A COMPLETE AXIOMATIC SYSTEM FOR PROVING DEDUCTIONS ABOUT RECURSIVE PROGRAMS

David Harel +
Massachusetts Institute of Technology, Cambridge, MA 02139

Amir Pnueli
Tel-Aviv University, Tel-Aviv, Israel

and
Jonathan Stavi
Bar-Ilan University, Ramat-Gan, Israel

Abstract.

Denoting a version of Hoare's system for proving partial correctness of recursive programs by ***H***, we present an extension ***D*** which may be thought of as ***H*** $\cup$ {$\wedge,\vee,\exists,\forall$} $\cup$ H^{-1}, including the rules of ***H***, four special purpose rules and inverse rules to those of Hoare. ***D*** is shown to be a complete system (in Cook's sense) for proving deductions of the form $\sigma_1,\dots,\sigma_n \vdash \sigma$ over a language, the wff's of which are assertions in some assertion language L and partial correctness specifications of the form p{α}q. All valid formulae of L are taken as axioms of ***D***. It is shown that ***D*** is sufficient for proving partial correctness, total correctness and program equivalence as well as other important properties of programs, the proofs of which are impossible in ***H***. The entire presentation is worked out in the framework of nondeterministic programs employing iteration and mutually recursive procedures.

1. Introduction.

The axiomatic method of specifying semantics of programs, as given by Hoare ([10], [11] and also [12]) lends itself very successfully to a specific goal, namely that of proving partial correctness of specific programs. A convenient description of the method employs an assertion language L and a formal proof system ***H*** having as axioms all logically valid formulae of L. A proof of a partial correctness specification σ: p{α}q where p,q are wff's in L, is carried out in ***H*** by composing α from more primitive program segments, starting from a finite number of assumptions in L. A well known result is that the conventional Hoare system and its variants are complete if L is strong enough to express all needed assertions. Various definitions of this strength are <u>expressiveness</u> of L (Cook[3]), or <u>tidiness</u> of all programs (Pratt[15]). Cook[3] showed that first order arithmetic is expressive, thus proving completeness of ***H*** for this important special case of L. Extensions of Hoare's system to cover recursion and mutual recursion have also been proved complete under similar conditions (see Gorelick[7], Harel et al[9]).

A suitable such system ***H*** can in fact be thought of as a formal system for proving the correctness of <u>deductions</u> of the form $\sigma_1,\dots,\sigma_n \vdash$ p{α}q under the restriction that each of the σ_i is a procedure declaration or a formula of L. However, when considering general deductions of the form $\sigma_1,\dots,\sigma_n \vdash \sigma$ (where the σ_i may also be partial correctness specifications), it is easy to come up with semantically valid deductions which cannot be derived in ***H***. Two examples are

(1) p{<u>if</u> r <u>then</u> α <u>else</u> β <u>fi</u>}q
$\models$ p{<u>if</u> ¬r <u>then</u> β <u>else</u> α <u>fi</u>}q

(2) p{α}q , r{α}q $\models$ pvr{α}q

(a rule which, while being underivable in ***H***, can be shown to be superfluous for any concrete proof of partial correctness, Igarashi et al[12]).

These examples illustrate the absense (in ***H***) of mechanisms for (1) extracting information from a specification p{α}q about parts of α (where α is a complex program segment), and (2) combining the information given in different specifications about

---

+ The work of this author was partially supported by NSF under contract MCS76-18461.

the same program segment. H can be seen to be complete only for "simple" deductions, in which the antecedents σ_i include for each given α, at most one specification of the form p{α}q, and all such α's are simple specifications consisting of a single assignment or call statement, or a single program segment variable (PSV), which is a symbol standing for an arbitrary program segment.

In Section II we present our system D which is an extension of Hoare's system, and in Section III show that D is sound and complete for any deduction ($\sigma_1,\ldots,\sigma_n \vdash_D \sigma$ iff $\sigma_1,\ldots,\sigma_n \models \sigma$), that is, σ can be proved in D from assumptions $\sigma_1,\ldots,\sigma_n$, iff σ is true in every model satisfying $\sigma_1,\ldots,\sigma_n$. Here the σ_i can themselves be any partial correctness specifications.

The completeness result is shown by proving a series of more restricted theorems, holding for successively richer subsystems of D, thus clarifying the whole process and also achieving a side effect of indicating the precise role in D played by its important components.

A variety of properties of programs can be proved using D, and the completeness result ensures us that when L is expressive (e.g. in arithmetic), a proof exists for each valid such property. The following possibilities are described in Section IV:

(i) proving the partial correctenss of a given program,

(ii) proving the total correctness of a given program,

(iii) proving the (strong) equivalence of programs,

(iv) establishing derived rules,

(v) carrying out modular proofs of program correctness given properties of segments of the program,

(vi) simplifying complex program segments and establishing valid program transformations.

Schematically speaking, D will consist of a suitable version of H for composing the conclusion of the deduction, four rules (∧,∨,∃,∀) for collecting information about unspecified program segments, and a "mirror image" of H containing inverse rules for decomposing complex program segments appearing among the premises. D, having the flavour of a natural deduction system, has all valid formulae of L as axioms.

II. The System.

Syntax

The alphabet Σ contains symbols for individual constants and variables, functions and predicates, connectives and operators. L=L(Σ) is a logical language with equality over Σ (having at least the power of the first order language over Σ). A well-formed formula of L will be called a logical wff (L-wff). P=P(Σ) is a programming language over Σ, with the following syntax:

<statement>::=<elementary action> | <procedure call> |
<statement>;<statement> |
if<boolean>then<statement>else<statement>fi |
while<boolean>do<statement>od

<declaration>::=<procedure name>(<name parameter list>,
<value parameter list>)proc<statement>end .

An elementary action is a non deterministic assignment of the form $\underline{x} \leftarrow \epsilon \underline{x}' \hat{A}(\underline{x},\underline{x}',\underline{u})$ reading: "assign to $\underline{x}$ some $\underline{x}'$ such that $\hat{A}$ holds". This will usually be abbreviated as $A(\underline{x},\underline{u})$, where $\underline{x}$ is the vector of variables which can be modified by A, and $\underline{u}$ is the vector of additional variables upon which the assignment might depend. When $\hat{A}$ is of the form $\underline{x}'=\underline{t}(\underline{x},\underline{u})$, A is the conventional assignment statement.

A procedure call is a statement of the form call $P(\underline{x},\underline{t})$, where P is a procedure name, $\underline{x}$ is a vector of actual name parameters (variables), and $\underline{t}$ is a vector of actual value parameters (terms). The $\underline{x}$'s are assumed to be distinct and the $\underline{t}$'s to be independant of the $\underline{x}$'s.

A boolean is a quantifier-free L-wff.

A P-segment will simply be a statement in P. We extend Σ to Σ' by adding a set of new symbols $\{R_1,R_2,\ldots\}$ which stand for arbitrary P-segments, and are therefore called program-segment variables (PSV's). The programming language P' is an extension of P obtained by allowing statements of the form $R_i(\underline{x},\underline{u})$, where $\underline{x}$ and $\underline{u}$ have a meaning similar to that given in the elementary actions. Note that the difference between a PSV and an elementary action is that for the latter we are given a formula defining its effect. Similarly, the difference between a PSV and a procedure call is that the latter may have an explicit declaration. We will use $\alpha(\underline{x},\underline{u})$ to denote an arbitrary P'-segment such that $\underline{x}$ is the vector of all modifiable variables of α, and $\underline{u}$ consists of all other variables appearing in α.

A <u>specification</u> is a construct σ of the form σ: $p(\underline{x},\underline{u},\underline{v})\{\alpha(\underline{x},\underline{u})\}q(\underline{x},\underline{u},\underline{v})$, where p and q are L-wffs and α is a P'-segment. Here the elements of $\underline{v}$ are said to be the <u>free variables</u> of the specification σ. Where no confusion can arise we will occasionaly omit the $\underline{v}$'s and regard the $\underline{u}$ as consisting of all the variables appearing in the specification not assigned to in α. A specification $p\{\alpha\}q$ is <u>simple</u> if α is a PSV, an elementary action or a call statement (<u>simple statements</u>).

The formulae of our language W (called <u>W-wff</u>'s) are

(1) L-wffs,
(2) specifications,
(3) declarations.

(Note that W-wffs cannot be combined by logical connectives.)

Semantics

An <u>interpretation</u> of a set Γ of W-wffs is a tuple $I=\langle D,\Sigma,\underline{R}_1,\ldots,\underline{R}_k,\underline{P}_1,\ldots,\underline{P}_m\rangle$, where D is a nonempty domain, Σ is an interpretation of all individuals (including constants and free variables), function and predicate symbols of L, each $\underline{R}_i(\underline{x},\underline{x}',\underline{u})$ is a relation for a PSV $R_i(\underline{x},\underline{u})$ appearing in Γ, and each $\underline{P}_i(\underline{x},\underline{x}',\underline{u})$ is a relation for a procedure P_i that appears in Γ, but does not have a corresponding declaration in Γ. $\underline{R}_i$ and $\underline{P}_i$ describe the effect of the P'-segments R_i and <u>call</u>P_i respectively, under I.

We now show how an interpretation I assigns <u>truth values</u> to W-wffs. An L-wff is assigned a truth value by I in the standard way. A program segment $\alpha(\underline{x},\underline{u})$ all of whose procedure calls are interpreted (see below), is interpreted under I as a relation ρ_α in the following way (relational notation from e.g. deBakker and Meertens[1]):

For an elementary action A $\quad \rho_A = \hat{A}$
(the interpretation Σ gives to $\hat{A}$),

For a PSV R_i $\quad \rho_{R_i} = \underline{R}_i$

For a procedure P_i $\quad \rho_{\underline{call}P_i} = \underline{P}_i$

$\rho_{\alpha;\beta} = \rho_\alpha\rho_\beta$,

$\rho_{\underline{if}\ r\ \underline{then}\ \alpha\ \underline{else}\ \beta\ \underline{fi}} = r\rho_\alpha \cup r'\rho_\beta$,

$\rho_{\underline{while}\ r\ \underline{do}\ \alpha\ \underline{od}} = (r\rho_\alpha)^* r'$.

Using this definition, we are now able to assign relations to the procedure calls which <u>have</u> corresponding body declarations in Γ. The relations assigned to these procedures are the least fixpoint relations solving the system of mutually recursive procedure declarations in Γ (here too we will refer to this interpretation of such P as $\underline{P}$). We now have an interpretation under I for each P'-segment in Γ.

A specification $p(\underline{x},\underline{u},\underline{v})\{\alpha(\underline{x},\underline{u})\}q(\underline{x},\underline{u},\underline{v})$ is <u>true</u> under I if $\forall\underline{x},\underline{u}(p(\underline{x},\underline{u},\underline{v})\wedge\rho_\alpha(\underline{x},\underline{x}',\underline{u}) \supset q(\underline{x}',\underline{u},\underline{v}))$ is true (note that the free variables $\underline{v}$ have been assigned by I).

A set Γ of W-wffs is defined to be <u>true</u> under an interpretation I of Γ, if all non-declaration formulae of Γ are true in I. I is called a <u>model</u> of Γ.

A tuple $S=(\sigma_1,\ldots\sigma_n,\sigma)$ where σ is not a declaration, is called a <u>valid deduction</u> (written $\sigma_1,\ldots,\sigma_n \models \sigma$), if σ is true in any interpretation I of S which is a model of $\{\sigma_1,\ldots,\sigma_n\}$.

We denote a P'-segment α containing the statements $\underline{call}P_1(\underline{x}_1,\underline{t}_1),\ldots,\underline{call}P_n(\underline{x}_n,\underline{t}_n)$ by $\alpha(\text{"}\underline{call}P_1\text{"},\ldots,\text{"}\underline{call}P_n\text{"})$, and the elementary action $\underline{x}\leftarrow\epsilon\underline{x}'(\phi(\underline{x},\underline{t})\supset\psi(\underline{x}',\underline{t}))$ by $[\phi,\psi](\underline{x},\underline{t})$.

We now present our system D. The basic statements to be proved in D are deductions of the form $\Gamma\vdash\sigma$ where Γ is a set of W-wffs, and σ a non-declaration W-wff. Our inference rules are rule-schemata in which $\alpha,\beta,\ldots$ stand for arbitrary P'-segments and p,q,... for arbitrary L-wffs.

AXIOMS

A1 $\quad \vdash p$

where p is any logically valid L-wff.

A2 $\quad \Gamma,\sigma\vdash\sigma$

where σ is not a declaration.

A3 Frame axiom

$$\vdash p(\underline{v})\{\alpha(\underline{x},\underline{u})\}p(\underline{v})$$

where $\underline{v}$ and $\underline{x}$ are disjoint.

RULES OF INFERENCE

L1 Introduction

$$\frac{\Gamma \vdash \sigma_1}{\Gamma, \sigma_2 \vdash \sigma_1}$$

L2 Modus Ponens

$$\frac{\Gamma, \sigma_2 \vdash \sigma_1 \quad , \quad \Gamma \vdash \sigma_2}{\Gamma \vdash \sigma_1}$$

where p and q are L-wffs.

L3 Deduction

$$\frac{\Gamma \, , \, p \vdash q}{\Gamma \vdash p \supset q} \quad \text{and} \quad \frac{\Gamma \vdash p \supset q}{\Gamma \, , \, p \vdash q}$$

where p and q are L-wffs.

D1 Elementary Action

$$\frac{\Gamma \vdash p(\underline{x},\underline{u},\underline{v}) \wedge \hat{A}(\underline{x},\underline{x}',\underline{u}) \supset q(\underline{x}',\underline{u},\underline{v})}{\Gamma \vdash p(\underline{x},\underline{u},\underline{v}) \{A(\underline{x},\underline{u})\} q(\underline{x},\underline{u},\underline{v})}$$

D2 Consequence

$$\frac{\Gamma \vdash p \supset s \; , \; \Gamma \vdash s\{\alpha\}r \; , \; \Gamma \vdash r \supset q}{\Gamma \vdash p\{\alpha\}q}$$

D3 Composition

$$\frac{\Gamma \vdash p\{\alpha\}s \; ; \; \Gamma \vdash s\{\beta\}q}{\Gamma \vdash p\{\alpha;\beta\}q}$$

D4 Conditional

$$\frac{\Gamma \vdash p \wedge r\{\alpha\}q \; , \; \Gamma \vdash p \wedge \neg r\{\beta\}q}{\Gamma \vdash p\{\underline{\text{if}}\ r\ \underline{\text{then}}\ \alpha\ \underline{\text{else}}\ \beta\ \underline{\text{fi}}\}q}$$

D5 Iteration

$$\frac{\Gamma \vdash p \wedge r\{\alpha\}p}{\Gamma \vdash p\{\underline{\text{while}}\ r\ \underline{\text{do}}\ \alpha\ \underline{\text{od}}\}p \wedge \neg r}$$

D6 Substitution

(a)
$$\frac{\Gamma \vdash p(\underline{x},\underline{u})\{\alpha(\underline{x},\underline{u})\}q(\underline{x},\underline{u})}{\Gamma \vdash p(\underline{z},\underline{u})\{\alpha(\underline{z},\underline{u})\}q(\underline{z},\underline{u})}$$

where $\underline{z}$ is disjoint from $\underline{u}$ and is free for $\underline{x}$ in p and q.

(b)
$$\frac{\Gamma \vdash p(\underline{x},\underline{u})\{\alpha(\underline{x},\underline{u})\}q(\underline{x},\underline{u})}{\Gamma \vdash p(\underline{x},\underline{t})\{\alpha(\underline{x},\underline{t})\}q(\underline{x},\underline{t})}$$

where $\underline{t}$ is a vector of terms which is free for $\underline{u}$ in p and q, and does not depend on $\underline{x}$.

D7 Recursion

$$\frac{\Gamma([\phi,\psi]) \vdash \phi\{\alpha([\phi,\psi])\}\psi}{\Gamma(\text{"}\underline{\text{call}}\,P\text{"}) \; , \; P\ \underline{\text{proc}}\ \alpha(\text{"}\underline{\text{call}}\,P\text{"})\,\underline{\text{end}} \vdash \phi\{\underline{\text{call}}\,P\}\psi}$$

(Here $\Gamma([\phi,\psi])$ is Γ with the elementary action $[\phi,\psi](\underline{z},\underline{t})$ substituted for occurencies of $\underline{\text{call}}\,P(\underline{z},\underline{t})$. A clarification of rule D7 appears at the end of the Section.)

D8 $\wedge$-rule

$$\frac{\Gamma \vdash p\{\alpha\}q_1, \ldots, \Gamma \vdash p\{\alpha\}q_n}{\Gamma \vdash p\{\alpha\} \bigwedge_{i=1}^{n} q_i} \qquad n \geq 0$$

D9 $\vee$-rule

$$\frac{\Gamma \vdash p_1\{\alpha\}q, \ldots, \Gamma \vdash p_n\{\alpha\}q}{\Gamma \vdash \bigvee_{i=1}^{n} p_i\{\alpha\}q} \qquad n \geq 0$$

(D8 and D9 reduce to $\Gamma \vdash p\{\alpha\}\underline{\text{true}}$ and $\Gamma \vdash \underline{\text{false}}\{\alpha\}q$ respectively when n=0).

D10 $\forall$-rule

$$\frac{\Gamma \vdash p\{\alpha\}q}{\Gamma \vdash p\{\alpha\}(\forall \underline{u})q}$$

$\underline{u}$ not free in p or Γ, and does not appear in α.

D11 $\exists$-rule

$$\frac{\Gamma \vdash p\{\alpha\}q}{\Gamma \vdash (\exists \underline{u})p\{\alpha\}q}$$

$\underline{u}$ not free in q, and does not appear in α.

D12 Inverse Elementary Action

$$\frac{\Gamma \;,\; p(\underline{x},\underline{u},\underline{v}) \wedge A(\underline{x},\underline{x}',\underline{u}) \supset q(\underline{x}',\underline{u},\underline{v}) \vdash \sigma}{\Gamma \;,\; p(\underline{x},\underline{u},\underline{v})\{A(\underline{x},\underline{u})\}q(\underline{x},\underline{u},\underline{v}) \vdash \sigma}$$

D13 Inverse Composition

$$\frac{\Gamma,\; p\{\alpha\}\lambda,\; \lambda\{\beta\}q \vdash \sigma}{\Gamma,\; p\{\alpha;\beta\}q \vdash \sigma}$$

where λ does not appear in any other component of the rule.

Note that D13 (and similarly for the other inverse rules) is an indirect way of expressing the more natural

$$\frac{\Gamma \vdash p\{\alpha;\beta\}q}{\Gamma \vdash \exists\lambda(p\{\alpha\}\lambda \wedge \lambda\{\beta\}q)}$$

the conclusion of which is unfortunately not well formed in W.

D14 Inverse Conditional

$$\frac{\Gamma \;,\; p \wedge r\{\alpha\}q \;,\; p \wedge \neg r\{\beta\}q \vdash \sigma}{\Gamma \;,\; p\{\underline{\text{if}}\ r\ \underline{\text{then}}\ \alpha\ \underline{\text{else}}\ \beta\ \underline{\text{fi}}\}q \vdash \sigma}$$

D15 Inverse Iteration

$$\frac{\Gamma \;,\; p \supset \lambda \;,\; \lambda \wedge r\{\alpha\}\lambda \;,\; \lambda \wedge \neg r \supset q \vdash \sigma}{\Gamma \;,\; p\{\underline{\text{while}}\ r\ \underline{\text{do}}\ \alpha\ \underline{\text{od}}\}q \vdash \sigma}$$

where λ does not appear in any other component of the rule.

D16 Inverse Recursion

$$\frac{\Gamma([\delta,\lambda]) \;,\; \delta\{\alpha([\delta,\lambda])\}\lambda \vdash \sigma}{\Gamma(\text{"}\underline{\text{call}}P\text{"}) \;,\; P\ \underline{\text{proc}}\ \alpha(\text{"}\underline{\text{call}}P\text{"})\underline{\text{end}} \vdash \sigma}$$

where δ and λ do not appear in any other component of the rule.

A proof in $\mathcal{D}$ is a sequence of deductions $\Gamma_i \vdash \sigma_i$ $i=1,2,\ldots$, where any line (i.e. deduction) is an axiom or is derived from previous lines by one of the inference rules. A deduction $\Gamma \vdash \sigma$ is said to be derivable in $\mathcal{D}$ (written $\Gamma \vdash_{\mathcal{D}} \sigma$) if it is a line of a proof in $\mathcal{D}$.

Our formulation of D7 employs the substitution of $[\phi,\psi]$ for "$\underline{\text{call}}$P" in the proof of the body α. This corresponds to the familiar notion of assuming $\phi\{\underline{\text{call}}P\}\psi$ when proving α. Employing the same substitution for the premises used in proving α, provides us with a concise way of constructing a recursion rule for mutually recursive procedures which avoids refering to all n procedures (as is done in [7] and [9]). In order to illustrate the way in which D7 (and similarly D16) is used, consider two procedures P_1 and P_2, with declarations P_i $\underline{\text{proc}}$ $\alpha_i(\text{"}\underline{\text{call}}P_1\text{"},\text{"}\underline{\text{call}}P_2\text{"})\underline{\text{end}}$ $1 \le i \le 2$. A framework for a proof of a $\underline{\text{call}}P_1$-specification is:

(1) $\vdash \phi_2\{\alpha_2([\phi_1,\psi_1],[\phi_2,\psi_2])\}\psi_2$,

(2) $P_2\ \underline{\text{proc}}\ \alpha_2([\phi_1,\psi_1],\text{"}\underline{\text{call}}P_2\text{"})\underline{\text{end}}$
$\vdash \phi_2\{\underline{\text{call}}P_2\}\psi_2$,

(3) $P_2\ \underline{\text{proc}}\ \alpha_2([\phi_1,\psi_1],\text{"}\underline{\text{call}}P_2\text{"})\underline{\text{end}}$
$\vdash \phi_1\{\alpha_1([\phi_1,\psi_1],\underline{\text{call}}P_2)\}\psi_1$,

(4) $P_1\ \underline{\text{proc}}\ \alpha_1\ \underline{\text{end}}\ ,\ P_2\ \underline{\text{proc}}\ \alpha_2\ \underline{\text{end}}$
$\vdash \phi_1\{\underline{\text{call}}P_1\}\psi_1$.

Lines (2) and (4) are proved using D7 with an empty Γ, and Γ consisting of the P_2-declaration respectively.

The following (standardly verified) fact is very useful in proving deductions involving unspecified program segments:

Substitution Theorem - If $\Gamma \vdash_{\mathcal{D}} \sigma$, and Γ' and σ' are obtained by replacing all occurencies of a PSV R by an arbitrary P'-segment in Γ and σ, then $\Gamma' \vdash_{\mathcal{D}} \sigma'$. ∎

III. Results.

One of our basic assumptions throughout, is that the language L is expressive (Cook[3]). This means that for each P-segment α in the context of a given set of declarations, it is possible to express as an L-wff the relation ρ_α computed by α, i.e. L has constructs powerful enough to express the $*$, $\cup$, composition and fixpoint operators. A special important case of an expressive language is (as pointed out by Cook) first order arithmetic.

All subsystems considered in this section have A1-A3 as axioms, L1-L3 as logical rules and differ only in their D-rules.

Consider the system *D1* which consists of rules D1-D5. *D1* is a version of the usual Hoare system for proving partial corectness of programs with regular control structure, and for it we have the following result (proved e.g. in [1,3,9,15]):

<u>Theorem 1</u> - If $\sigma_1,\ldots,\sigma_n$ are L-wffs, then

$$\sigma_1,\ldots,\sigma_n \models \sigma \quad \text{iff} \quad \sigma_1,\ldots,\sigma_n \vdash_{D1} \sigma \;. \quad \blacksquare$$

Consider *D2* consisting of D1-D7. This is an extension of Hoare's method to deal with mutually recursive procedures. A proof of Theorem 2 can be found for similar versions in Harel et al[9] or Gorelick[7].

<u>Theorem 2</u> - If $\sigma_1,\ldots,\sigma_n$ are L-wffs or procedure declarations, then

$$\sigma_1,\ldots,\sigma_n \models \sigma \quad \text{iff} \quad \sigma_1,\ldots,\sigma_n \vdash_{D2} \sigma \;. \quad \blacksquare$$

We now consider *D3* which consists of rules D1-D6 and D8-D12.

<u>Theorem 3</u> - If $\sigma_1,\ldots,\sigma_n$ are L-wffs or simple specifications, then

$$\sigma_1,\ldots,\sigma_n \models \sigma \quad \text{iff} \quad \sigma_1,\ldots,\sigma_n \vdash_{D3} \sigma \;.$$

(<u>Note</u> - In this and the following theorems we omit the proofs of the soundness direction. The reader is urged to convince himself that the rules are indeed sound, a rigorous proof of this would be based on Scott's induction principle in a standard way. Rather, the proofs presented are designed to demonstrate the completeness direction in a constructive manner).

<u>Proof</u> - Given a valid deduction w: $\sigma_1,\ldots,\sigma_n \models \sigma$ we reduce the problem as follows:

(1) The absence of procedure declarations among the premises means that each <u>call</u> statement can be regarded as a new PSV. This follows from the arbitrary interpretations both PSV's and <u>call</u>'s can take on, in a model of $\sigma_1,\ldots,\sigma_n$.

(2) Use rule D12 to replace every elementary-statement specification $p\{A\}q$ by an L-wff. (Here, as well as at other points in the paper, we describe the natural order of the derivation. Formally, this application of rule D12, for example, appears at the end of the proof in *D*. Nevertheless, we may think of this stage as being first in the derivation process.) We are left with premises consisting of PSV-specifications of the form $p\{R\}q$, and L-wffs. Denote by τ the conjunction of the latter. Formally, τ can be derived by using D8 with the identity program and p=<u>true</u>.

(3) If σ is an L-wff then the validity of the deduction w is equivalent to the validity of some L-wff. This can be seen by considering an interpretation in which each PSV is assigned the empty relation. In this case all specification premises hold and therefore we must have $\tau \models \sigma$, which is equivalent to the validity of $\tau \supset \sigma$, which in turn is an axiom in A1. Using L2, σ is obtained.

(4) Employing a similar argument with an interpretation assigning the empty relation to all PSV's not appearing in σ, we can omit any PSV-specification for a PSV not appearing in σ. We are now left with a situation of the form

$$\tau \;,\; R_1\text{-specifications} \;,\ldots,\; R_k\text{-specifications} \;\vdash\; p\{\alpha(R_1,\ldots,R_k)\}q$$

where α is a P'-segment involving PSV's $R_1,\ldots,R_k$. Denote the specification premises by Γ. These premises contain all available information about $R_1,\ldots,R_k$. We therefore construct for each $1\le i\le k$ an "approximation from above" μ_{R_i} to the relation computed by R_i. μ_{R_i} will be an L-wff which can easily be seen to be true in any model of Γ , and hence in any model of $\{\tau,\Gamma\}$. This is the sense in which it is an approximation. We will simplify notation by refering to the case where k=1 and to R_1 as R, with the understanding that the following can be done for all k PSV's for any k.

Assume that Γ is the set $p_j(\underline{x},\underline{u},\underline{v})\{R(\underline{x},\underline{u})\}q_j(\underline{x},\underline{u},\underline{v})$ $\quad 1\le j\le m$. This can be brought about by using D6 and collecting free variables in $\underline{v}$. Define

$$\mu_R(\underline{x},\underline{x}',\underline{u})=\forall\underline{v}\bigwedge_{j=1}^{m}(p_j(\underline{x},\underline{u},\underline{v})\supset q_j(\underline{x}',\underline{u},\underline{v}))$$

Clearly μ_R serves to "collect information" about the PSV R.

Define A_R as the elementary action $\underline{x}\leftarrow\epsilon\underline{x}'\mu_R(\underline{x},\underline{x}',\underline{u})$. Obviously $\mu_{A_R}=\mu_R$.
From the way A_R was defined, it is clear that for every j we have $\models p_j\{A_R\}q_j$.

Thus under the substitution that replaces the PSV R by the P-segment A_R, every interpretation satisfying τ also satisfies Γ, and therefore also satisfies $p\{\alpha(A_R)\}q$. Hence $\tau\models p\{\alpha(A_R)\}q$, and by Theorem 1 there exists a proof

(*) $\quad \tau\vdash_{D1} p\{\alpha(A_R)\}q$.

Without loss of generality (having in mind the standard techniques used in proving Theorem 1, in e.g. [1,3,9,15]), we may assume that in the process of proving the deduction (*) in *D1*, the strongest consequent approach was adopted, in which every subderivation of a simple A_R-specification is preceeded by a derivation of a specification of the form $s\{A_R\}s\circ\mu_R$ for some s, where for $s(\underline{x},\underline{u})$ and $\mu_R(\underline{x},\underline{x}',\underline{u})$ we define $s\circ\mu_R(\underline{x},\underline{u})=\exists\underline{x}'(s(\underline{x}',\underline{u})\wedge\mu_R(\underline{x}',\underline{x},\underline{u}))$. (See e.g. [1]).

If we now manage to replace every such subproof by a proof in *D3* of $s\{R\}s\circ\mu_R$ from assumptions Γ and substitute R for A_R elsewhere, then this modified proof of (*) serves as our proof of $\Gamma,\tau\vdash_{D3} p\{\alpha(R)\}q$. Indeed this can be done using the following four derived rules of *D3*:

D8' $\wedge\wedge$-rule

$$\frac{\Gamma\vdash p_1\{\alpha\}q_1,\ldots,\ \Gamma\vdash p_n\{\alpha\}q_n}{\Gamma\vdash\bigwedge_{i=1}^{n}p_i\{\alpha\}\bigwedge_{i=1}^{n}q_i}\qquad n\geq 0$$

D9' $\vee\vee$-rule

$$\frac{\Gamma\vdash p_1\{\alpha\}q_1,\ldots,\ \Gamma\vdash p_n\{\alpha\}q_n}{\Gamma\vdash\bigvee_{i=1}^{n}p_i\{\alpha\}\bigvee_{i=1}^{n}q_i}\qquad n\geq 0$$

D10' $\forall\forall$-rule

$$\frac{\Gamma\vdash p\{\alpha\}q}{\Gamma\vdash(\forall\underline{u})p\{\alpha\}(\forall\underline{u})q}$$

$\underline{u}$ not free in Γ, and does not appear in α.

D11' $\exists\exists$-rule

$$\frac{\Gamma\vdash p\{\alpha\}q}{\Gamma\vdash(\exists\underline{u})p\{\alpha\}(\exists\underline{u})q}$$

$\underline{u}$ does not appear in α.

Now (for any s) replace every subproof of $s\{A_R\}s\circ\mu_R$ in the proof of (*) by:

$\Gamma\vdash p_j(\underline{x}',\underline{u},\underline{v})\supset p_j(\underline{x},\underline{u},\underline{v})\{R(\underline{x},\underline{u})\}p_j(\underline{x}',\underline{u},\underline{v})\supset q_j(\underline{x},\underline{u},\underline{v})$

for every $1\leq j\leq m$. (Use A3 with $\neg p_j(\underline{x}',\underline{u},\underline{v})$, and D8')

$\Gamma\vdash\forall\underline{v}(p_j(\underline{x}',\underline{u},\underline{v})\supset p_j(\underline{x},\underline{u},\underline{v}))\{R(\underline{x},\underline{u})\}\forall\underline{v}(p_j(\underline{x}',\underline{u},\underline{v})\supset q_j(\underline{x},\underline{u},\underline{v}))$ (D10')

$\Gamma\vdash s(\underline{x}',\underline{u})\wedge\bigwedge_{j=1}^{m}\forall\underline{v}(p_j(\underline{x}',\underline{u},\underline{v})\supset p_j(\underline{x},\underline{u},\underline{v}))\{R(\underline{x},\underline{u})\}$
$s(\underline{x}',\underline{u})\wedge\bigwedge_{j=1}^{m}\forall\underline{v}(p_j(\underline{x}',\underline{u},\underline{v})\supset q_j(\underline{x},\underline{u},\underline{v}))$

(Use A3 with $s(\underline{x}',\underline{u})$, and D9')

$\Gamma\vdash\exists\underline{x}'(s(\underline{x}',\underline{u})\wedge\bigwedge_{j=1}^{m}\forall\underline{v}(p_j(\underline{x}',\underline{u},\underline{v})\supset p_j(\underline{x},\underline{u},\underline{v})))\{R(\underline{x},\underline{u})\}$
$\exists\underline{x}'(s(\underline{x}',\underline{u})\wedge\bigwedge_{j=1}^{m}\forall\underline{v}(p_j(\underline{x}',\underline{u},\underline{v})\supset q_j(\underline{x},\underline{u},\underline{v})))$ (D11')

$\Gamma\vdash s(\underline{x},\underline{u})\supset\exists\underline{x}'(s(\underline{x}',\underline{u})\wedge\bigwedge_{j=1}^{m}\forall\underline{v}(p_j(\underline{x}',\underline{u},\underline{v})\supset p_j(\underline{x},\underline{u},\underline{v})))$

(A1 and L1)

$\Gamma\vdash s(\underline{x},\underline{u})\{R(\underline{x},\underline{u})\}(s\circ\mu_R)(\underline{x},\underline{u})$ (D2). ∎

We remark here that restricting the premises to have no free variables not appearing in α (i.e. no $\underline{v}$), makes possible a different proof of Theorem 3 which does not use rules D10-D11.

We now consider *D4* consisting of D1-D11.

Theorem 4 - If $\sigma_1,\ldots,\sigma_n$ are L-wffs, simple specifications or declarations for procedure names not appearing in these simple specifications (but possibly in σ, and in other declarations), then

$$\sigma_1,\ldots,\sigma_n \models \sigma \quad \text{iff} \quad \sigma_1,\ldots,\sigma_n \vdash_{D4} \sigma \ .$$

Proof - Assume given w: $\sigma_1,\ldots,\sigma_n \models \sigma$, with procedure declarations P_i proc α_i end $1\le i\le m$, among the premises. $D4$ illustrates the extra feature of call's (in σ) to procedures with given bodies, thus forcing the use of D7. We will find a similar approximation μ_{P_i} for each such procedure. As before regard each call to a procedure other than the P_i's as a new PSV. We now construct μ_R for every PSV R, and as above, substitute A_R for each appearance of R in w. Denote the resulting modified body of P_i by α_i', and modified σ by σ'.

This system of m PSV-free declarations now gives rise to a least-fixpoint solution, in the form of m relations. Denote the L-wff equivalents to these relations by μ_{P_i} $1\le i\le m$. Define A_{P_i} to be the elementary action $\underline{x}\leftarrow\underline{x}'\epsilon\mu_{P_i}(\underline{x},\underline{x}',\underline{u})$. (For clarity throughout this proof we omit indices of $\underline{x},\underline{x}'$ and $\underline{u}$.). Denoting as before by τ and Γ the L-wff and specification premises respectively, we now observe that any interpretation I satisfying τ, satisfies (substituted) Γ. Recalling the definition of the relation that I assigns to each P_i, we have $\tau \models \sigma''$, where σ'' is σ' further modified by substituting A_{P_i} for callP_i, $1\le i\le m$. Therefore there exists a proof

$$(**) \qquad \tau \vdash_{D1} \sigma''.$$

Denoting the declaration premises of w by Π, we will obtain a proof of w in $D4$ by first replacing (in the proof of (**)) subproofs of $\tau \vdash_{D1} s\{A_{P_i}\}s\cdot\mu_{P_i}$ by proofs of $\tau,\Pi \vdash_{D4} s\{\text{call}P_i\}s\cdot\mu_{P_i}$, and then dealing with PSV's as in Theorem 3. We will really show how $\underline{x}=\underline{x}_0\{\text{call}P_i\}\mu_{P_i}(\underline{x}_0,\underline{x},\underline{u})$ can be derived in $D4$ from τ and Π, where $\underline{x}_0$ is a vector of new symbols. Easy applications of A3,D8' and D11' will give $s\{\text{call}P_i\}s\cdot\mu_{P_i}$.

We prove that $\Pi,\tau \vdash_{D4} \underline{x}=\underline{x}_0\{\text{call}P_i\}\mu_{P_i}(\underline{x}_0,\underline{x},\underline{u})$ by induction on m. For m≥1 assume that if Π contains m-1 declarations $P_1,\ldots,P_{m-1}$ (denoted $\Pi^{(m-1)}$), then for every $1\le i\le m-1$

$$\Pi^{(m-1)},\tau \vdash_{D4} \underline{x}=\underline{x}_0\{\text{call}P_i\}\mu_{P_i}(\underline{x}_0,\underline{x},\underline{u}).$$

Given $\Pi^{(m)}$, consider the first m-1 declarations with A_{P_m} substituted for "callP_m" (denote this by $\Pi^{(m-1)}(A_{P_m})$). It is not difficult to see that

$$\tau\ ,\{\underline{x}=\underline{x}_0\{\text{call}P_i\}\mu_{P_i}(\underline{x}_0,\underline{x},\underline{u}) \quad 1\le i\le m-1\} \models \underline{x}=\underline{x}_0\{\underline{\alpha}_m(A_{P_m})\}\mu_{P_m}(\underline{x}_0,\underline{x},\underline{u}),$$

and hence by Theorem 3

$$\tau\ ,\{\underline{x}=\underline{x}_0\{\text{call}P_i\}\mu_{P_i}(\underline{x}_0,\underline{x},\underline{u}) \quad 1\le i\le m-1\} \vdash_{D3} \underline{x}=\underline{x}_0\{\underline{\alpha}_m(A_{P_m})\}\mu_{P_m}(\underline{x}_0,\underline{x},\underline{u}).$$

However by the inductive hypothesis, for every $1\le i\le m-1$

$$\tau\ ,\ \Pi^{(m-1)}(A_{P_m}) \vdash_{D4} \underline{x}=\underline{x}_0\{\text{call}P_i\}\mu_{P_i}(\underline{x}_0,\underline{x},\underline{u}).$$

We therefore have

$$\tau\ ,\ \Pi^{(m-1)}(A_{P_m}) \vdash_{D4} \underline{x}=\underline{x}_0\{\underline{\alpha}_m(A_{P_m})\}\mu_{P_m}(\underline{x}_0,\underline{x},\underline{u}),$$

and applying rule D7 we obtain

$$\tau\ ,\ \Pi^{(m)} \vdash_{D4} \underline{x}=\underline{x}_0\{\text{call}P_m\}\mu_{P_m}(\underline{x}_0,\underline{x},\underline{u}). \ \blacksquare$$

The process described in the last two theorems can be summarized as a process for "composing" a complex conclusion from simple premises. We now begin the process of "decomposing" complex premises.

Consider $D5$, consisting of rules D1-D15.

Theorem 5 - If $\sigma_1,\ldots,\sigma_n$ are as in Theorem 4 without the requirement that specifications be simple, then

$$\sigma_1,\ldots,\sigma_n \models \sigma \quad \text{iff} \quad \sigma_1,\ldots,\sigma_n \vdash_{D5} \sigma \ .$$

Proof - All non-simple specifications among the premises are decomposed using rules D12-D15 (see remark after Theorem 6) to obtain only simple specifications (the validity of the deduction implies that the new symbols introduced at this stage will disappear in the process of deriving σ). Theorem 4 can now be applied. $\blacksquare$

Our main result is

<u>Theorem 6</u> -

$$\sigma_1,\dots,\sigma_n \models \sigma \quad \text{iff} \quad \sigma_1,\dots,\sigma_n \vdash_D \sigma \ .$$

<u>Proof</u> - The only new feature here is the possibility of having <u>call</u> statements among the specification premises, with given declarations (implying that their "meaning" is fixed, and they can no longer be regarded as PSV's). Rule D16 is applied to all such procedures, effectively getting rid of the <u>call</u>'s, and "trading" them in for new body-specifications. The situation is now precisely that described in the hypothesis of Theorem 5. Here too the validity of the original deduction implies that the new symbols ϕ and λ (standing for the least fixpoints) will disappear in the derivation process. ▮

Note the decompose-collect-compose symmetry of the entire derivation process described in the above theorems:

(1) "trade" <u>call</u>'s for bodies
(2) decompose bodies and premises
(3) collect PSV information
(4) compose bodies
(5) "trade" bodies for <u>call</u>'s
(6) compose conclusion.

As remarked above, step (2) shows up in a formal proof as the composition of the premises. This is a consequence of the deductive character of ***D***, the decomposed premises being "carried along" throughout the derivation and composed towards the end. However, we prefer to regard this step as "decomposition" because it is usually carried out first in a manner similar to subgoaling. A glance at the proof in the Appendix might help clarify this remark.

We remark here that restricting L to be first order can destroy the completeness, as shown in [9], a result which reminds one of (and in fact subsumes, and as such provides a new proof of) Wands result [16]. This result and the rather obvious fact that if L is weak second order then it is expressive, should now be clarified by the hierarchy result appearing in [8].

IV. The Power of *D*.

We will try to be slightly more specific about our claims as to what can be done in ***D***.

(i) (partial correctness) Given a <u>program</u> $(P_1,\dots,P_n,\alpha)$ consisting of n declarations and a statement α, and some L-wffs $\tau_1,\dots,\tau_m$, a proof that the program is partially correct with respect to p and q, assuming the τ_i are true, is carried out simply by proving in ***D***

$$P_1,\dots,P_n,\ \tau_1,\dots,\tau_m \ \vdash p\{\alpha\}q$$

(ii) (total correctness), Given a program and L-wffs as in (i), a proof that α is totally correct assuming the L-wffs true, can be carried out by proving in ***D***

$$P_1,\dots,P_n,\tau_i,\dots,\tau_m, \quad p(\underline{x},\underline{u},\underline{v})\wedge\lambda(\underline{x},\underline{u})\{\alpha(\underline{x},\underline{u})\}\neg q(\underline{x},\underline{u},\underline{v}) \ \vdash \ \forall\underline{x},\underline{u}(\neg\lambda(\underline{x},\underline{u})).$$

Another way is by using constant symbols $(\underline{a},\underline{b})$ and proving in ***D***

$$P_1,\dots,P_n,\tau_i,\dots,\tau_m, \quad p(\underline{a},\underline{b},\underline{v}) \ , \ (\underline{a},\underline{b})=(\underline{x},\underline{u})\{\alpha(\underline{x},\underline{u})\}\neg q(\underline{x},\underline{u},\underline{v}) \ \vdash \underline{false}.$$

We wish to clarify this somewhat suprising result as related to the commonly accepted view that termination of programs with loops or recursion must employ some form of induction on a well founded set. The fact is that the induction has been buried deep in L, and its utilization is no longer the concern of the user of ***D***. Rather, an inductive argument might be handy when the valid formulae (taken by us as axioms A1) are to be proved in L. We illustrate this point. Take L to be the language of arithmetic, and prove that α: <u>while</u> x>0 <u>do</u> x←x-1 <u>od</u> is totally correct with respect to $p(x)$: $x\geq 0$ and $q(x)$: $x=0$. Subgoaling (using the second formulation above), we obtain the deduction $a\geq 0 \ , \ x=a\{\alpha\}\neg x=0 \vdash \underline{false}$. Applications of D15 and D12 yield $a\geq 0 \ , \ \forall x(x=a\supset\lambda(x)) \ ,$ $\forall x(\lambda(x)\wedge x\leq 0\supset\neg x=0) \ , \ \forall x(\lambda(x)\wedge x>0\supset\lambda(x-1)) \vdash \underline{false}$. This, in turn, is equivalent to proving $(a\geq 0 \wedge\forall x(x=a\supset\lambda(x)) \wedge \forall x(\lambda(x)\wedge x\leq 0\supset\neg x=0) \wedge \forall x(\lambda(x)\wedge x>0\supset\lambda(x-1))) \supset \underline{false}$, a valid L-wff (and hence an axiom of ***D***), which can easily be proved in arithmetic using an induction axiom.

Another complete formal system in which total correctness can be proved is that introduced by Pratt [15] and proved complete in Harel et al [8]. Pratt's approach is to formulate a uniform induction principle explicitly in the system, in the form of a rule which is analogous to D5, and which composes a specification about the loop dual to partial correctness. In ***D***, the dual to D5 is D15 (similarly for recursive calls), which merely "breaks up" the loop, providing all the information the loop specification carries with it, and leaves the rest to the logic of the underlying language.

(iii) (equivalence) Take programs

$$(P_1,\ldots,P_n,\alpha),\ (T_1\ldots,T_m,\beta).$$

Their strong equivalence (see Manna[14]), can be proved in ***D*** by proving

$$P_1,\ldots,P_n,T_1,\ldots,T_m\ ,\ \delta\{\alpha\}\lambda \vdash \delta\{\beta\}\lambda$$

where δ and λ are a new predicate symbols, and proving the dual (with α and β exchanged). For example the reader might care to prove

[P <u>proc</u> <u>if</u> p(x) <u>then</u> x←f(x); <u>call</u>P; <u>call</u>P <u>else</u> x←x <u>fi</u> <u>end</u>, T <u>proc</u> <u>if</u> p(x) <u>then</u> x←f(x); <u>call</u>T <u>else</u> x←x <u>fi</u> <u>end</u>, δ(x) {<u>call</u>P}λ(x)] ⊢ δ(x) {<u>call</u>T}λ(x)

and its dual (a proof of this equivalence is given in the Appendix), or

δ(x,y) {<u>while</u> r(x) <u>do</u> x←f(x) <u>od</u>;<u>while</u> s(y) <u>do</u> y←g(y) <u>od</u>}λ(x,y) ⊢ δ(x,y) {<u>while</u> r(x)∨s(y) <u>do</u> <u>if</u> r(x) <u>then</u> x←f(x) <u>else</u> y←g(y) <u>fi</u> <u>od</u>}λ(x,y)

and its dual. (In both examples we write the elementary action with relation $x'=f(x)$ as $x\leftarrow f(x)$.)

(iv) (derived rules). Here we make use of a meta-theorem which states that if $\sigma_1,\ldots,\sigma_n \vdash \sigma$, then the following is a valid inference rule of ***D***:

$$\frac{\Gamma \vdash \sigma_1,\ldots,\Gamma \vdash \sigma_n}{\Gamma \vdash \sigma}$$

For example proving

p∧r⊃s , p∧¬r⊃q , t∧r⊃s , t∧¬r⊃q , s{α}t ⊢ p{<u>while</u> r <u>do</u> α <u>od</u>}q

in ***D*** , establishes the corresponding derived rule.

(v) (modular proofs). Take as a premiss anything previously established and prove the desired conclusion as a consequent. Sometimes it is possible to denote the established segment by a PSV and make the premises simple, this having the effect of shortening the proof and adding to its clarity.

(vi) (simplification and transformations) Using ***D***, it is possible to validate general program transformations. Once a sufficient set of transformations has been established, this set can then be used to simplify, develop and synthesize correct programs. (See [2],[4],[6] and [13] for the use of such sets). Alternatively ***D*** can be part of a program development system in which the user may create and validate his own transformations and apply them immediately to verified program segments.

Some simple examples of such transformations are

p{<u>if</u> r <u>then</u> α <u>else</u> α <u>fi</u>}q ⊢ p{α}q

p{<u>while</u> ¬p <u>do</u> α <u>od</u>; β}q ⊢ p{β}q

p{<u>if</u> r <u>then</u> α <u>else</u> β <u>fi</u>}q ⊢ p{<u>if</u> ¬r <u>then</u> β <u>else</u> α <u>fi</u>}q

Other examples are transformations for recursion removal (See[2]).

<u>VI. Conclusion.</u>

We have presented a complete system ***D***, in which (besides providing for other important but somewhat less spectacular possibilities) equivalence and partial as well as total correctness of programs can be proved.

The notion of proof from assumptions can be regarded as a natural and important extension of the better known notion of proofs of program correctness using Hoare-like systems. If one chooses to take the view that Hoare's method essentially "cheats" by reducing the problem of proving a partial correctness specification to that of proving a formula of L, then we might say that ***D*** extends the "cheating" too, and reduces the problem of proving a deduction over partial correctness specifications to that of proving a deduction in L, and therefore requires a slightly stronger logical component than is needed in

Hoare's system. The proofs of soundness and completeness of D reduce to the traditional proofs of the same for Hoare's system when D is stripped of its extra features. The relationship can be schematically seen by viewing D in the following pictorial way:

```
D = H ∪ {∧,∨,∃,∀} ∪ H^-1
    ^       ^        ^
    |       |        |
    |       |    decompose
    |       |
    |    collect
    |
 compose
```

Appendix.

We show how to prove the equivalence of the following two procedures:

P proc if p(x) then x←f(x); callP; callP else x←x fi end, and
T proc if p(x) then x←f(x); callT else x←x fi end.

We make use of the following derived rule

DR

$$\frac{\Gamma \vdash p\{\alpha\}q \quad , \quad \Gamma \vdash r\{[p,q]\}s}{\Gamma \vdash r\{\alpha\}s}$$

Define Γ as the set $\{\gamma\wedge\neg p\supset\delta \ , \ \gamma\wedge p\{x\leftarrow fx\}\lambda, \lambda\{[\gamma,\delta]\}\mu \ , \ \mu\{[\gamma,\delta]\}\delta \ , \ \phi\{[\gamma,\delta]\}\psi\}$.

We refer to the declaration of P as P proc...end, and similarly for T.

(1) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \gamma\wedge p\{x\leftarrow fx\}\lambda$ hyp.
(2) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \lambda\{[\gamma,\delta]\}\mu$ hyp.
(3) Γ , true$\{[\gamma,\delta]\}\neg p \vdash$ true$\{[\gamma,\delta]\}\neg p$ hyp.
(4) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \lambda\{[\gamma,\delta]\}\mu\wedge\neg p$ D2,D8(2,3)
(5) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \mu\{[\gamma,\delta]\}\delta$ hyp.
(6) Γ , true$\{[\gamma,\delta]\}\neg p \vdash$
$\forall x,x'(\mu(x)\wedge(\gamma(x)\supset\delta(x'))\supset\delta(x'))$ D12
(7) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \mu\wedge(\gamma\supset\delta)\supset\delta$ A1,L2(with x'=x)
(8) Γ , true$\{[\gamma,\delta]\}\neg p$, $\mu,\neg p,\gamma \vdash (\gamma\wedge\neg p)\supset\delta$ hyp.,L1
(9) Γ , true$\{[\gamma,\delta]\}\neg p$, $\mu,\neg p,\gamma \vdash \delta$
L2(we also use L3, and D8 with the empty program to create a conjunction of the hypotheses)
(10) Γ , true$\{[\gamma,\delta]\}\neg p$, $\mu,\neg p \vdash \gamma\supset\delta$ L3
(11) Γ , true$\{[\gamma,\delta]\}\neg p$, $\mu,\neg p \vdash \delta$ L3,L2(7,10)
(12) Γ , true$\{[\gamma,\delta]\}\neg p \vdash (\mu\wedge\neg p)\supset\delta$ L3(and D8)
(13) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \lambda\{[\gamma,\delta]\}\delta$ D2(4,12)
(14) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \gamma\wedge p\{x\leftarrow fx; [\gamma,\delta]\}\delta$ D3(1,13)
(15) Γ , true$\{[\gamma,\delta]\}\neg p \vdash \gamma\wedge\neg p\{x\leftarrow x\}\delta$ hyp.,D1
(16) Γ , true$\{[\gamma,\delta]\}\neg p \vdash$
$\gamma\{$if p then $x\leftarrow fx; [\gamma,\delta]$ else $x\leftarrow x$ fi$\}\delta$
D4(14,15)
(17) T proc...end , Γ , true$\{[\gamma,\delta]\}\neg p \vdash \gamma\{$callT$\}\delta$ D7
(18) T proc...end , Γ , true$\{[\gamma,\delta]\}\neg p \vdash \phi\{[\gamma,\delta]\}\psi$ hyp.
(19) T proc...end , Γ , true$\{[\gamma,\delta]\}\neg p \vdash \phi\{$callT$\}\psi$ DR
(20) T proc...end , $\gamma\{$if p then $x\leftarrow fx; [\gamma,\delta]; [\gamma,\delta]$ else $x\leftarrow x$ fi$\}\delta$, $\phi\{[\gamma,\delta]\}\psi$, true$\{[\gamma,\delta]\}\neg p$
$\vdash \phi\{$callT$\}\psi$ hyp.,D12,D13,D14
(21) T proc...end , P proc...end , $\phi\{$callP$\}\psi$,
true$\{$callP$\}\neg p \vdash \phi\{$callT$\}\psi$ D16
(22) T proc...end , P proc...end , $\phi\{$callP$\}\psi \vdash$
true$\{$callP$\}\neg p$ (This is proved from P proc...end as a standard partial correctness proof. We omit the details.)
(23) T proc...end , P proc...end , $\phi\{$callP$\}\psi \vdash$
$\phi\{$callT$\}\psi$ L2(21,22).

This establishes one direction. The other is very similar and uses $\Gamma = \{\gamma\wedge\neg p\supset\delta \ , \ \gamma\wedge p\{x\leftarrow fx\}\lambda \ , \ \lambda\{[\gamma,\delta]\}\delta \ , \ \phi\{[\gamma,\delta]\}\psi\}$. We now also need another fact about a call to T besides true$\{[\gamma,\delta]\}\neg p$. The new fact is needed in order to show that the second call to P leaves x unchanged. A suitable specification (which is proved as in line (22) above) is $x=v\wedge\neg p(x)\{[\gamma,\delta]\}x=v$, where v is free. We omit the details of this direction.

Acknowledgements.

We wish to thank Nachum Dershowitz for suggestions following a detailed reading of a previous version of the paper. The first author benefited from many related discussions with Vaughan R. Pratt.

REFERENCES

[1] J. W. de Bakker and L. G. L. T. Meertens, "On the Completeness of the Inductive Assertion Method", Journal of Computer & System Sciences, 11, 323-357, (1975).

[2] R. M. Burstall and J. Darlington, "Some Transformations for Developing Recursive Programs", Proc. International Conference on Reliable Software, LA Calif., (1975).

[3] S. A. Cook, "Soundness and Completeness of an Axiom System for Program Verification",TR-95 (a revision of "Axiomatic and Interpretive Semantics for an Algol Fragment", TR-79, (1975)), Dept. of Computer Science, University of Toronto, Canada, (1976).

[4] J. Darlington, "Application of Program Transformation to Program Synthesis", Proving and Improving Programs, Colloques Iria, (1975).

[5] R. W. Floyd, "Assigning Meaning to Programs", In J.T.Schwartz (ed.)Mathematical Aspects of Computer Science, Proceedings Symp. in Appl. Math. 19, Prov. R.I., American Mathematical Society, 19-32 (1967).

[6] S. L. Gerhart, "Correctness-Preserving Program Transformations", Proc. of the 2nd Symposium on Principles of Programming Languages, Palo Alto, Calif., (1975).

[7] G. A. Gorelick, "A Complete Axiomatic System for Proving Assertions about Recursive and Non-Recursive Programs", TR-75, Dept. of Computer Science, Univ. of Toronto (1975).

[8] D. Harel, A. R. Meyer and V. R. Pratt, "Computability and Completeness in Logics of Programs", Proceedings of 9th Annual ACM Symp. on Theory of Computing, (1977).

[9] D. Harel, A. Pnueli and J. Stavi, "Completeness Issues for Inductive Assertions and Hoare's Method", Technical Report, Dept. of Mathematical Sciences, Tel-Aviv Univ., Israel (1976).

[10] C. A. R. Hoare, "An Axiomatic Basis for Computer Programming", CACM 12, 576-580 (1969).

[11] C. A. R. Hoare, "Procedures and Parameters: An Axiomatic Approach", In E. Engeler(ed.), Symp. on Semantics of Algorithmic Languages, LNM 188, Berlin, Springer, 102-116 (1971).

[12] S. Igarashi, R.L. London and D.C. Luckham, "Automatic Program Verification I: A Logical Basis and its Implementation", Acta Informatica 4, 145-182 (1975).

[13] D. E. Knuth, "Structured Programming with Goto Statements", Computing Surveys, Vol 6, No 4, pp.261-301, (1974).

[14] Z. Manna, "Mathematical Theory of Computation", McGraw Hill, (1974).

[15] V. R. Pratt, "Semantical Considerations on Floyd-Hoare Logic", Proceedings 17th Symp. on Found. of Computer Science, Houston, Texas 109-121, (1976).

[16] M. Wand, "A New Incompleteness Result for Hoare's System", Proceedings 8th ACM Symp. Theory of Computing, 87-91 (1976).