i of trees associated with procedure α_i, for every input $\bar{\xi}$, if $\phi_i(\bar{\xi})$ holds then $W_T^{\exists}(\Psi_i)(\bar{\xi})$ holds in $\mathscr{I}^{\exists}$. We also know that if the root $\bar{c}$ of the tree T is the entry of procedure α_i, then $W_T^{\exists}(\Psi_i)(\bar{x}_i) \equiv R_{\bar{c}}^{\exists}(\bar{x}_i)$ and otherwise, $W_T^{\exists}(\Psi_i)(\bar{x}_i) \equiv \forall \bar{y}(Q_{\bar{c}}(\bar{x}_i, \bar{y}) \supset R_{\bar{c}}^{\exists}(\bar{x}_i, \bar{y}))$. For every procedure α_i, $0 \leq i \leq N$, for every tree T associated with α_i, for every node c in the tree T we shall define below a predicate k_c and prove the following claim.

Claim. If $(\alpha, \mathscr{I})$ is partially $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ, then, for all $\bar{\eta}$, if $k_c(\bar{\xi}, \bar{\eta})$ holds, then $r_c(\xi, \bar{\eta})$ holds in $\mathscr{I}^{\exists}$.

The predicates k_c are defined in the following way: We have three subcases.

Case 1: Node c is the entry of a procedure α_i. For all input $\bar{\xi}$, $k_c(\bar{\xi})$ holds if and only if the program $(\alpha_i, \mathscr{I})$ with main procedure α_i is partially $\exists$-correct for input $\bar{\xi}$ with respect to ϕ_i and ψ_i.

Case 2: Node c is the exit of a procedure α_i. Then k_c is equivalent to ψ_i.

Case 3: Node c is neither the entry nor the exit of a procedure. Then, for all $(\bar{\xi}, \bar{\eta})$, $k_c(\bar{\xi}, \bar{\eta})$ holds if and only if $\phi_i(\bar{\xi})$ holds and either there exists an infinite execution path, or a FAILURE path, or a SUCCESS path such that $\psi_i(\bar{\xi}, \bar{\eta}_2)$ holds for the output $\bar{\eta}_2$ of that path, where all execution paths start from c with $\bar{x} = \bar{\xi}$ and $\bar{y} = \bar{\eta}$.

Now, since $(\alpha, \mathscr{F})$ is assumed to be partially $\exists$-correct with respect to ϕ and ψ for all input, we can derive the following conclusions from the claim:

(i) If c is the entry of a procedure α_i, then by the definition of k_c, $W_T^{\exists}(\Psi_i)(\bar{\xi}) \equiv r_c^{\exists}(\bar{\xi})$ holds since $r_c^{\exists}(\bar{\xi})$ holds by the claim.

(ii) If c is neither the entry nor the exit of a procedure and c is not just before nor just after a procedure call, since $k_c \equiv q_c$, $W_T^{\exists}(\Psi_i)(\bar{\xi})$ holds.

(iii) If c is just after a procedure call, we prove that the logical implication $q_c \supset k_c$ holds, which implies that $W_T^{\exists}(\Psi_i)(\bar{\xi})$ holds. In the present case, we know that $Q_c(\bar{x}, \bar{y}) \equiv Q_e(\bar{u}, \bar{w}) \wedge \Psi_j(\bar{u}, \bar{v})$, where Q_e is the procedure invariant associated with e, $\bar{u}$ and $\bar{v}$ are respectively the actual input parameters and actual output parameters of the procedure call, $\bar{w}$ is the rest of the variables and $\bar{u}$ and $\bar{w}$ are unchanged by the procedure call. Let the values of $\bar{u}$, $\bar{v}$, $\bar{w}$ at c be $\bar{u} = \bar{a}$, $\bar{v} = \bar{b}$, $\bar{w} = \bar{d}$ and let $\bar{c}$ be the source of e. By definition of q_e and ψ_j, if $q_c(\bar{\xi}, \bar{\eta})$ holds at c then $(\bar{a}, \bar{b}) \in R_j$ and there exists an execution path π from $\bar{c}$ with $\bar{u} = \bar{a}$ and $\bar{w} = \bar{d}$ in the underlying graph of the program $(\alpha_i, \mathscr{F})$ such that, either π is a FAILURE path or an infinite path, or π is a SUCCESS path and $\psi_i(\bar{\xi}, \bar{\eta}_3)$ holds for some output state $\bar{\eta}_3$. But then, π is of the form $\pi = e\pi_1$ where π_1 is an execution path from c with $\bar{u} = \bar{a}$, $\bar{v} = \bar{b}$ and $\bar{w} = \bar{d}$ and having the same properties as π. But then, $k_c(\bar{a} \cup \bar{b} \cup \bar{d})$ holds by definition.

Therefore, it only remains to prove the claim. The proof proceeds by induction of the depth of the subtree rooted at c.

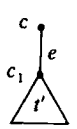
Proof of the claim. If c is a leaf of the tree T, we have three subcases:

(i) Node c is not the exit of a procedure nor the source of a procedure call. Then, $R_c^{\exists}(\bar{x}, \bar{y}) \equiv Q_c(\bar{x}, \bar{y})$, $k_c \equiv q_c$ and since $k_c(\bar{\xi}, \bar{\eta})$ is assumed to hold, $r_c^{\exists}(\bar{\xi}, \bar{\eta})$ holds.

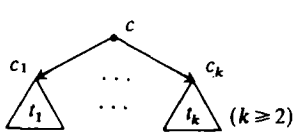
(ii) Node c is the exit of procedure α_i $(0 \leqslant i \leqslant N)$. Then, $R_c^{\exists}(\bar{x}, \bar{z}) \equiv \Psi_i(\bar{x}, \bar{z})$, $k_c \equiv \psi_i$ and since $k_c(\bar{\xi}, \bar{\eta})$ is assumed to hold, $r_c^{\exists}(\bar{\xi}, \bar{\eta})$ holds.

(iii) Node c is the source of a procedure call. Since $R_c^{\exists}(\bar{x}, \bar{y}) \equiv Q_c(\bar{u}, \bar{w}) \equiv Q_e(\bar{u}, \bar{w}) \wedge \Phi_i(\bar{u})$, and with our choice $\phi_i \equiv$ **true**, we have $r_c \equiv q_e$. But it is obvious by definition of q_e and k_c that the logical implication $k_c \supset q_e$ holds, and so r_c holds at c.

If c is not a leaf of the tree T, then the subtree rooted at c is either of the form



or of the form



(i) Edge e is labeled with a functional assignment $\bar{y} \leftarrow f(\bar{x}, \bar{y})$. Since $\delta(f)$ is a total function, there is a unique execution path from c to c_1, and $\eta_2 = \delta(f)(\bar{\xi}, \bar{\eta})$ at c_1. Since $k_c(\bar{\xi}, \bar{\eta})$ is assumed to hold, if there is an execution path from c which is a FAILURE path or an infinite path, there must be an execution path from c_1 which is a

FAILURE path or an infinite path, and then, by definition of k_{c_1}, $k_{c_1}(\bar{\xi}, \bar{\eta}_2)$ holds. Applying the inductive hypothesis, $r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta}_2)$ holds, and since $R^{\exists}_c(\bar{x}, \bar{y}) \equiv R^{\exists}_{c_1}(\bar{x}, f(\bar{x}, \bar{y}))$, $r^{\exists}_c(\bar{\xi}, \bar{\eta})$ holds as desired. If there exists a SUCCESS path π from c such that $\psi_i(\bar{\xi}, \bar{\eta}_3)$ holds for some output state η_3, then π is of the form $\pi = e\pi_1$ and we see that π_1 satisfies the conditions for $k_{c_1}(\bar{\xi}, \bar{\eta}_2)$ to hold. Applying the inductive hypothesis, $r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta}_2)$ holds, and the rest of the proof is identical.

(ii) Edge e is labeled with a relational assignment $y_1 \Leftarrow F(x, y)$. Since $\delta(F)$ is a total relation, every execution path from c passes through c_1. If there is an execution path from c which is a FAILURE path or an infinite path, there must be an execution path from c_1 which is a FAILURE path or an infinite path, and as in case (i) we can conclude that $k_{c_1}(\bar{\xi}, \bar{\eta}_2)$ holds and therefore, $r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta}_2)$ holds for the state $\bar{\eta}_2$ at c_1. If there exists a SUCCESS path from c such that $\psi_i(\bar{\xi}, \bar{\eta}_3)$ holds for some state $\bar{\eta}_3$, we can show as in case (i) that $r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta}_2)$ holds at c_1. But $R^{\exists}_c(\bar{x}, \bar{y}) \equiv \exists \bar{y}_1(F(\bar{x}, \bar{y}, \bar{y}_1) \supset R^{\exists}_{c_1}(\bar{x}, \bar{y}_1))$, and since there exists an execution path with $\bar{y}_1 = \bar{\eta}_2$ at c_1 such that $\delta(F)(\bar{\xi}, \bar{\eta}, \bar{\eta}_2) \wedge r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta}_2)$ holds, $r^{\exists}_c(\bar{\xi}, \bar{\eta})$ holds at c as desired.

(iii) Edge e is labeled with a filter test $p(\bar{x}, \bar{y})$. If $\delta(p)(\bar{\xi}, \bar{\eta}) = \text{false}$, since $R^{\exists}_c(\bar{x}, \bar{y}) \equiv (p(\bar{x}, \bar{y}) \supset R^{\exists}_{c_1}(\bar{x}, \bar{y}))$, $r^{\exists}_c(\bar{\xi}, \bar{\eta})$ holds at c. If $\delta(p)(\bar{\xi}, \bar{\eta}) = \textbf{true}$, every execution path from c is of the form $\pi = e\pi_1$, and by a reasoning similar to that of the previous cases we can show that $r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta})$ holds, and so $r^{\exists}_c(\bar{\xi}, \bar{\eta})$ holds.

(iv) Node c is a choice point. Assume that there are exactly k edges with source c, $k \geqslant 2$. Since all these edges are labeled either with functional assignments or relational assignments, every execution path from c is of the form $\pi = e_j\pi_1$ for one of these edges. By a reasoning similar to that of (i) and (ii), we can show that there exists an execution path π passing through some c_j with $\bar{y} = \bar{\eta}_2$ and such that $r_{c_j}(\bar{\xi}, \bar{\eta}_2)$ holds at c_j. But since every $R^{\exists}_j$ is obtained from $R^{\exists}_{c_j}$ as in case (i) and (ii) of the algorithm to construct the verification conditions, $r^{\exists}_j(\bar{\xi}, \bar{\eta}_2)$ also holds and therefore, $r^{\exists}_c(\bar{\xi}, \bar{\eta})$ holds since $R^{\exists}_c(\bar{x}, \bar{y}) \equiv \bigvee_{j=1}^{k} R^{\exists}_j(\bar{x}, \bar{y})$.

(v) Node c corresponds to a binary test. Since $R^{\exists}_c(\bar{x}, \bar{y}) \equiv \textbf{if } p(\bar{x}, \bar{y}) \textbf{ then } R^{\exists}_{c_1}(\bar{x}, \bar{y}) \textbf{ else } R^{\exists}_{c_2}(\bar{x}, \bar{y})$, if $\delta(p)(\bar{\xi}, \bar{\eta}) = \textbf{true}$, we can show as in (i) that $r^{\exists}_{c_1}(\bar{\xi}, \bar{\eta})$ holds, and if $\delta(p)(\bar{\xi}, \bar{\eta}) = \textbf{false}$, we can show that $r^{\exists}_{c_2}(\bar{\xi}, \bar{\eta})$ holds, and in both cases $r^{\exists}_c(\bar{\xi}, \bar{\eta})$ holds.

This completes the proof of Lemma 8.2. We have a similar lemma for partial $\forall$-correctness.

Lemma 8.3. *If the program $(\alpha, \mathscr{F})$ is partially $\forall$-correct with respect to ϕ and ψ, then the formula $\forall \bar{x} W^{\forall}_{\alpha}(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ is true in the extension $\mathscr{F}^{\forall}$.*

Proof. The proof is analogous to the proof of Lemma 8.2, but using the following choice for the predicates k_c. There are three subcases.

Case 1. Node c is the entry of a procedure α_i. For all input $\bar{\xi}$, $k_c(\bar{\xi})$ holds if and only if the program $(\alpha_i, \mathscr{F})$ with main procedure α_i is partially $\forall$-correct for input $\bar{\xi}$ with respect to ϕ_i and ψ_i.

Case 2. Node c is the exit of a procedure α_i. Then k_c is equivalent to ψ_i.

Case 3. Node c is neither the entry nor the exit of a procedure. Then, for all $(\bar{\xi}, \bar{\eta})$, $k_c(\bar{\xi}, \bar{\eta})$ holds if and only if $\phi_i(\bar{\xi})$ holds and for every SUCCESS execution path π starting from c with $\bar{x} = \bar{\xi}$ and $\bar{y} = \bar{\eta}$, $\psi_i(\bar{\xi}, \bar{\eta}_2)$ holds for the output $\bar{\eta}_2$ of π.

The proof of the claim differs only in (ii) and (iv) and is left to the reader.

Combining the results of Theorem 7.8, Lemma 8.2 and Lemma 8.3, we obtain a proof of Theorem 6.3.2 which we restate for convenience.

Theorem 8.4 (same as Theorem 6.3.2). *Let $(\alpha, \mathscr{I})$ be a program and let ϕ and ψ be the input and output predicates respectively. We have the following:*

(1) *Program $(\alpha, \mathscr{I})$ is partially $\exists$-correct with respect to ϕ and ψ if and only if there exists an extension $\mathscr{I}^+$ of the interpretation $\mathscr{I}$ such that the formula $\forall \bar{x} W_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ holds in $\mathscr{I}^+$.*

(2) *Program $(\alpha, \mathscr{I})$ is partially $\forall$-correct with respect to ϕ and ψ if and only if there exists an extension $\mathscr{I}^+$ of the interpretation $\mathscr{I}$ such that the formula $\forall \bar{x} W_\alpha^\forall(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ holds in $\mathscr{I}^+$.*

It is easily seen from the proofs that Lemma 8.2 and Lemma 8.3 also hold for fixed input, and together with the results of Lemma 7.6 and Lemma 7.7 yield the following theorem which will be used in Section 9 to formalize extension and equivalence.

Theorem 8.5. *Let $(\alpha, \mathscr{I})$ be a program and let ϕ and ψ be the input and output predicates respectively. The following equivalences hold:*

(1) *For every input $\bar{\xi}$, $(\alpha, \mathscr{I})$ is partially $\exists$-correct for input $\bar{\xi}$ with respect to ϕ and ψ if and only if there exists an extension $\mathscr{I}^+$ of the interpretation $\mathscr{I}$ such that the formula $W_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{\xi})$ holds in $\mathscr{I}^+$ for $\bar{\xi}$.*

(2) *For every input $\bar{\xi}$, $(\alpha, \mathscr{I})$ is partially $\forall$-correct for input $\bar{\xi}$ with respect to ϕ and ψ if and only if there exists an extension $\mathscr{I}^+$ of the interpretation $\mathscr{I}$ such that the formula $W_\alpha^\forall(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{\xi})$ holds in $\mathscr{I}^+$ for $\bar{\xi}$.*

It should be noted that Theorems 8.4 and 8.5 are not equivalent, but we need them both to formalize total correctness, extension and equivalence. Theorem 8.4 will be used to formalize partial and total correctness, and Theorem 8.5 will be used to formalize extension and equivalence.

9. Applications of the main theorem: total correctness, extension, equivalence

Before stating our results, it is convenient to introduce the following terminology borrowed from [30].

Definition 9.1. Let $(\alpha, \mathscr{I})$ be a program and ϕ and ψ be an input and an output predicate. We say that the formula $\forall \bar{x} W_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ is *satisfiable* if and only if

there exists an extension $\mathscr{I}^+$ of the interpretation $\mathscr{I}$ as defined in Definition 5.4 such that $\forall \bar{x} W_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ is true in $\mathscr{I}^+$. We say that the formula $\forall \bar{x}_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ is *unsatisfiable* if and only if it is not satisfiable by any extension $\mathscr{I}^+$. We say that the formula $\forall \bar{x} W_\alpha^\exists(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ is *valid* if and only if it is satisfiable in all extensions $\mathscr{I}^+$. The same definitions also apply for the formula $\forall \bar{x} W_\alpha^\forall(\bar{\Phi}, \bar{\Psi}, \bar{Q})(\bar{x})$ corresponding to the $\forall$-case.

In order to make more obvious the fact that Φ_0 and Ψ_0 play a special role, we will redefine $\bar{\Phi}$ as $\bar{\Phi} = (\Phi_1, \ldots, \Phi_N)$ and $\bar{\Psi}$ as $\bar{\Psi} = (\Psi_1, \ldots, \Psi_N)$ and write the correctness formulae as:

$$\forall \bar{x} W_\alpha^\exists(\Phi_0, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x}) \quad \text{and} \quad \forall \bar{x} W_\alpha^\forall(\Phi_0, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x}).$$

Combining Lemma 6.2.5 and Theorem 8.4, we obtain the following theorem, formalizing partial correctness, total correctness and termination in the language of the (first-order) predicate calculus.

Theorem 9.2. *For every program $(\alpha, \mathscr{I})$, where α is a scheme $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_N)$ with main procedure α_0 and $\mathscr{I}$ is an interpretation, for every input predicate ϕ and every output predicate ψ, the following properties hold:*

(1) *$(\alpha, \mathscr{I})$ is partially $\exists$-correct with respect to ϕ and ψ if and only if $\forall \bar{x} W_\alpha^\exists(\Phi_0, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})$ is satisfiable.*

(2) *$(\alpha, \mathscr{I})$ is partially $\forall$-correct with respect to ϕ and ψ if and only if $\forall \bar{x} W_\alpha^\forall(\Phi_0, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})$ is satisfiable.*

(3) *$(\alpha, \mathscr{I})$ is totally $\exists$-correct with respect to ϕ and ψ if and only if*

$$\forall \bar{x}[\Phi_0(\bar{x}) \supset \sim W_\alpha^\forall(T, \bar{\Phi}, \sim \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is valid, or equivalently if and only if

$$\exists \bar{x}[\Phi_0(\bar{x}) \wedge W_\alpha^\forall(T, \bar{\Phi}, \sim \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is unsatisfiable.

(4) *$(\alpha, \mathscr{I})$ is totally $\forall$-correct with respect to ϕ and ψ if and only if*

$$\forall \bar{x}[\Phi_0(\bar{x}) \supset \sim W_\alpha^\exists(T, \bar{\Phi}, \sim \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is valid, or equivalently if and only if

$$\exists \bar{x}[\Phi_0(\bar{x}) \wedge W_\alpha^\exists(T, \bar{\Phi}, \sim \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is unsatisfiable.

(5) *$(\alpha, \mathscr{I})$ weakly terminates for ϕ if and only if*

$$\forall \bar{x}[\Phi_0(\bar{x}) \supset \sim W_\alpha^\forall(T, \bar{\Phi}, F, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is valid, or equivalently,

$$\exists \bar{x}[\Phi_0(\bar{x}) \wedge W_\alpha^\forall(T, \bar{\Phi}, F, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is unsatisfiable.

(6) $(\alpha, \mathscr{P})$ *strongly terminates for* ϕ *if and only if*

$$\forall \bar{x}[\Phi_0(\bar{x}) \supset \sim W_\alpha^\exists(T, \bar{\Phi}, F, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is valid, or equivalently,

$$\exists \bar{x}[\Phi_0(\bar{x}) \wedge W_\alpha^\exists(T, \bar{\Phi}, F, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is unsatisfiable.

9.2. Extension and equivalence

Theorem 8.5 also allows us to formalize in (second-order) predicate calculus, properties such as extension and equivalence. Because nondeterminism is allowed, many definitions are possible, depending on the requirements that are placed on execution paths. The two notions of extension and equivalence that we define below have been chosen because they are closely related to partial $\exists$-correctness and partial $\forall$-correctness, and so, they can be formalized in predicate calculus using Theorem 8.5. It might be objected that the notion of universal extension defined below is a bit unnatural, but the notion of universal equivalence associated with it makes more sense and moreover this is the definition which works for proving the theorem. Concerning the latter point, it appears that the definition given in [30] is not correct, in the sense that it does not imply clause (j) of Theorem 3 in [30].

Let $(\alpha_1, \mathscr{P})$ and $(\alpha_2, \mathscr{P})$ be two compatible programs, that is, programs having the same input variables and the same output variables. We define the notions of existential extension, universal extension, existential equivalence and universal equivalence. But first, in order to simplify the definitions and proofs, we redefine the semantics of a program using the undefined symbol $\perp$ for representing the result of FAILURE paths or diverging paths.

Definition 9.3. The relation $B(\alpha, \mathscr{P})$ computed by a nondeterministic recursive program $(\alpha, \mathscr{P})$ is defined as follows:

- $(\bar{\xi}, \bar{\eta}) \in B(\alpha, \mathscr{P})$ if and only if there is a SUCCESS path with input $\bar{\xi}$ and output $\bar{\eta}$;
- $(\bar{\xi}, \perp) \in B(\alpha, \mathscr{P})$ if and only if there is either a FAILURE path or a diverging path for input $\bar{\xi}$.

Definition 9.4. (1) $(\alpha_2, \mathscr{P})$ is an $\exists$-*extension* of $(\alpha_1, \mathscr{P})$ if, for all input $\bar{a}$, for all $\bar{b} \neq \perp$ such that $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{P})$, $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{P})$.

(2) $(\alpha_2, \mathscr{P})$ is an $\forall$-*extension* of $(\alpha_1, \mathscr{P})$ if, for all input $\bar{a}$, either $(\bar{a}, \perp) \in B(\alpha_2, \mathscr{P})$($B(\alpha_2, \mathscr{P})$ diverges for $\bar{a}$) or $(\bar{a}, \perp) \notin B(\alpha_2, \mathscr{P})$ and $(\bar{a}, \perp) \notin B(\alpha_1, \mathscr{P})$ and for all $\bar{b} \neq \perp$, $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{P})$ implies $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{P})$.

(3) $(\alpha_1, \mathscr{P})$ and $(\alpha_2, \mathscr{P})$ are $\exists$-*equivalent* if and only if $(\alpha_2, \mathscr{P})$ $\exists$-extends $(\alpha_1, \mathscr{P})$ and $(\alpha_1, \mathscr{P})$ $\exists$-extends $(\alpha_2, \mathscr{P})$.

(4) $(\alpha_1, \mathscr{P})$ and $(\alpha_2, \mathscr{P})$ are $\forall$-*equivalent* if and only if $(\alpha_2, \mathscr{P})$ $\forall$-extends $(\alpha_1, \mathscr{P})$ and $(\alpha_1, \mathscr{P})$ $\forall$-extends $(\alpha_2, \mathscr{P})$.

We use the following notations: $\overset{\exists}{\leqslant}$ for $\exists$-extension, $\overset{\forall}{\leqslant}$ for $\forall$-extension, $\overset{\exists}{\equiv}$ for $\exists$-equivalence and $\overset{\forall}{\equiv}$ for $\forall$-equivalence. $\exists$-equivalence and $\forall$-equivalence can also be defined more explicitly as follows:

(3') $(\alpha_1, \mathscr{I})$ and $(\alpha_2, \mathscr{I})$ are $\exists$-equivalent if, for all input $\bar{a}$, for all $\bar{b} \neq \perp$, $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})$ if and only if $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})$.

(4') $(\alpha_1, \mathscr{I})$ and $(\alpha_2, \mathscr{I})$ are $\forall$-equivalent if, for all input $\bar{a}$, either $(\bar{a}, \perp) \in B(\alpha_1, \mathscr{I})$ and $(\bar{a}, \perp) \in B(\alpha_2, \mathscr{I})$, or $(\bar{a}, \perp) \notin B(\alpha_1, \mathscr{I})$ and $(\bar{a}, \perp) \notin B(\alpha_2, \mathscr{I})$ and for all $\bar{b} \neq \perp$, $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})$ if and only if $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})$.

We note that $\forall$-equivalence tells us that $(\alpha_1, \mathscr{I})$ has a diverging computation for input $\bar{a}$, if and only if $(\alpha_2, \mathscr{I})$ has a diverging computation for input $\bar{a}$, but in that case, we have no information about the $\bar{b} \neq \perp$ such that either $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})$ or $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})$. On the other hand, $\exists$-equivalence only gives information about the $\bar{b} \neq \perp$ such that $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})$ and $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})$. This suggests the definition of a stronger kind of equivalence, which holds if $(\alpha_1, \mathscr{I})$ and $(\alpha_2, \mathscr{I})$ are at the same time $\exists$-equivalent and $\forall$-equivalent. This strong equivalence is denoted $\equiv$, and from the previous definitions it can be stated in the following way:

Strong Equivalence: $(\alpha_1, \mathscr{I})$ and $(\alpha_2, \mathscr{I})$ are *strongly equivalent* if, for all input $\bar{a}$, $(\alpha_1, \mathscr{I})$ has a diverging computation for input $\bar{a}$ if and only if $(\alpha_2, \mathscr{I})$ has a diverging computation for input $\bar{a}$ and, for all $\bar{b} \neq \perp$, $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})$ if and only if $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})$.

The relationship between $\exists$-extension and partial $\exists$-correctness on one hand, and $\forall$-extension and partial $\forall$-correctness on the other hand, is given by the following lemma.

Lemma 9.5. (1) *$(\alpha_2, \mathscr{I})$ is an $\exists$-extension of $(\alpha_1, \mathscr{I})$ if and only if, for all output predicates ψ, for all inputs $\bar{a}$, if $(\alpha_2, \mathscr{I})$ is partially $\forall$-correct with respect to* **true** *and ψ for input $\bar{a}$, then $(\alpha_1, \mathscr{I})$ is partially $\forall$-correct with respect to* **true** *and ψ for input $\bar{a}$.*

(2) *$(\alpha_2, \mathscr{I})$ is an $\forall$-extension of $(\alpha_1, \mathscr{I})$ if and only if, for all output predicates ψ, for all inputs $\bar{a}$, if $(\alpha_1, \mathscr{I})$ is partially $\exists$-correct with respect to* **true** *and ψ for input $\bar{a}$, then $(\alpha_2, \mathscr{I})$ is partially $\exists$-correct with respect to* **true** *and ψ for input $\bar{a}$.*

Proof. Formally, (1) can be expressed as:

$$(\alpha_1, \mathscr{I}) \overset{\exists}{\leqslant} (\alpha_2, \mathscr{I}) \equiv \forall \psi \forall \bar{a} [pc^{\forall}(\mathbf{true}, (\alpha_2, \mathscr{I}), \psi)(\bar{a}) \supset pc^{\forall}(\mathbf{true}, (\alpha_1, \mathscr{I}), \psi)(\bar{a})] \quad (9.1)$$

and (2) can be expressed as:

$$(\alpha_1, \mathscr{I}) \overset{\forall}{\leqslant} (\alpha_2, \mathscr{I}) \equiv \forall \psi \forall \bar{a} [pc^{\exists}(\mathbf{true}, (\alpha_1, \mathscr{I}), \psi)(\bar{a}) \supset pc^{\exists}(\mathbf{true}, (\alpha_2, \mathscr{I}), \psi)(\bar{a})]. \quad (9.2)$$

We prove the contra-positive of (9.1) and (9.2). We begin with (9.1). The negation of the left-hand side of (9.1) can be written as

$$(L)': \quad \exists \bar{a} \exists \bar{b}[(\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})) \wedge ((\bar{a}, \bar{b}) \notin B(\alpha_2, \mathscr{I})].$$

The negation of the right-hand side of (9.1) can be written as

$$(R)': \quad \exists \psi \exists \bar{a}[\forall \bar{b}[(\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})) \supset \psi(\bar{a}, \bar{b})]$$
$$\wedge \exists \bar{b}[(\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})) \wedge \sim \psi(\bar{a}, \bar{b})]].$$

Suppose that $(L)'$ holds. Then if we choose for ψ the predicate $\psi(\bar{a}, \bar{b}) \equiv (\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I}))$, we see that $(R)'$ holds. Suppose that $(R)'$ holds. Then, there exists $\bar{a}$ and $\bar{b}$ such that $\bar{b} \neq \perp$ and $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})$ and $\psi(\bar{a}, \bar{b}) =$ **false**. If $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})$, using $(R)'$ we have that $\psi(\bar{a}, \bar{b}) =$ **true**, contradicting $\psi(\bar{a}, \bar{b}) =$ **false**, so $(\bar{a}, \bar{b}) \notin B(\alpha_2, \mathscr{I})$ and $(L)'$ holds for $\bar{a}$ and $\bar{b}$ as above. Therefore $(L)'$ and $(R)'$ are equivalent, and so, (9.1) holds as desired.

Let us now consider (9.2). The negation of the left-hand side of (9.2) can be written as

$$(L)': \quad \exists \bar{a}[((\bar{a}, \perp) \notin B(\alpha_2, \mathscr{I})) \wedge [((a, \perp) \in B(\alpha_1, \mathscr{I}))$$
$$\vee \exists \bar{b}[(\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})) \wedge ((\bar{a}, \bar{b}) \notin B(\alpha_2, \mathscr{I}))]]].$$

The negation of the right-hand side of (ii) can be written as

$$(R)': \quad \exists \psi \exists \bar{a}[[((\bar{a}, \perp) \in B(\alpha_1, \mathscr{I})) \vee \exists \bar{b}[(\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})) \wedge \psi(\bar{a}, \bar{b})]]$$
$$\wedge [((\bar{a}, \perp) \notin B(\alpha_2, \mathscr{I})) \wedge \forall \bar{b}[(\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})) \supset \sim \psi(\bar{a}, \bar{b})]].$$

Suppose that $(L)'$ holds. If $(\bar{a}, \perp) \in B(\alpha_1, \mathscr{I})$ holds, it suffices to choose $\psi \equiv$ **false** to make $(R)'$ true. If $(\bar{a}, \perp) \notin B(\alpha_1, \mathscr{I})$, it suffices to choose $\psi(\bar{a}, \bar{b}) \equiv \sim[(\bar{b} \neq \perp) \wedge ((\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I}))]$ to make $(R)'$ true. Suppose that $(R)'$ holds. If $(\bar{a}, \perp) \in B(\alpha_1, \mathscr{I})$ holds, since $(\bar{a}, \perp) \notin B(\alpha_2, \mathscr{I})$ holds, $(L)'$ holds. If $(\bar{a}, \perp) \in B(\alpha_1, \mathscr{I})$ does not hold, there exists $\bar{a}$ and $\bar{b}$ such that $\bar{b} \neq \perp$ and $(\bar{a}, \bar{b}) \in B(\alpha_1, \mathscr{I})$ and $\psi(\bar{a}, \bar{b}) =$ **true**. If $(\bar{a}, \bar{b}) \in B(\alpha_2, \mathscr{I})$ holds, using $(R)'$ we see that $\psi(\bar{a}, \bar{b}) =$ **false**, contradicting $\psi(\bar{a}, \bar{b}) =$ **true**. Therefore $(\bar{a}, \bar{b}) \notin B(\alpha_2, \mathscr{I})$ holds, and so $(L)'$ holds as desired. This concludes the proof that (9.2) holds, and the proof of the lemma is complete.

Corollary 9.6. (1) *$(\alpha_1, \mathscr{I})$ is $\exists$-equivalent to $(\alpha_2, \mathscr{I})$ if and only if, for all ψ, for all $\bar{a}$, $(\alpha_1, \mathscr{I})$ is partially $\forall$-correct with respect to* **true** *and ψ for input $\bar{a}$ if and only if $(\alpha_2, \mathscr{I})$ is partially $\forall$-correct with respect to* **true** *and ψ for input $\bar{a}$.*

(2) *$(\alpha_1, \mathscr{I})$ is $\forall$-equivalent to $(\alpha_2, \mathscr{I})$ if and only if, for all ψ, for all $\bar{a}$, $(\alpha_1, \mathscr{I})$ is partially $\exists$-correct with respect to* **true** *and ψ for input $\bar{a}$ if and only if $(\alpha_2, \mathscr{I})$ is partially $\exists$-correct with respect to* **true** *and ψ for input $\bar{a}$.*

Using Theorem 8.5 and Lemma 9.5, we obtain the following theorem, formalizing the various notions of extension and equivalence in the second-order predicate calculus.

Theorem 9.7. *For every pair of compatible programs* $(\alpha_1, \mathscr{I})$ *and* $(\alpha_2, \mathscr{I})$, *the following equivalences hold*:

(1) $(\alpha_2, \mathscr{I})$ $\exists$*-extends* $(\alpha_1, \mathscr{I})$ *if and only if the formula*

$$\forall \Psi_0 \forall \bar{x}[\exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_2^{\forall}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x}) \supset \exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_1^{\forall}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is true in $\mathscr{I}$.

(2) $(\alpha_2, \mathscr{I})$ $\forall$*-extends* $(\alpha_1, \mathscr{I})$ *if and only if the formula*

$$\forall \Psi_0 \forall \bar{x}[\exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_1^{\exists}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x}) \supset \exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_2^{\exists}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is true in $\mathscr{I}$.

(3) $(\alpha_1, \mathscr{I})$ *is* $\exists$*-equivalent to* $(\alpha_2, \mathscr{I})$ *if and only if the formula*

$$\forall \Psi_0 \forall \bar{x}[\exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_1^{\forall}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x}) \equiv \exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_2^{\forall}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is true in $\mathscr{I}$.

(4) $(\alpha_1, \mathscr{I})$ *is* $\forall$*-equivalent to* $(\alpha_2, \mathscr{I})$ *if and only if the formula*

$$\forall \Psi_0 \forall \bar{x}[\exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_1^{\exists}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x}) \equiv \exists \bar{\Phi} \exists \bar{\Psi} \exists \bar{Q} W_2^{\exists}(T, \bar{\Phi}, \Psi_0, \bar{\Psi}, \bar{Q})(\bar{x})]$$

is true in $\mathscr{I}$.

Note that the formulae involved are second-order formulae and that it is not possible to eliminate the second-order existential quantifiers as we did in Theorem 9.2. We refer the interested reader to [11, 12] for examples illustrating the above methods.

Acknowledgment

I would like to thank Emily Friedman and Sheila Greibach for their help in improving earlier versions of this paper. I also would like to thank Ronald Book for many helpful comments and David Harel and Amir Pnueli for bringing certain references and details to my attention. Finally, I thank the referee for scrutinizing the manuscript very carefully and suggesting many improvements.

References

[1] K.R. Apt and J.W. De Bakker, Semantics and proof theory of PASCAL procedures, in *Automata, Languages and Programming*, Lecture Notes in Computer Science **52** (Springer, Berlin, 1977).

[2] K.R. Apt and L.G.L.T. Meertens, Completeness with finite systems of intermediate assertions for recursive program schemes, Technical Report IW 84/77, Mathematical Center, Amsterdam (1977).

[3] E.A. Ashcroft and Z. Manna, Formalization of properties of parallel programs, in *Machine Intelligence* 6 (Edinburgh University Press, Edinburgh, 1970) 17–41.
[4] E.M. Clarke, Programming language constructs for which it is impossible to obtain good Hoare-like axiom systems, *Proc. 4th Annual Symposium on Principles of Programming Languages*, Los Angeles, CA (1977) 10–20.
[5] E.M. Clarke, Program invariants as fixed points, *Proc. 18th Symposium on Foundations of Computer Science*, Providence, RI (1977) 18–29.
[6] J.W. De Bakker, Semantics and termination of nondeterministic recursive programs, in S. Michaelson and R. Milner, Eds., *Automata, Languages and Programming* (Edinburgh University Press, Edinburgh, 1976) 435–477.
[7] J.W. De Bakker and L.G.L.T. Meertens, On the completeness of the inductive assertion method, *J. Comput. System Sci.* **11**(3) (1975) 323–357.
[8] W.P. De Roever, Recursive program schemes: semantics and proof theory, Mathematical Center Tracts No. 70, Mathematisch Centrum, Amsterdam (1976).
[9] R.W. Floyd, Nondeterministic algorithms, *J. ACM* **14** (4) (1967) 636–644.
[10] R.W. Floyd, Assigning meanings to programs, in: J. T. Schwartz, Ed., *Mathematical Aspects of Computer Science* (1967) 19–32.
[11] J.H. Gallier, Semantics and correctness of classes of deterministic and nondeterministic recursive programs, Ph.D. dissertation, U.C.L.A.
[12] J.H. Gallier, Semantics and correctness of nondeterministic flowchart programs with recursive procedures, in: *Automata, Languages and Programming, Fifth Colloquium, Udine, Italy, July* 1978, Lecture Notes in Computer Science **62** (Springer, Berlin, 1978) 251–267.
[13] G.A. Gorelick, A complete axiomatic system for proving assertions about recursive and nonrecursive programs, Technical Report No. 75, Department of Computer Science, University of Toronto (1975).
[14] S.A. Greibach, Theory of program structures: schemes, semantics, verification, Lecture Notes in Computer Science **36** (Springer, Berlin 1975).
[15] J.A. Goguen and J. Meseguer, Correctness of recursive flow diagram programs, Semantics and Theory of Computation Report No. 8, Computer Science Department, University of California, Los Angeles (1977).
[16] F. Harary, *Graph Theory* (Addison-Wesley, Reading, MA, 1971) 274.
[17] D. Harel, Arithmetical completeness in logics of programs, Technical Report, Laboratory for Computer Science, MIT, Cambridge, MA (1977).
[18] D. Harel, Complete axiomatization of properties of recursive programs, Technical Report, Laboratory for Computer Science, MIT, Cambridge, MA (1977).
[19] D. Harel, On the correctness of regular deterministic programs; a unifying survey, Technical Report, Laboratory for Computer Science, MIT, Cambridge, MA (1977).
[20] D. Harel, *First-Order Dynamic Logic*, Lecture Notes in Computer Science **68** (Springer, Berlin, 1979).
[21] D. Harel and V.R. Pratt, Nondeterminism in logics of programs, *Proc. 5th ACM Symposium on Principles of Programming Languages*, Tucson, AZ (1978).
[22] D. Harel, A.R. Meyer and V.R. Pratt, Computability and completeness in logics of programs, *Proc. 9th Annual ACM Symposium on Theory of Computing*, Boulder, CO (1977) 261–268.
[23] D. Harel, A. Pnueli and J. Stavi, A complete axiomatic system for proving deductions about recursive programs, *Proc. 9th Annual ACM Symposium on Theory of Computing*, Boulder, CO (1977) 249–260.
[24] D. Harel, A. Pnueli and J. Stavi, Completeness issues for inductive assertions and Hoare's method, Computer Science Technical Report, Department of Mathematical Sciences, Tel-Aviv University (1976).
[25] C.A.R. Hoare, Procedure and parameters: an axiomatic approach, in: *Symposium on Semantics of Algorithmic Languages*, Lecture Notes in Mathematics **188** (Springer, New York, 1971) 102–116.
[26] S. Igarashi, R.L. London and D.C. Luckham, Automatic program verification I: A logical basis and its implementation, *Acta Informat.* **4** (1975) 145–182.
[27] R.J. Lipton, A necessary and sufficient condition for the existence of Hoare logics, *Proc. 18th Symposium on Foundations of Computer Science*, Providence, RI (1977) 1–6.

[28] Z. Manna, *Mathematical Theory of Computation* (McGraw-Hill, New York, 1974).
[29] Z. Manna, The correctness of programs, *J. Comput. System Sci.* **3** (1969) 119–127.
[30] Z. Manna, Mathematical theory of partial correctness, in: E. Engeler, Ed., *Symposium on Semantics of Algorithmic Languages*, Lecture Notes in Mathematics **188** (Springer, Berlin, 1971) 252–269.
[31] Z. Manna, The correctness of nondeterministic programs, *Artificial Intelligence* **1** (1970) 1–26.
[32] Z. Manna and A. Pnueli, Formalization of properties of functional programs, *JACM* **17** (3) (1970) 555–569.
[33] J. McCarthy, A basis for a mathematical theory of computation, in: P. Braffort and D. Hirschberg, Eds., *Computer Programming and Formal Systems* (North-Holland, Amsterdam, 1963) 37–70.
[34] Z. Manna and R. Waldinger, Is sometime sometimes better than always? Intermittent assertions in proving program correctness, *Comm. ACM* **21** (2) (1978) 159–172.
[35] V.R. Pratt, Semantical considerations on Floyd-Hoare Logic, Technical Report MIT/LCS/TR-168, MIT, Cambridge, MA (1976).