JOURNAL OF COMPUTER AND SYSTEM SCIENCES: 3, 119–127 (1969)

The Correctness of Programs*

ZOHAR MANNA

Computer Science Department, Stanford University, Stanford, California 94305

Received August 23, 1968

ABSTRACT

This paper is concerned with the relationship between the correctness of programs and the satisfiability (or unsatisfiability) of certain formulas of the first-order predicate calculus. Results on the equivalence of programs are also included.

INTRODUCTION

Substantial effort has recently been put into finding methods for proving the correctness of (computer) programs. A program is said to be correct if its execution terminates and yields the desired final result.

In this work we intend to formalize this problem by means of the satisfiability (or unsatisfiability) of certain first-order formulas. More precisely, an algorithm will be described for transforming any program P, of a given class $\{P\}$ of programs, into first-order formulas W_P and $\tilde{W}_P$, such that

(i) W_P is satisfiable *if and only if* either P is correct or P does not terminate, and
(ii) $\tilde{W}_P$ is unsatisfiable *if and only if* P is correct.

A similar result for the equivalence of programs will be presented. Two programs are said to be equivalent if for the same input values both terminate and yield the same final result.[1] The correctness and the equivalence problems are clearly related; for, instead of proving the correctness of a program directly, we may prefer to prove that the program is equivalent to some other program whose correctness is already known.

In order to minimize the preliminary definitions, we shall assume that the reader is familiar with standard conventions regarding the first-order predicate calculus (see, e.g., Mendelson [*6*]). Papers related to the results presented here include those of Floyd [*2*, *3*], Manna [*4*, *5*] and Cooper [*1*].

* The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183).

[1] This differs from the standard definition of equivalence, which is: For the same input values *either* both programs do not terminate *or* both programs terminate and yield the same final result.

Definition of Programs

A *program* P consists of

1. (a) an *input vector* $\mathbf{x}$,
   (b) a *program vector* $\mathbf{y}$,
   (c) an *output vector* $\mathbf{z}$,
2. (a) a nonempty *input domain* $D_{\mathbf{x}}$,
   (b) a nonempty *program domain* $D_{\mathbf{y}}$,
   (c) a nonempty *output domain* $D_{\mathbf{z}}$,
3. an *initial assignment function* $g(\mathbf{x})$, which is a total function mapping $D_{\mathbf{x}}$ into $D_{\mathbf{y}}$, and
4. a sequence of N ($N \geqslant 1$) statements, where the i-th *statement* ($1 \leqslant i \leqslant N$) is of the form

$$i : \textit{if } p_i(\mathbf{x}, \mathbf{y}) \textit{ then } [\mathbf{y} \leftarrow f_i^1(\mathbf{x}, \mathbf{y}); \textit{go to } i_1]$$
$$\textit{else } [\mathbf{y} \leftarrow f_i^2(\mathbf{x}, \mathbf{y}); \textit{go to } i_2],$$

where,

(a) $p_i(\mathbf{x}, \mathbf{y})$ is a total predicate over $D_{\mathbf{x}} \times D_{\mathbf{y}}$,[2]
(b) $f_i^1(\mathbf{x}, \mathbf{y})$ and $f_i^2(\mathbf{x}, \mathbf{y})$ are total functions mapping $D_{\mathbf{x}} \times D_{\mathbf{y}}$ into $D_{\mathbf{y}}$, and
(c) $1 \leqslant i_1, \quad i_2 \leqslant N$.

Each *go to* i_k instruction $[\mathbf{y} \leftarrow f_i^k(\mathbf{x}, \mathbf{y}); \textit{go to } i_k]$, $k = 1$ or 2, may be replaced by a *halt* instruction $[\mathbf{z} \leftarrow h_i^k(\mathbf{x}, \mathbf{y}); \textit{halt}]$, where $h_i^k(\mathbf{x}, \mathbf{y})$ is a total function mapping $D_{\mathbf{x}} \times D_{\mathbf{y}}$ into $D_{\mathbf{z}}$.

Execution of Programs

Given a program P and an input value $\xi \in D_{\mathbf{x}}$, for the input vector $\mathbf{x}$, the program can be executed. Execution always starts at the first statement (labeled 1) after initializing the value of $\mathbf{y}$ to be $g(\xi)$.

