The use of first-order predicate calculus in proving correctness and other properties of programs is shown to be possible in practical situations.

The necessary concepts and theory are explained, and some practical examples worked through.

The application of formal logic to programs and programming

C. D. Allen

Considerable progress has recently been made in the application of the methods of formal logic to the proof of properties of computer programs. Starting from the work of McCarthy and Floyd,[1,2] formal methods have been developed by Manna, Ashcroft, and others to the point where standard methods can be usefully applied to a wide class of programs to prove their correctness and to discover their limitations, if any. (By limitations, we mean restrictions on the range of inputs for which they will produce correct results.)

The formalism used is that of mathematics and predicate calculus of the first order, with occasional excursions into elementary set theory.[3] Where necessary, we make use of other axiomatic theories, particularly that of the natural numbers.[3,4]

In this paper, we attempt to show the kind of results that can be obtained, the techniques required, and their application to practical programs. Very little of the material presented is original with the present author. Theorem 4 is believed to be so, and the examples were constructed for this paper. The sources of the remainder are listed in the References.

The theory is presented as necessary from the basic logical and set theoretic ideas. These latter are summarized briefly in the first

section to make the paper self-contained; however broader knowledge of this basis such as given in References 3 and 4 is helpful. The second section develops the techniques along the lines of Manna[5-8] and Ashcroft[9,10] in as elementary a fashion as possible, while retaining rigor.

The logical basis

truth, falsehood, and propositions

Truth, falsehood, and propositions are the basic concepts of formal (symbolic, mathematical) logic. The concept of a *proposition* may be informally described as "any statement that is true or false." Thus, "2 is less than 3" and "twice 4 is 7" are propositions—the first true, the second false.[11] In the formalism, propositions are represented by lower-case letters, p, q, etc., and their *truth-values*, "true" or "false," by T and F, respectively.

the combination of propositions

Various operators are defined with propositions as operands, giving new propositions. The truth-values of such constructed propositions depend only on the truth-values of their constituents. We shall use the following operators and notation:

$\neg p$	not p, true if p is false and vice versa
$p \ \& \ q$	p and q, true if and only if each of p and q is true
$p \vee q$	p or q, false if and only if each of p and q is false
$p \supset q$	p implies q, false if and only if p is true and q is false
$p \equiv q$	p is equivalent to q, true if and only if p and q are both true or both false

The order of precedence of these operators in expressions is that in which they are written above, with $\neg$ highest. We may note that, with these definitions:

$$\neg(p \ \& \ q) \equiv \neg p \vee \neg q$$

$$\neg(p \vee q) \equiv \neg p \ \& \ \neg q$$

$$p \supset q \equiv \neg p \vee q$$

$$\neg(p \supset q) \equiv p \ \& \ \neg q$$

A syntactically correct expression (formula) formed from proposition symbols and these operators is called a *well-formed formula* (wff) and itself stands for a proposition.

tautologies

A tautology is a wff whose truth-value is T whatever truth-values its constituents may have. For example

$$p \ \& \ q \supset p$$

is a tautology. We may evaluate its truth-value in all possible cases as in Table 1.

Table 1 Truth-values of $p \ \& \ q \supset p$

p	q	$p \ \& \ q$	$p \ \& \ q \supset q$
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	T

propositional calculus

A formal system called the *propositional calculus* may be set up with a set of *axioms* (propositions assumed to be true) and a *rule of inference*, which allows one to infer true propositions from other propositions. Within such a system, a *deduction* of one proposition from other propositions may be defined as follows. A proposition q is deduced from a set P of propositions $p_1, p_2, \cdots$ in a *deduction step* if one of the following holds:

1. *Substitution:* q is obtained from p_i in P by substituting a wff for all occurrences of one or more of the propositional variables in p_i.
2. *Modus ponens:* propositions of the forms p and $p \supset q$ are each in P.

A *deduction* of q_n from a set P of propositions is a sequence of propositions $q_1, q_2, \cdots, q_n$ such that each q_i is obtained from the propositions P and/or $q_1, q_2, \cdots, q_{i-1}$ by one or more deduction steps.

If q can be obtained from $p_1, p_2, \cdots, p_n$ by deduction, we write:

$$p_1, p_2, \cdots, p_n \vdash q$$

(Usually we omit the axioms from such a list of p's. Thus if q is deducible from the axioms and r, we write:

$$r \vdash q,$$

and if q is deducible from the axioms alone:

$$\vdash q.)$$

Given an appropriate set of axioms (see for example References 3 and 4), we may prove the following theorems.

The completeness theorem—If p is a tautology, then $\vdash p$.

The deduction theorem—If $P, p \vdash q$, then $P \vdash p \supset q$ and conversely.

Using these theorems, we may derive rules of inference more powerful than modus ponens. For example:

$$(p \supset q) \supset ((p \supset r) \supset (p \supset q \ \& \ r))$$

is a tautology. Thus, by the completeness theorem

$$\vdash (p \supset q) \supset ((p \supset r) \supset (p \supset q \ \& \ r))$$

whence, by the deduction theorem:

$$p \supset q \vdash (p \supset r) \supset (p \supset q \ \& \ r)$$

whence, by the deduction theorem again:

$$p \supset q, p \supset r \vdash p \supset q \ \& \ r$$

and this may be used in modus ponens to deduce $p \supset q \ \& \ r$ from $p \supset q$ and $p \supset r$.

predicates

In developing a theory, we are interested in obtaining general results, such as:

$$x^2 - 1 = (x + 1)(x - 1)$$

This, as it stands, is not a proposition, since its truth can only be determined after substituting a value of x. With such a substitution, the statement becomes a proposition. Such statements are called *predicates*; they are usually written like functions, with their variables made explicit; thus $p(x, y)$ may be a predicate with variables x and y.

Propositions may be obtained from predicates by substitution for their variables. However, the resulting statement usually is a meaningful proposition only if the values substituted are chosen from some restricted set. (Thus the predicate of x given above is meaningful if the values of x are numbers.) