Properties of Programs and the First-Order Predicate Calculus

ZOHAR MANNA*

Carnegie-Mellon University,† Pittsburgh, Pennsylvania

ABSTRACT. This paper is concerned with the relationship of the termination problem for programs and abstract programs to the validity of certain formulas in the first-order predicate calculus. By exploiting this relationship, subclasses of abstract programs for which the termination problem is decidable can be isolated. Moreover, known proof procedures for the first-order predicate calculus (e.g. resolution) can be applied to prove the termination of both programs and abstract programs. The correctness and equivalence problems of abstract programs are shown to be reducible to the termination problem.

KEY WORDS AND PHRASES: termination, correctness, equivalence, programs, abstract programs, predicate calculus, unsatisfiability, validity

CR CATEGORIES: 5.20

Introduction

An *abstract program* (program schema) is a program, but with function, predicate, and constant symbols instead of specified functions, predicates, and constants. Thus an abstract program AP may be thought of as representing a family of (real) programs. By specifying an interpretation $\mathcal{I}$ for the symbols of AP, a program $(AP, \mathcal{I})$ of this family is obtained. The program contains a set of input variables. Each assignment of values to the input variables defines a (unique) execution of the program. Recent papers on abstract programs include those of Ianov (see Rutledge [16]); Luckham, Park, and Paterson [10]; Paterson [14]; Engeler [6]; and Kaplan [9].

In this paper we are concerned with the termination problem of programs and abstract programs. A program $(AP, \mathcal{I})$ is said to terminate if all possible executions of the program terminate. An abstract program AP is said to terminate if for every interpretation $\mathcal{I}$, the program $(AP, \mathcal{I})$ terminates.

Given an abstract program AP, an algorithm is described to construct a well-formed formula W_{AP} of the first-order predicate calculus such that AP terminates if and only if W_{AP} is unsatisfiable, i.e. $\sim W_{AP}$ is valid. This implies that conclusions about the termination of abstract programs can be obtained by applying well-known results in logic. A corresponding result for programs is presented.

The relation between termination of computations and the validity of well-formed formulas of the predicate calculus has also been considered in the classical paper by Turing [17] which shows that the decision problem for the predicate calculus is

* Present address: Computer Science Department, Stanford University, Stanford, California.
† Computer Science Department. This work is based on the author's Ph.D. Thesis [11]. The work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-146).

unsolvable (see also Büchi [2]). The technique we are using is a natural extension of Floyd's method [7] for proving the correctness of programs.

An abstract program AP is said to be correct if for every interpretation $\mathcal{I}$ the program $(AP, \mathcal{I})$ terminates and yields the desired final result. Two abstract programs AP and AP' are said to be equivalent if for every interpretation $\mathcal{I}$ both programs $(AP, \mathcal{I})$ and $(AP', \mathcal{I})$ terminate, and for the same input values they yield the same final values.

Both the correctness and the equivalence problems of abstract programs are shown to be reducible to the termination problem.

Certain extensions of these results, with examples and applications, can be found in Manna [11, 12], Manna and Pnueli [13], and Cooper [4].

1. *Mathematical Background*

1.1. The (First-Order) Predicate Calculus. In this section we partially follow the exposition of Davis and Putnam [5].

The symbols from which our formulas are constructed are:

(a) Improper symbols:

punctuation marks	, ( )
logical symbols	$\sim \supset \wedge \vee \equiv \exists$
primitive constants	T F

(b) Constants:

n-adic function constants	f_i^n $(i \geq 1,\ n \geq 0)$
[f_i^0 are called *individual constants*],	
n-adic predicate constants	p_i^n $(i \geq 1,\ n \geq 0)$
[p_i^0 are called *propositional constants*]	

(c) Variables:

Individual variables	x_i $(i \geq 1)$
n-adic predicate variables	q_i^n $(i \geq 1,\ n \geq 0)$
[q_i^0 are called *propositional variables*][1]	

We define recursively three classes of expressions as follows:

(a) *Terms*

1. Each individual variable x_i and each individual constant f_i^0 is a term.
2. If $t_1, t_2, \cdots, t_n$ $(n \geq 1)$ are terms, then so is $f_i^n(t_1, t_2, \cdots, t_n)$.
3. The terms consist exactly of these expressions generated by 1 and 2.

(b) *Atomic formulas*

1. T, F, p_i^0, and q_i^0 are atomic formulas.
2. If $t_1, t_2, \cdots, t_n$ $(n \geq 1)$ are terms, then the expressions $p_i^n(t_1, t_2, \cdots, t_n)$ and $q_i^n(t_1, t_2, \cdots, t_n)$ are atomic formulas.
3. The atomic formulas consist exactly of those expressions generated by 1 and 2.

(c) *Well-formed formulas (wffs)*

1. An atomic formula is a well-formed formula (wff).
2. If R is a wff, then so are $\sim R$, $(x_i)R$ (x_i is said to be *universally* quantified), and $(\exists x_i)R$ (x_i is said to be *existentially* quantified).

[1] In the following we also use y_i as individual variables and a_i as individual constants.

3. If R and S are wffs, then so are $(R \supset S)$, $(R \wedge S)$, $(R \vee S)$, and $(R \equiv S)$.
4. The wffs consist exactly of those expressions generated by 1, 2, and 3.

Parentheses, subscripts, and superscripts are omitted whenever their omission causes no confusion.

An occurrence of x_i in a wff R is a *bound occurrence* if it is in a part of R which is a wff of the form $(x_i)S$ or $(\exists x_i)S$. An occurrence of x_i which is not bound is called a *free occurrence*. x_i is *free* in R if it has at least one free occurrence in R.

Those wffs which are *logically valid* can be singled out either by specifying axioms and rules of inference or by referring to "interpretations" of the wffs of the system; by Gödel's Completeness Theorem, both of these procedures lead to the same class of formulas. It is most convenient here to use the latter formulation employing "interpretation."

An *interpretation* $\mathcal{I}$ for a wff W consists of a nonempty set of elements $D_{\mathcal{I}}$ (called the *domain of the interpretation*) and the following assignments to the *constants* of W:

1. To each function constant f_i^n which occurs in W, we assign a total function mapping $(D_{\mathcal{I}})^n$ into $D_{\mathcal{I}}$. (If $n = 0$, the individual constant f_i^0 is assigned some fixed element of $D_{\mathcal{I}}$.)
2. To each predicate constant p_i^n which occurs in W, we assign a total function mapping $(D_{\mathcal{I}})^n$ into $\{\mathrm{T}, \mathrm{F}\}$. (If $n = 0$, the propositional constant p_i^0 is assigned the value T or F.)

Given a wff W and an interpretation $\mathcal{I}$ for W (notation: $(W, \mathcal{I})$), an *assignment* Γ for $(W, \mathcal{I})$ consists of the following assignments to the *variables* of W:

1. To each free individual variable x_i in W, we assign some fixed element of $D_{\mathcal{I}}$.
2. To each predicate variable q_i^n which occurs in W, we assign a total function mapping $(D_{\mathcal{I}})^n$ into $\{\mathrm{T}, \mathrm{F}\}$. (If $n = 0$, the propositional variable q_i^0 is assigned the value T or F.)

Let W be a wff. Then given an interpretation $\mathcal{I}$ for W and an assignment Γ for $(W, \mathcal{I})$ (notation: $(W, \mathcal{I}, \Gamma)$), the value T or F will be assigned to $(W, \mathcal{I}, \Gamma)$. This value is obtained simply by using the assignments of $\mathcal{I}$ and Γ, interpreting F as falsehood and T as truth, using the usual truth tables for $\sim$, $\wedge$, $\vee$, $\supset$, and $\equiv$, and interpreting the universally and existentially quantified variables in the standard way.

$(W, \mathcal{I})$ is said to be:

(1) *valid* if for every assignment Γ, $(W, \mathcal{I}, \Gamma)$ has the value T;

(2) *satisfiable* if $(W, \mathcal{I}, \Gamma)$ has the value T for some assignment Γ; or

(3) *unsatisfiable* if it is not satisfiable.

Clearly, *$(W, \mathcal{I})$ is valid if and only if $(\sim W, \mathcal{I})$ is unsatisfiable.*

A wff W is said to be:

(1) *valid* if for every interpretation $\mathcal{I}$, $(W, \mathcal{I})$ is valid;

(2) *satisfiable* if $(W, \mathcal{I})$ is satisfiable for some interpretation $\mathcal{I}$; or

(3) *unsatisfiable* if it is not satisfiable.

Clearly, *W is valid if and only if $\sim W$ is unsatisfiable.*

A wff is called *quantifier-free* if it contains no occurrence of (x_i) or $(\exists x_i)$.

A wff is in *prenex normal form* if it begins with a sequence of quantifiers (x_i) and $(\exists x_i)$ in which no variable occurs more than once (called the *prefix*), and if the sequence is followed by a quantifier-free wff (called the *matrix*).

Let W be a wff in prenex normal form. Then *the functional form of* W is defined as follows.

Let the variables in the prefix of W (in order of occurrence) be $x_1, x_2, \cdots, x_N$. Let the existentially quantified variables in the prefix be $x_{i_1}, x_{i_2}, \cdots, x_{i_M}$. Then for every j, $1 \leq j \leq M$, (1) the quantifier $(\exists x_{i_j})$ is to be deleted from the prefix; and (2) each occurrence of x_{i_j} in the matrix of W is to be replaced by an occurrence of the term $f_{i_j}^q(x_{k_1}, x_{k_2}, \cdots, x_{k_q})$, where $(x_{k_1}), (x_{k_2}), \cdots, (x_{k_q})$, $q \geq 0$, are all the universal quantifiers which precede $(\exists x_{i_j})$ in the prefix of W, and $f_{i_j}^q$ is the first q-adic function constant which does not occur in W and has not been used previously in this process.

We use the following known result: *W is satisfiable if and only if its functional form is satisfiable.*

1.2. The Validity Problem of the Predicate Calculus. *The validity problem of the predicate calculus is undecidable.* That is, there can be no algorithm which takes as input any wff and in all cases terminates with a decision as to whether the wff is valid or not. But *the validity problem of the predicate calculus is semidecidable.* That is, there are algorithms, called *semidecision procedures,* which take as input any wff, and (1) if the wff is valid the algorithm will stop and say so; or (2) if the wff is not valid the algorithm will never stop. The algorithms have undergone successive reductions so that by now they have a simple structure. Many recent algorithms are based on the *resolution principle* (Robinson [15]). Furthermore, there exist classes of wffs for which the problem is decidable. For example, the validity problem is decidable for the following three classes:[2]

1. $W_1 = \{W \mid W$ is a wff in prenex normal form without function constants and with prefix of the form $\forall \cdots \forall \exists \cdots \exists\}$,
2. $W_2 = \{W \mid W$ is a wff in prenex normal form without function constants and with prefix of the form $\forall \cdots \forall \exists \forall \cdots \forall\}$,
3. $W_3 = \{W \mid W$ is a wff in prenex normal form without function constants and with prefix of the form $\forall \cdots \forall \exists \exists \forall \cdots \forall\}$.

2. *Definitions*

2.1. Abstract Programs. An *abstract program* (or program schema) AP consists of:

1. A finite directed graph[3] $\langle V, L, A \rangle$ such that
   (a) there exists exactly one vertex $S \in V$ with in-degree 0 (i.e. with no arcs leading *to* S), called the *start vertex;*
   (b) there exists exactly one vertex $H \in V$ with out-degree 0 (i.e. with no arcs leading *from* H), called the *halt vertex;* and
   (c) every vertex $v \in V$ is on some path that joins S and H.
2. (a) A set of m ($m \geq 0$) distinct individual variables $\bar{y} = (y_1, y_2, \cdots, y_m)$, called *input variables;* and
   (b) a set of n ($n \geq 1$) distinct individual variables $\bar{x} = (x_1, x_2, \cdots, x_n)$, called *program variables.*
3. With each arc $\alpha = (v, l, v') \in A$ there is associated

[2] See Ackermann [1] or Church [3, Sec. 46].

[3] I.e. V (vertices), L (labels), and A (arcs) are nonempty finite sets. $A \subseteq V \times L \times V$.

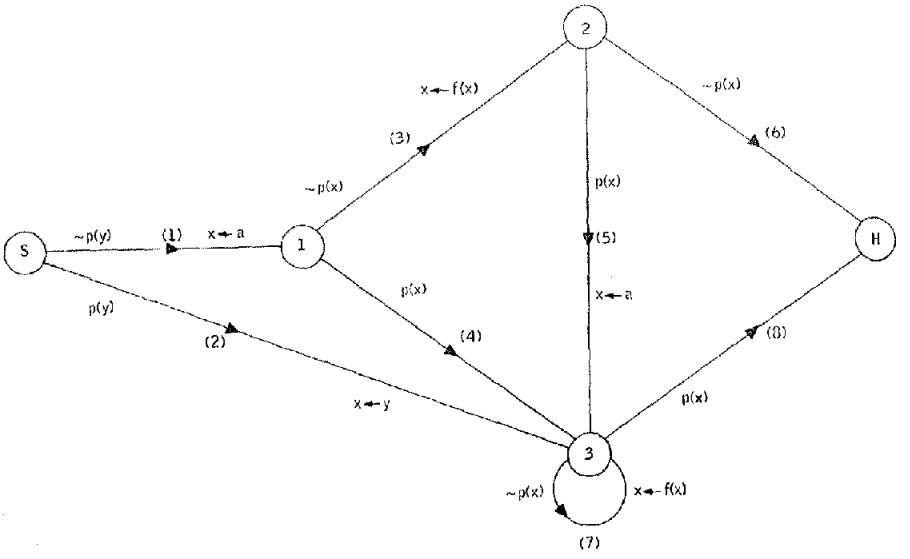

FIG. 1. The abstract program AP^*

(a) a quantifier-free wff φ_α called the *test predicate of* α; and
(b) an n-tuple $\bar{t}_\alpha = (t_1^{(\alpha)}, t_2^{(\alpha)} \cdots, t_n^{(\alpha)})$ of terms called the *assignment function*[4] *of* α.

The wff φ_α does not contain any predicate variables. In addition, the wff φ_α and the terms $t_i^{(\alpha)}$ do not contain individual variables other than $\bar{y}$ and $\bar{x}$. If $v = S$ (i.e. α is an arc leading from the start vertex), the wff φ_α and the terms t_i^α do not contain the program variables $\bar{x}$.[5]

In addition, an abstract program should satisfy the following restriction:

4. For every vertex v ($v \neq H$), if $\alpha_1, \alpha_2, \cdots, \alpha_N$ is the set of all arcs leading from v, the set of the test predicates $\varphi_{\alpha_1}, \varphi_{\alpha_2}, \cdots, \varphi_{\alpha_N}$ is
   (a) *complete*, i.e. $(\bar{x})(\bar{y})\ [\varphi_{\alpha_1} \vee \varphi_{\alpha_2} \vee \cdots \vee \varphi_{\alpha_N}]$ is valid, and
   (b) *mutually exclusive*,[6] i.e. $(\exists \bar{x})(\exists \bar{y})\ [\varphi_{\alpha_i} \wedge \varphi_{\alpha_j}]$ is unsatisfiable for every pair (i, j), $1 \leq i \neq j \leq N$.

Example. Figure 1 represents an abstract program. We refer later to this abstract program as AP^*. Here, a is an individual constant, f is a monadic function constant, p is a monadic predicate constant, y is an input variable, and x is a program variable.

2.2. PROGRAMS. An *interpretation* $\mathcal{I}$ of an abstract program AP consists of a nonempty set of elements $D_\mathcal{I}$ (called the *domain of the interpretation*) and assignments to the constants of AP:

1. To each function constant f_i^n which occurs in AP we assign a total function mapping $(D_\mathcal{I})^n$ into $D_\mathcal{I}$.

[4] The intended interpretation is: v: if φ_α then [replace simultaneously each variable x_i by $t_i^{(\alpha)}$ and go to v'].

[5] We have restricted φ_α to be a quantifier-free wff. However, the theorems presented in this work still hold in the case when φ_α is any wff that does not contain free individual variables other than $\bar{y}$ and $\bar{x}$.

[6] I.e. under each interpretation and each assignment exactly one of the test predicates is true.

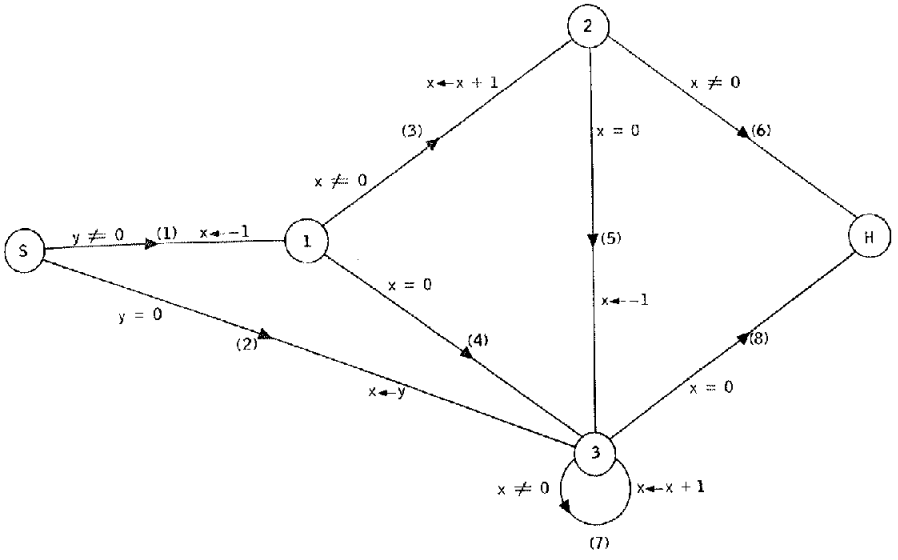

FIG. 2. The program $(AP^*, \mathfrak{I}^*)$

2. To each predicate constant p_i^n which occurs in AP we assign a total function mapping $(D_\mathfrak{I})^n$ into $\{T, F\}$.

Let AP be an abstract program and $\mathfrak{I}$ an interpretation of AP. The pair $(AP, \mathfrak{I})$ is called a *program*.

Example. Consider the abstract program AP^* of Section 2.1. Let $\mathfrak{I}^*$ be the following interpretation of AP^*: D is I (the integers), $f(x)$ is $x + 1$, $p(x)$ is $x = 0$, and a is -1. Then the program $(AP^*, \mathfrak{I}^*)$ can be represented by Figure 2.

In order to give a rough idea of what follows in Section 2.3, let us only mention that the "ALGOL" meaning of Figure 2 is:

START: if $y = 0$ then [$x \leftarrow y$; go to 3] else [$x \leftarrow -1$; go to 1];
1: if $x = 0$ then [$x \leftarrow x$; go to 3] else [$x \leftarrow x + 1$; go to 2];
2: if $x = 0$ then [$x \leftarrow -1$; go to 3] else [$x \leftarrow x$; HALT];
3: if $x = 0$ then [$x \leftarrow x$; HALT] else [$x \leftarrow x + 1$; go to 3].

2.3. INTERPRETED PROGRAMS. Let (AP, $\mathfrak{I}$) be a program. Then the result obtained by assigning values $\bar{\gamma}$, $\bar{\gamma} \in (D_\mathfrak{I})^m$, for the input variables $\bar{y}$ of the program is called the *interpreted program*[7] $(AP, \mathfrak{I}, \bar{\gamma})$.

Example. By assigning the value 1 to the input variable y of the program $(AP^*, \mathfrak{I}^*)$ of Section 2.2, we obtain the interpreted program $(AP^*, \mathfrak{I}^*, 1)$ represented by Figure 3.

The interpreted program $(AP, \mathfrak{I}, \bar{\gamma})$ defines an *execution sequence* $\langle AP, \mathfrak{I}, \bar{\gamma} \rangle$ which is a (finite or infinite) sequence of triples

$$(l^{(1)}, v^{(1)}, \bar{x}^{(1)}), \quad (l^{(2)}, v^{(2)}, \bar{x}^{(2)}), \quad (l^{(3)}, v^{(3)}, \bar{x}^{(3)}), \cdots$$

where:

1. $(l^{(j)}, v^{(j)}, \bar{x}^{(j)}) \in L \times V \times (D_\mathfrak{I})^n$ for every j, $j \geq 1$.

[7] Programs with no input variables (i.e. $m = 0$) will be considered as interpreted programs.

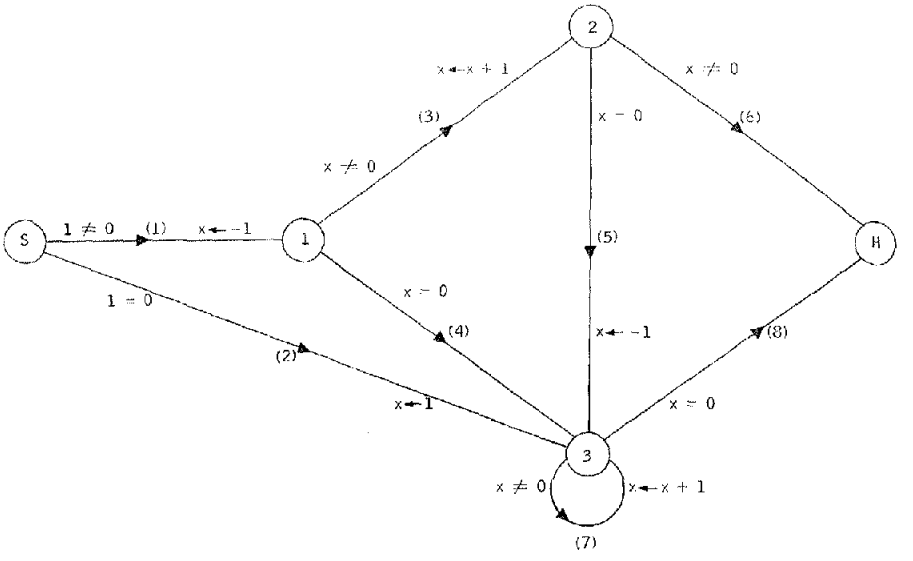

FIG. 3. The interpreted program $(AP^*, \mathcal{I}^*, 1)$

2. $(l^{(1)}, v^{(1)}, \bar{x}^{(1)})$ is the first triple in the sequence if and only if there exists an arc $\alpha = (S, l^{(1)}, v^{(1)}) \in A$ such that[8] $\varphi_\alpha(\bar{\gamma}) = T$ and $\bar{x}^{(1)} = \bar{t}_\alpha(\bar{\gamma})$.
3. $(l^{(j)}, v^{(j)}, \bar{x}^{(j)})$ and $(l^{(j+1)}, v^{(j+1)}, \bar{x}^{(j+1)})$ are two successive triples in the sequence if and only if there exists an arc $\alpha = (v^{(j)}, l^{(j+1)}, v^{(j+1)}) \in A$ s.t.[9] $\varphi_\alpha(\bar{x}^{(j)}, \bar{\gamma}) = T$ and $\bar{x}^{(j+1)} = \bar{t}_\alpha(\bar{x}^{(j)}, \bar{\gamma})$.
4. The sequence is finite, of length $k \geq 1$, if and only if $v^{(k)} = H$. In this case $\bar{x}^{(k)}$ is called the *value of the execution sequence* $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ and is denoted by *val* $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$.

In other words, execution always starts at the start vertex. On execution of the jth step, $j \geq 1$, control moves along the arc $\alpha = (v^{(j-1)}, l^{(j)}, v^{(j)})$ where $v^{(0)} = S$, and φ_α represents the condition that this arc is entered. The value of each program variable x_i is replaced in the jth step by the current value of $t_i^{(\alpha)}$, simultaneously. So $\bar{x}^{(j)}$ represents the current value of the program variables $\bar{x}$ after executing the jth step. Execution stops whenever control reaches the halt vertex.

Example. The interpreted program $(AP^*, \mathcal{I}^*, 1)$ defines the execution sequence $\langle AP^*, \mathcal{I}^*, 1\rangle$: $(1, 1, -1)$, $(3, 2, 0)$, $(5, 3, -1)$, $(7, 3, 0)$, $(8, H, 0)$.

Let $(AP, \mathcal{I}, \bar{\gamma})$ be an interpreted program, and let $v \in V$ be any vertex of AP. Let δ be a specified total predicate from $(D_\mathcal{I})^n$ into $\{T, F\}$. Then

(1) δ is called a *valid predicate of* v *for* $(AP, \mathcal{I}, \bar{\gamma})$ if $\forall \bar{\xi}, \bar{\xi} \in (D_\mathcal{I})^n$: $(\exists l \in L)$ $[(l, v, \bar{\xi})$ occurs in $\langle AP, \mathcal{I}, \bar{\gamma}\rangle] \Rightarrow \delta(\bar{\xi}) = T$; and

(2) δ is called the *minimal valid predicate of* v *for* $(AP, \mathcal{I}, \bar{\gamma})$ if $\forall \bar{\xi}, \bar{\xi} \in (D_\mathcal{I})^n$: $(\exists l \in L)[(l, v, \bar{\xi})$ occurs in $\langle AP, \mathcal{I}, \bar{\gamma}\rangle] \Leftrightarrow \delta(\bar{\xi}) = T$.

Example. The predicate $x \leq 0$ is a valid predicate; the predicate $x = -1$ is the minimal valid predicate of the vertex 1 for the interpreted program $(AP^*, \mathcal{I}^*, 1)$.

[8] $\varphi_\alpha(\bar{\gamma})$ and $\bar{t}_\alpha(\bar{\gamma})$ stand for the result of substituting $\bar{\gamma}$ for $\bar{y}$ in φ_α and $\bar{t}_\alpha$.
[9] $\varphi_\alpha(\bar{x}^{(j)}, \bar{\gamma})$ and $\bar{t}_\alpha(\bar{x}^{(j)}, \bar{\gamma})$ stand for the result of substituting $\bar{x}^{(j)}$ for $\bar{x}$ and $\bar{\gamma}$ for $\bar{y}$ in φ_α and $\bar{t}_\alpha$.

3. *Termination of Programs and Abstract Programs*

3.1. THE ALGORITHM TO CONSTRUCT W_{AP}. In this section we describe an algorithm to construct from a given abstract program AP a wff W_{AP}. In Section 3.3 we state results about the relationship between AP and W_{AP}.

ALGORITHM 1. Let AP be any abstract program with program variables $\bar{x} = (x_1, x_2, \cdots, x_n)$, $n \geq 1$, and input variables $\bar{y} = (y_1, y_2, \cdots, y_m)$, $m \geq 0$. Associate with every vertex v_i of AP a distinct n-adic predicate variable q_i. For each arc $\alpha = (v_i, l, v_j)$, define W_α as

$$(q_i(\bar{x}) \wedge \varphi_\alpha) \supset q_j(\bar{t}_\alpha).$$

However, if $v_i = S$ (i.e. v_i is the start vertex of AP), replace the occurrence of $q_i(\bar{x})$ in W_α by T, and if $v_j = H$ (i.e. v_j is the halt vertex of AP), replace the occurrence of $q_j(\bar{t}_\alpha)$ in W_α by F.

Let $\alpha_1, \alpha_2, \cdots, \alpha_N$ be the set of all the arcs of AP. Then define W_{AP} as:[10]

$$(\bar{x})[W_{\alpha_1} \wedge W_{\alpha_2} \wedge \cdots \wedge W_{\alpha_N}].$$

Example. The wff W_{AP^*} of the abstract program AP^* of Section 2.1 is $(x)(\wedge_{i=1}^{8} W_i)$, where:

W_1: $T \wedge \sim p(y) \supset q_1(a)$
W_2: $T \wedge p(y) \supset q_3(y)$
W_3: $q_1(x) \wedge \sim p(x) \supset q_2(f(x))$
W_4: $q_1(x) \wedge p(x) \supset q_3(x)$
W_5: $q_2(x) \wedge p(x) \supset q_3(a)$
W_6: $q_2(x) \wedge \sim p(x) \supset F$
W_7: $q_3(x) \wedge \sim p(x) \supset q_3(f(x))$
W_8: $q_3(x) \wedge p(x) \supset F$

3.2. TERMINATION OF PROGRAMS

Definition 1. The program $(AP, \mathcal{I})$ is said to *terminate* if $\forall \bar{\gamma}, \bar{\gamma} \in (D_{\mathcal{I}})^m$, the execution sequence $\langle AP, \mathcal{I}, \bar{\gamma} \rangle$ is finite.

We are ready now to state the main result.

THEOREM 1. *The program $(AP, \mathcal{I})$ terminates iff $(W_{AP}, \mathcal{I})$ is unsatisfiable (or equivalently, $(\sim W_{AP}, \mathcal{I})$ is valid).*

PROOF. We prove that the program $(AP, \mathcal{I})$ does not terminate iff $(W_{AP}, \mathcal{I})$ is satisfiable.

(1) If $(AP, \mathcal{I})$ does not terminate then $(W_{AP}, \mathcal{I})$ is satisfiable.

If the program $(AP, \mathcal{I})$ does not terminate, there exists a $\bar{\gamma}$, $\bar{\gamma} \in (D_{\mathcal{I}})^m$, such that the execution sequence $\langle AP, \mathcal{I}, \bar{\gamma} \rangle$ is infinite.

Let us assign to each predicate variable q_i in W_{AP} the minimal valid predicate of the vertex v_i for the interpreted program $(AP, \mathcal{I}, \bar{\gamma})$.

Note that since the execution sequence $\langle AP, \mathcal{I}, \bar{\gamma} \rangle$ is infinite, i.e. control never reaches the halt vertex, it follows that the predicate F is the minimal valid predicate of the vertex H for the interpreted program $(AP, \mathcal{I}, \bar{\gamma})$.

Let Γ consist of the above assignments for the q_i's, with $\bar{\gamma}$ assigned to $\bar{y}$. Following the construction of W_{AP}, it is clear that the value of $(W_{AP}, \mathcal{I}, \Gamma)$ is T; i.e. $(W_{AP}, \mathcal{I})$ is satisfiable; this completes the proof in one direction.

(2) If $(W_{AP}, \mathcal{I})$ is satisfiable then $(AP, \mathcal{I})$ does not terminate.

If $(W_{AP}, \mathcal{I})$ is satisfiable, then there exists an assignment Γ for $(W_{AP}, \mathcal{I})$ such that the value of $(W_{AP}, \mathcal{I}, \Gamma)$ is T. Γ consists of assignments of specified total

[10] Note that the input variables $\bar{y}$ are free variables in W_{AP}.

predicates δ_i, mapping $(D_\mathcal{I})^n$ into {T, F}, for the predicate variables q_i, and an assignment $\bar{\gamma}, \bar{\gamma} \in (D_\mathcal{I})^m$, for the free variables $\bar{y}$.

By the construction of W_{AP}, this implies that each δ_i is a valid predicate of the vertex v_i for $(AP, \mathcal{I}, \bar{\gamma})$, and therefore that F is a valid predicate of the halt vertex for $(AP, \mathcal{I}, \bar{\gamma})$. This implies that the execution sequence $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ is infinite (i.e. execution does not reach the halt vertex). So $(AP, \mathcal{I})$ does not terminate. Q.E.D.

3.3. Termination of Abstract Programs

Definition 2. An abstract program AP is said to *terminate* if for every interpretation $\mathcal{I}$ the program $(AP, \mathcal{I})$ terminates.

The following theorem follows from Theorem 1 and Definition 2.

Theorem 2. *An abstract program AP terminates iff W_{AP} is unsatisfiable (or equivalently, $\sim W_{AP}$ is valid).*

Proof. AP terminates iff the program $(AP, \mathcal{I})$ terminates for every interpretation $\mathcal{I}$ iff $(W_{AP}, \mathcal{I})$ is unsatisfiable for every interpretation $\mathcal{I}$ iff W_{AP} is unsatisfiable. Q.E.D.

Theorem 2 transforms completely the problem of the termination of abstract programs into an equivalent problem in logic. This enables us to obtain many results about the problem of the termination of abstract programs just by using well-known results in logic. In the remainder of this section several such results are presented.

It is known that *the termination problem of abstract programs is undecidable* (see Luckham, Park, and Paterson [10]). But, since the validity problem of the predicate calculus is semidecidable, we have, from Theorem 1,

Corollary 1. *The termination problem of abstract programs is semidecidable.*

Moreover, any known semidecision procedure for solving the validity problem of the predicate calculus can be used, together with Algorithm 1, as a semidecision procedure for solving the termination problem of abstract programs.

Though the termination problem of abstract programs is undecidable, there nevertheless exist subclasses of abstract programs for which the termination problem is decidable.

Corollary 2. *The termination problem for the following classes is decidable:*

1. $C_1 = \{AP \mid AP$ *is an abstract program without function constants* $f_i^n, n \geq 1\}$.

2. $C_2 = \{AP \mid AP$ *is an abstract program which has only one program variable* x (i.e. $n = 1$), *and all the occurrences of function constants in* AP *are in terms of the form* f_i^0 *or* $f_i^1(x)\}$.

3. $C_3 = \{AP \mid AP$ *is an abstract program which has only two program variables* x_1 *and* x_2 (i.e. $n = 2$), *and all the occurrences of function constants in* AP *are in terms of the form* f_i^0 *or* $f_i^2(x_1, x_2)\}$.

Proof. For each i, $1 \leq i \leq 3$, the decidability of the termination problem for the class C_i follows, by using Theorem 2, from the decidability of the validity problem for the class W_i (see Section 1.2).

For example, to prove the decidability of the termination problem for the class C_2 we use Theorem 2 and the decidability of the validity problem for the class $W_2 = \{W \mid W$ is a wff in prenex normal form without function constants and with prefix of the form $\forall \cdots \forall\exists\forall \cdots \forall\}$. The proof of the assertion for the other classes is similar.

If AP is any member of the class C_2, it has only one program variable x (i.e.

$n = 1$), and all the occurrences of function constants in AP are in terms of the form $f_1^0, f_2^0, \cdots, f_k^0$ and $f_1^1(x), f_2^1(x), \cdots, f_l^1(x)$ $(k, l \geq 0)$. Then W_{AP} is of the form $(x)M$ where M is a quantifier-free wff and all the occurrences of function constants in M are in terms of the form $f_1^0, f_2^0, \cdots, f_k^0$ and $f_1^1(x), f_2^1(x), \cdots, f_l^1(x)$.

Let W'_{AP} be the wff $(\exists w_1) \cdots (\exists w_k)(x)(\exists z_1) \cdots (\exists z_l)M'$ where M' is the result of substituting w_i, $i = 1, 2, \cdots, k$, for each occurrence of f_i^0 in M and substituting z_i, $i = 1, 2, \cdots, l$, for each occurrence of $f_i^1(x)$ in M; i.e. M' contains no function constants. *W'_{AP} is satisfiable if and only if W_{AP} is satisfiable,* since W_{AP} is the functional form of W'_{AP}.

Let W''_{AP} be the wff $(w_1) \cdots (w_k)(\exists x)(z_1) \cdots (z_l)[\sim M']$; i.e. W''_{AP} is just $\sim W'_{AP}$. Clearly, *W''_{AP} is valid if and only if W'_{AP} is unsatisfiable.*

Because W''_{AP} is a member of W_2, and the validity problem for the class W_2 is decidable, it is decidable whether W''_{AP} is valid or not. Since by the previous assertions W''_{AP} is valid if and only if AP terminates, this implies that it is decidable whether AP terminates or not. Q.E.D.

Any decision procedure for solving the validity problem for the class W_i can be used, together with Algorithm 1, as a decision procedure for solving the termination problem for the class C_i. For example, we can use Friedman's semidecision procedure for the predicate calculus [8], which is a decision procedure for the classes W_1, W_2, and W_3.

Note that the abstract program AP^* of Section 2.1 belongs to the class C_2.

In Sections 3.4 and 3.5 it is shown that the correctness and the equivalence problems of abstract programs can be reduced to the termination problem.

3.4. Correctness of Abstract Programs

Definition 3. A triple (AP, Φ, ψ) is said to be *compatible* if AP is an abstract program with program variables $\bar{x}$ and input variables $\bar{y}$, $\Phi(\bar{y})$ is a wff with no free individual variables other than $\bar{y}$, and $\psi(\bar{y}, \bar{x})$ is a wff with no free individual variables other than $\bar{y}$ and $\bar{x}$. The wffs $\Phi(\bar{y})$ and $\psi(\bar{y}, \bar{x})$ do not contain any predicate variables.

Definition 4. Let (AP, Φ, ψ) be a compatible triple. Then the abstract program AP is said to be *correct with respect to Φ and ψ* if for every interpretation $\mathcal{I}$ (that contains assignments for all the constants that occur in AP, Φ, or ψ) and for every $\bar{\gamma}$ (such that $\Phi(\bar{\gamma}) = T$), $\psi(\bar{\gamma}, \text{val } \langle AP, \mathcal{I}, \bar{\gamma}\rangle) = T$ and the execution sequence $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ is finite.

Theorem 3. *For every compatible triple (AP, Φ, ψ) there exists an abstract program AP' such that AP is correct with respect to Φ and ψ iff AP' terminates.*

Proof. Consider the abstract program AP' represented by Figure 4.

3.5. Equivalence of Abstract Programs

Definition 5. The triple (AP, AP', Φ) is said to be *compatible* if AP and AP' are abstract programs with the same set of program variables $\bar{x}$, and the same set of input variables[11] $\bar{y}$. $\Phi(\bar{y})$ is a wff with no free individual variables other than $\bar{y}$ and does not contain any predicate variables.

Definition 6. Let (AP, AP', Φ) be a compatible triple. Then the abstract pro-

[11] Note that any two abstract programs can be considered as satisfying the second condition, for if the two abstract programs do not have the same sets of input variables, just add to each program an appropriate set of dummy input variables.

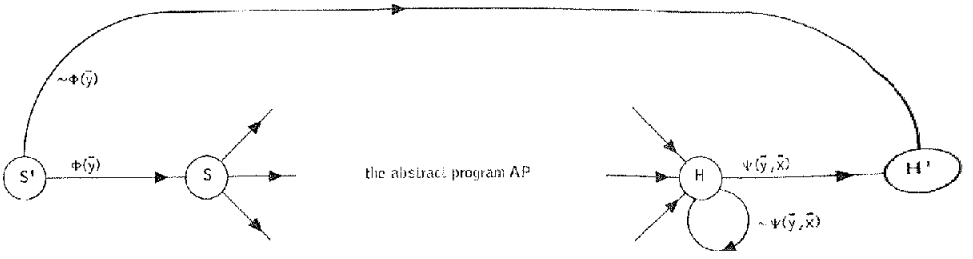

FIG. 4. The abstract program AP'

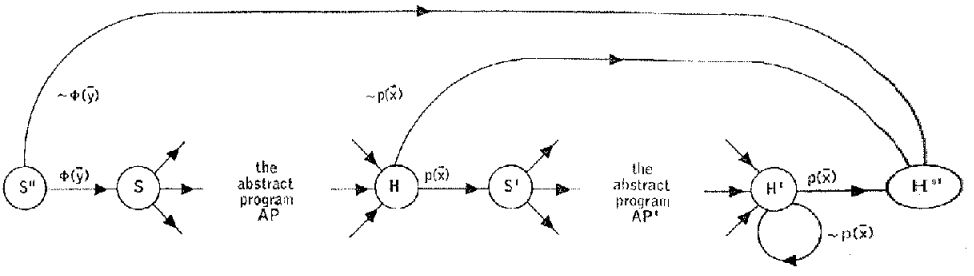

FIG. 5. The abstract program AP''

grams AP and AP' are said to be *equivalent with respect to* Φ if for every interpretation $\mathcal{I}$ (that contains assignments for all the constants that occur in Φ, AP, or AP') and for every $\bar{\gamma}$ (such that $\Phi(\bar{\gamma}) = T$) both execution sequences $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ and $\langle AP', \mathcal{I}, \bar{\gamma}\rangle$ are finite and val $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ = val $\langle AP', \mathcal{I}, \bar{\gamma}\rangle$.

THEOREM 4. *For every compatible triple* (AP, AP', Φ) *there exists an abstract program* AP'' *such that* AP *and* AP' *are equivalent with respect to* Φ *iff* AP'' *terminates.*

PROOF. Consider the abstract program AP'' represented by Figure 5. p is any predicate constant of n arguments that does not occur in Φ, AP, or AP'.

The reader can verify easily that AP and AP' are not equivalent with respect to Φ if and only if AP'' does not terminate. Note that AP and AP' are not equivalent with respect to Φ if and only if there exist an interpretation $\mathcal{I}$ and a $\bar{\gamma}$, $\Phi(\bar{\gamma}) = T$, such that: (1) $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ is infinite; or (2) $\langle AP', \mathcal{I}, \bar{\gamma}\rangle$ is infinite; or (3) both $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ and $\langle AP', \mathcal{I}, \bar{\gamma}\rangle$ are finite and $\bar{\xi} \neq \bar{\xi}'$ (where $\bar{\xi}$ = val $\langle AP, \mathcal{I}, \bar{\gamma}\rangle$ and $\bar{\xi}'$ = val $\langle AP', \mathcal{I}, \bar{\gamma}\rangle$). If for some assignment for p, $p(\bar{\xi}) = T$ and $p(\bar{\xi}') = F$ then $\bar{\xi} \neq \bar{\xi}'$, and conversely, if $\bar{\xi} \neq \bar{\xi}'$ then there exists an assignment for p such that $p(\bar{\xi}) = T$ and $p(\bar{\xi}') = F$.

ACKNOWLEDGMENTS. I am indebted to Robert Floyd for his help and encouragement throughout this research. Also of great value was the help of Alan Perlis, John McCarthy, Peter Andrews, Donald Loveland, Martin Davis, David Cooper, Richard Waldinger, and James King.

REFERENCES

1. ACKERMANN, W. *Solvable Cases of the Decision Problem.* North-Holland Publishing Co., Amsterdam, 1954.
2. BÜCHI, J. B. Turing machines and the Entscheidungsproblem. *Math. Ann. 148* (1962), 201–213.

3. CHURCH, A. *Introduction to Mathematical Logic, Vol. 1.* Princeton U. Press, Princeton, N.J., 1956.
4. COOPER, D. C. Program scheme equivalences and second order logic. Fourth Ann. Machine Intelligence Workshop, U. of Edinburgh, Aug. 1968.
5. DAVIS, M., AND PUTNAM, H. A computing procedure for quantification theory. *J. ACM 7*, 3 (July 1960), 201–215.
6. ENGELER, E. Algorithmic properties of structures. *Math. Syst. Theory 1*, 3 (1967), 183–195.
7. FLOYD, R. W. Assigning meaning to programs. Proc. Symp. Appl. Math., Amer. Math. Soc., Vol. 19, 1967, 19–32.
8. FRIEDMAN, J. A semi-decision procedure for the functional calculus. *J. ACM 10*, 1 (Jan. 1963), 1–24.
9. KAPLAN, D. M. The formal theoretic analysis of strong equivalence for elemental programs. Ph.D. Th., Computer Science Dept., Stanford U., Stanford, Calif., June 1968.
10. LUCKHAM, D. C., PARK, D. M. R., AND PATERSON, M. S. On formalised computer programs. Program. Res. Group, Oxford U., England, Aug. 1967.
11. MANNA, Z. Termination of algorithms. Ph.D. Th., Computer Science Dept., Carnegie-Mellon U., Pittsburgh, Pa., April 1968.
12. MANNA, Z. Formalization of properties of programs. Memo No. AI-64, Stanford Artificial Intelligence Rep., Stanford, Calif., July 1968.
13. MANNA, Z., AND PNUELI, A. Formalization of properties of recursively defined functions. To be presented at ACM Symp. on Theory of Computation, Marina del Rey, Calif., May 1969.
14. PATERSON, M. S. Equivalence problems in a model of computation. Ph.D. Th., U. of Cambridge, Cambridge, England, Aug. 1967.
15. ROBINSON, J. A. A machine-oriented logic based on the resolution principle. *J. ACM 12*, 1 (Jan. 1965), 23–41.
16. RUTLEDGE, J. D. On Ianov's program schemata. *J. ACM 11*, 1 (Jan. 1964), 1–9.
17. TURING, A. M. On computable numbers with an application to the Entscheidungsproblem. *Proc. London Math. Soc.* [2], *42* (1936–7), 230–265; correction: *43* (1937), 544–546.

RECEIVED JUNE, 1968; REVISED NOVEMBER, 1968