In general, if we reach the i-th statement ($1 \leqslant i \leqslant N$) with $\mathbf{y} = \eta$, then the execution performed is as follows:

if $p_i(\xi, \eta) = \mathrm{T}$ *then* [replace the value of $\mathbf{y}$ by $f_i^1(\xi, \eta)$ and go to the i_1-th statement]
else [replace the value of $\mathbf{y}$ by $f_i^2(\xi, \eta)$ and go to the i_2-th statement],

[2] i.e., for every pair of elements $(\xi, \eta) \in D_{\mathbf{x}} \times D_{\mathbf{y}}$ the value of $p_i(\xi, \eta)$ is either T (True) or F (False).

or if a *go to* i_k instruction was replaced by a *halt* instruction, the execution performed is: Assign $h_i^k(\xi, \eta)$ to $\mathbf{z}$ and halt.

If the execution terminates with $\mathbf{z} = \zeta$, then we say that $P(\xi)$ *is defined and* $P(\xi) = \zeta$. Otherwise, i.e., if the execution never terminates, we say that $P(\xi)$ *is undefined.* In other words, the program P should be considered as representing a partial function $\mathbf{z} = P(\mathbf{x})$ mapping $D_{\mathbf{x}}$ into $D_{\mathbf{z}}$.

Let P be a program, $\varphi(\mathbf{x})$ be a total predicate over $D_{\mathbf{x}}$ (called the *input predicate*) and $\psi(\mathbf{x}, \mathbf{z})$ be a total predicate over $D_{\mathbf{x}} \times D_{\mathbf{z}}$ (called the *output predicate*). We say that:

1. *P is correct with respect to* φ *and* ψ, if for every ξ, such that $\varphi(\xi) = \mathrm{T}$, $P(\xi)$ is defined and $\psi(\xi, P(\xi)) = \mathrm{T}$.

2. *P is partially correct with respect to* φ *and* ψ, if for every ξ, such that $\varphi(\xi) = \mathrm{T}$, if $P(\xi)$ is defined then $\psi(\xi, P(\xi)) = \mathrm{T}$.

EXAMPLE. Let us consider the program P^* (for multiplying an integer x_1 by a nonnegative integer x_2 by repeated additions):

$$\begin{aligned} \mathbf{x} &= (x_1, x_2) \text{ is the input vector,} \\ \mathbf{y} &= (y_1, y_2) \text{ is the program vector,} \\ \mathbf{z} &= z \text{ is the output vector,} \\ D_{\mathbf{x}} &= I \times I \text{ is the input domain (where } I \text{ is the set of integers),} \\ D_{\mathbf{y}} &= I \times I \text{ is the program domain,} \\ D_{\mathbf{z}} &= I \text{ is the output domain,} \\ g(\mathbf{x}) &= (0, x_2) \text{ is the initial assignment function,} \end{aligned}$$

and the only statement is

$$\begin{aligned} 1 : \textit{if } y_2 = 0 \textit{ then } &[z \leftarrow y_1; \textit{halt}] \\ \textit{else } &[(y_1, y_2) \leftarrow (y_1 + x_1, y_2 - 1); \textit{go to } 1]. \end{aligned}$$

In the sequel we shall discuss the correctness problem of the program P^* with respect to the input predicate $x_2 \geqslant 0$ and the output predicate $z = x_1 x_2$.

THE ALGORITHM

Given a program P, an input predicate $\varphi(\mathbf{x})$ and an output predicate $\psi(\mathbf{x}, \mathbf{z})$, one can construct the formulas $W_P[\varphi, \psi]$ and $\tilde{W}_P[\varphi, \psi]$ as follows:

1. For each statement of the form

$$\begin{aligned} i : \textit{if } p_i(\mathbf{x}, \mathbf{y}) \textit{ then } &[\mathbf{y} \leftarrow f_i^1(\mathbf{x}, \mathbf{y}); \textit{go to } i_1] \\ \textit{else } &[\mathbf{y} \leftarrow f_i^2(\mathbf{x}, \mathbf{y}); \textit{go to } i_2] \end{aligned}$$

define W_i as

$$\forall \mathbf{y}\{q_i(\mathbf{x},\mathbf{y}) \supset \textit{if } p_i(\mathbf{x},\mathbf{y}) \textit{ then } q_{i_1}(\mathbf{x}, f_i^1(\mathbf{x},\mathbf{y})) \\ \textit{else } q_{i_2}(\mathbf{x}, f_i^2(\mathbf{x},\mathbf{y}))\},^3$$

where

(a) the q_i's are distinct predicate symbols, i.e., symbols representing predicates that have not yet been specified;

(b) if a *go to* i_k instruction was replaced by a *halt* instruction in P, then replace $q_{i_k}(\mathbf{x}, f_i^k(\mathbf{x},\mathbf{y}))$ by $\psi(\mathbf{x}, h_i^k(\mathbf{x},\mathbf{y}))$ in W_i.

2. Define $[P,\psi](\mathbf{x})$ as:

$$q_1(\mathbf{x}, g(\mathbf{x})) \wedge W_1 \wedge W_2 \wedge \cdots \wedge W_N .$$

3. Finally define

$$W_P[\varphi,\psi] \quad \text{as} \quad \forall \mathbf{x}\{\varphi(\mathbf{x}) \supset [P,\psi](\mathbf{x})\},$$

and

$$\tilde{W}_P[\varphi,\psi] \quad \text{as} \quad \exists \mathbf{x}\{\varphi(\mathbf{x}) \wedge [P,\sim\psi](\mathbf{x})\}.$$

A formula W, of the form of W_P or $\tilde{W}_P$, is said to be *satisfiable* if to each predicate symbol q_i which occurs in it, we can assign a total predicate over $D_\mathbf{x} \times D_\mathbf{y}$, under which W is true. W is said to be *unsatisfiable* if it is not satisfiable.

EXAMPLE. Let us consider the program P^* with the input predicate $x_2 \geqslant 0$ and the output predicate $z = x_1x_2$. Following the algorithm[4] we obtain that $W_{P^*}[x_2 \geqslant 0, z = x_1x_2]$ is

$$\forall x_1 \forall x_2 \{x_2 \geqslant 0 \supset [q(0,x_2) \wedge \forall y_1 \forall y_2 [q(y_1,y_2) \supset \textit{if } y_2 = 0 \textit{ then } y_1 = x_1x_2 \\ \textit{else } q(y_1 + x_1, y_2 - 1)]]\},$$

and $\tilde{W}_{P^*}[x_2 \geqslant 0, z = x_1x_2]$ is

$$\exists x_1 \exists x_2 \{x_2 \geqslant 0 \wedge q(0,x_2) \wedge \forall y_1 \forall y_2 [q(y_1,y_2) \supset \textit{if } y_2 = 0 \textit{ then } y_1 \neq x_1x_2 \\ \textit{else } q(y_1 + x_1, y_2 - 1)]\}.$$

One can easily verify that by assigning the predicate $y_1 = x_1(x_2 - y_2)$ to $q(y_1, y_2)$ in W_{P^*}, the expression obtained is true. Thus, W_{P^*} is satisfiable. On the other hand, we shall show later that one can derive a contradiction from $\tilde{W}_{P^*}$, which implies that $\tilde{W}_{P^*}$ is unsatisfiable.

[3] which is logically equivalent to:

$$\forall \mathbf{y}\{[[q_i(\mathbf{x},\mathbf{y}) \wedge p_i(\mathbf{x},\mathbf{y})] \supset q_{i_1}(\mathbf{x}, f_i^1(\mathbf{x},\mathbf{y}))] \\ \wedge [[q_i(\mathbf{x},\mathbf{y}) \wedge \sim p_i(\mathbf{x},\mathbf{y})] \supset q_{i_2}(\mathbf{x}, f_i^2(\mathbf{x},\mathbf{y}))]\}.$$

[4] We write $q_1(x_1, x_2, y_1, y_2)$ as $q(y_1, y_2)$ for short.

The Lemma

The results presented in this paper are proved by the use of the following definitions and lemma.

Let $\xi \in D_{\mathbf{x}}$. A total predicate $\delta(\mathbf{y})$ over $D_{\mathbf{y}}$ is called:

1. *a valid predicate of the i-th statement for $P(\xi)$*, if for every $\eta \in D_{\mathbf{y}}$, such that during the execution of $P(\xi)$ we reach the i-th statement with $\mathbf{y} = \eta$, $\delta(\eta) = T$.

2. *The minimal valid predicate of the i-th statement for $P(\xi)$*, if for every $\eta \in D_{\mathbf{y}}$, such that during the execution of $P(\xi)$ we reach the i-th statement with $\mathbf{y} = \eta$, *and for no other* η, $\delta(\eta) = T$.

Example. Let us consider the program P^* with the input value $\xi = (3, 4)$. The predicate $(0 \leqslant y_1 \leqslant 12) \wedge (0 \leqslant y_2 \leqslant 4)$ is a valid predicate [while the predicate $\delta(y_1, y_2)$ that is true only for $(0, 4)$, $(3, 3)$, $(6, 2)$, $(9, 1)$, and $(12, 0)$ is the minimal valid predicate] of the first statement for $P^*(3, 4)$.

Lemma 1

(i) *$P(\xi)$ is undefined, or*

(ii) *$P(\xi)$ is defined and $\psi(\xi, P(\xi)) = \mathrm{T}$,*

if and only if $[P, \psi](\xi)$ is satisfiable.

Proof.

$\Rightarrow$: For each predicate symbol q_i, $1 \leqslant i \leqslant N$, in $[P, \psi](\xi)$, assign to $q_i(\xi, \mathbf{y})$ the minimal valid predicate of the i-th statement for $P(\xi)$. Since $P(\xi)$ is either undefined or defined and $\psi(\xi, P(\xi)) = \mathrm{T}$ (in other words, if $P(\xi)$ is defined then $\psi(\xi, P(\xi)) = \mathrm{T}$), it follows by the construction of $[P, \psi]$ that the value of $[P, \psi](\xi)$, with the above assignments for its q_i's, is true; i.e. $[P, \psi](\xi)$ is satisfiable.

$\Leftarrow$: If $[P, \psi](\xi)$ is satisfiable, it means that there exist assignments of specified total predicates $\delta_i(\mathbf{y})$ over $D_{\mathbf{y}}$ for the predicate symbols $q_i(\xi, \mathbf{y})$, $1 \leqslant i \leqslant N$, under which the value of $[P, \psi](\xi)$ is true. By the construction of $[P, \psi]$ this implies that each δ_i, $1 \leqslant i \leqslant N$, is a valid predicate of the i-th statement for $P(\xi)$, and therefore that if $P(\xi)$ is defined then $\psi(\xi, P(\xi)) = \mathrm{T}$.

Q.E.D.

By Lemma 1 it follows that

1. $[P, \mathrm{T}](\xi)$ is always satisfiable, and
2. $[P, \mathrm{F}](\xi)$ is satisfiable if and only if $P(\xi)$ is undefined.

The Correctness of Programs

The following results formalize the correctness of programs:

Theorem 1. *P is partially correct with respect to φ and ψ, if and only if $W_P[\varphi, \psi]$ is satisfiable.*

Theorem 2. *P is correct with respect to φ and ψ, if and only if $\tilde{W}_P[\varphi, \psi]$ is unsatisfiable.*

Theorem 1 represents Floyd's result [2] and it's proof is immediate by Lemma 1. Theorem 2 can also be proved by Lemma 1 by showing that:

P is *not* correct with respect to φ and ψ [i.e., $\exists\xi$, such that $\varphi(\xi) = \mathrm{T}$, and

(i) $P(\xi)$ is undefined, or

(ii) $P(\xi)$ is defined and $\psi(\xi, P(\xi)) = \mathrm{F}$],

if and only if $\tilde{W}_P[\varphi, \psi]$ is satisfiable.

By Theorem 1 (with $\psi \equiv \mathrm{F}$) and Theorem 2 (with $\psi \equiv \mathrm{T}$), respectively, it follows that

Corollary 1. *For every ξ, such that $\varphi(\xi) = \mathrm{T}$, $P(\xi)$ is undefined, if and only if $W_P[\varphi, F]$ is satisfiable.*

Corollary 2. *For every ξ, such that $\varphi(\xi) = \mathrm{T}$, $P(\xi)$ is defined, if and only if $\tilde{W}_P[\varphi, \mathrm{T}]$ is unsatisfiable.*

Example. Theorem 2 implies that we can prove the correctness of the program P^* with respect to the input predicate $x_2 \geqslant 0$ and the output predicate $z = x_1x_2$, by showing that $\tilde{W}_{P^*}[x_2 \geqslant 0, z = x_1x_2]$ is unsatisfiable. $\tilde{W}_{P^*}[x_2 \geqslant 0, z = x_1x_2]$ was already found to be

$$\exists x_1 \exists x_2 \{x_2 \geqslant 0 \wedge q(0, x_2) \wedge \forall y_1 \forall y_2 [q(y_1, y_2) \supset \textit{if } y_2 = 0 \textit{ then } y_1 \neq x_1x_2 \\ \textit{else } q(y_1 + x_1, y_2 - 1)]\}.$$

We shall show that $\tilde{W}_{P^*}[x_2 \geqslant 0, z = x_1x_2]$ is unsatisfiable by deriving a contradiction using the following four clauses:

(1) $x_2 \geqslant 0$,

(2) $q(0, x_2)$,

(3) $\forall y_1 \forall y_2 \{[q(y_1, y_2) \wedge y_2 > 0] \supset q(y_1 + x_1, y_2 - 1)\}$, and

(4) $\forall y_1 \forall y_2 \{[q(y_1, y_2) \wedge y_2 = 0] \supset y_1 \neq x_2x_2\}$.

By substituting $x_1(x_2 - y_2)$ for y_1 in (3), we obtain:

(3') $\forall y_2 \{[q(x_1(x_2 - y_2), y_2) \wedge y_2 > 0] \supset q(x_1(x_2 - y_2) + x_1, y_2 - 1)\}$.

Using the Induction Principle

$$\exists x\{x \geqslant 0 \wedge Q(x)\} \wedge \forall y\{[Q(y) \wedge y > 0] \supset Q(y - 1)\} \supset Q(0)$$

with $Q(t) \equiv q(x_1(x_2 - t), t)$, we obtain by (1), (2), and (3′):

$$Q(0), \quad \text{i.e.} \quad q(x_1x_2, 0), \text{ which contradicts (4).}$$

The Equivalence of Programs

Two programs P_1 and P_2 are said to be *comparable*, if they have the same input variable $\mathbf{x}$, the same output variable $\mathbf{z}$, the same input domain $D_\mathbf{x}$, and the same output domain $D_\mathbf{z}$.

Let P_1 and P_2 be any two comparable programs, and let $\varphi(\mathbf{x})$ be a total predicate over $D_\mathbf{x}$ (called the *input predicate*). We say that P_1 *and* P_2 *are equivalent with respect to* φ, if for every ξ, such that $\varphi(\xi) = \text{T}$, both $P_1(\xi)$ and $P_2(\xi)$ are defined and $P_1(\xi) = P_2(\xi)$.

The equivalence of programs can be formalized using the formula $W_{P_1,P_2}[\varphi]$, which is defined as:

$$\exists \mathbf{x}\{\varphi(\mathbf{x}) \wedge [P_1, r](\mathbf{x}) \wedge [P_2, \sim r](\mathbf{x})\},$$

where

(a) $r(\mathbf{x}, \mathbf{z})$ is any predicate symbol, and

(b) the symbols r, q_i' (used in constructing $[P, r]$), and q_i'' (used in constructing $[P_2, \sim r]$) must be distinct.

Theorem 3. *P_1 and P_2 are equivalent with respect to φ, if and only if $W_{P_1,P_2}[\varphi]$ is unsatisfiable.*

Proof. We shall prove that: P_1 and P_2 are *not* equivalent with respect to φ, i.e., $\exists \xi$, such that $\varphi(\xi) = T$, and

(i) $P_1(\xi)$ is undefined, or

(ii) $P_2(\xi)$ is undefined, or

(iii) $P_1(\xi)$ and $P_2(\xi)$ are defined but $P_1(\xi) \neq P_2(\xi)$,

if and only if $W_{P_1,P_2}[\varphi]$ is satisfiable.

(i) $\Rightarrow$ Assign F to r and use Lemma 1.

(ii) $\Rightarrow$ Assign T to r and use Lemma 1.

(iii) $\Rightarrow$ Assign the predicate δ (where $\delta(\mathbf{x}, \mathbf{z}) = \text{T}$ if and only if $(\mathbf{x}, \mathbf{z}) = (\xi, P_1(\xi))$) to r. By Lemma 1, $[P_1, \delta](\xi)$ is satisfiable. Moreover, since $P_1(\xi) \neq P_2(\xi)$, it follows that $\delta(\xi, P_2(\xi)) = \text{F}$, i.e., $\sim\delta(\xi, P_2(\xi)) = \text{T}$. Therefore, by Lemma 1, $[P_2, \sim\delta](\xi)$ is also satisfiable. This implies that $W_{P_1,P_2}[\varphi]$ is satisfiable.

$\Leftarrow$ Suppose that $W_{P_1,P_2}[\varphi]$ is satisfiable, with ξ assigned to $\mathbf{x}$ and δ assigned to r. Since $[P_1, \delta](\xi)$ is satisfiable, it follows by Lemma 1, that if $P_1(\xi)$ is defined then $\delta(\xi, P_1(\xi)) = \mathrm{T}$. Since $[P_2, \sim\delta](\xi)$ is satisfiable, it follows, also by Lemma 1, that if $P_2(\xi)$ is defined then $\sim\delta(\xi, P_2(\xi)) = \mathrm{T}$. This implies that if both $P_1(\xi)$ and $P_2(\xi)$ are defined then $\delta(\xi, P_1(\xi)) = \mathrm{T}$ and $\delta(\xi, P_2(\xi)) = \mathrm{F}$, i.e., $P_1(\xi) \neq P_2(\xi)$.

Q.E.D.

Example. Let us consider the two programs P_1^* and P_2^* (for computing $x!$ for a nonnegative integer x), where

1. in both programs

$$\mathbf{x} = x, \quad \mathbf{y} = (y_1, y_2), \quad \mathbf{z} = z, \quad D_{\mathbf{x}} = I, \quad D_{\mathbf{y}} = I \times I, \quad \text{and} \quad D_{\mathbf{z}} = I;$$

2. in addition:

   (a) P_1^* consists of the initial assignment function $g_1(x) = (1, x)$, and the statement

$$\begin{aligned} 1: & \textit{ if } y_2 = 0 \textit{ then } [z \leftarrow y_1;\ \textit{halt}] \\ & \qquad \textit{else } [(y_1, y_2) \leftarrow (y_1 y_2, y_2 - 1);\ \textit{go to } 1]; \end{aligned}$$

   (b) P_2^* consists of the initial assignment function $g_2(x) = (1, 0)$, and the statement

$$\begin{aligned} 1: & \textit{ if } y_2 = x \textit{ then } [z \leftarrow y_1;\ \textit{halt}] \\ & \qquad \textit{else } [(y_1, y_2) \leftarrow (y_1(y_2 + 1), y_2 + 1);\ \textit{go to } 1]. \end{aligned}$$

P_1^* and P_2^* are clearly comparable.

Theorem 3 implies that we can prove the equivalence of the programs P_1^* and P_2^* with respect to the input predicate $x \geqslant 0$, by showing that $W_{P_1^*,P_2^*}[x \geqslant 0]$ is unsatisfiable, where $W_{P_1^*,P_2^*}[x \geqslant 0]$ is:

$$\begin{aligned} &\exists x\{x \geqslant 0 \\ &\quad \wedge q_1'(x, 1, x) \wedge \forall y_1 \forall y_2[q_1'(x, y_1, y_2) \supset \textit{if } y_2 = 0 \textit{ then } r(x, y_1) \\ &\qquad\qquad\qquad\qquad\qquad\qquad \textit{else } q_1'(x, y_1 y_2, y_2 - 1)] \\ &\quad \wedge q_1''(x, 1, 0) \wedge \forall y_1 \forall y_2[q_1''(x, y_1, y_2) \supset \textit{if } y_2 = x \textit{ then } \sim r(x, y_1) \\ &\qquad\qquad\qquad\qquad\qquad\qquad \textit{else } q_1''(x, y_1(y_2 + 1), y_2 + 1)]\}. \end{aligned}$$

Acknowledgments

I am indebted to Professor Robert W. Floyd, whose early results served as motivation for this work, for his encouragement while I carried out this research.

Thanks are also due to Professor John McCarthy for his interest. It was his suggestion to extend my early results for termination and equivalence to include correctness as well.

I am also grateful to Thomas Bredt and Lockwood Morris for their detailed reading of the manuscript.

REFERENCES

1. D. C. COOPER. "Program Scheme Equivalences and Second Order Logic." Presented at Fourth Annual Machine Intelligence Workshop, University of Edinburgh (August 1968).
2. R. W. FLOYD. "Assigning Meaning to Programs." Proceedings of Symposia in Applied Mathematics, American Mathematical Society, Vol. 19, 19–32 (1967).
3. R. W. FLOYD. "The Verifying Compiler." Computer Science Research Review, Carnegie-Mellon University (December 1967).
4. Z. MANNA. "Termination of Algorithms." Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University (April 1968).
5. Z. MANNA. "Properties of Programs and the First-Order Predicate Calculus." To be published in the *JACM* (April 1969).
6. E. MENDELSON. "Introduction to Mathematical Logic." Van Nostrand Company, Princeton (1964).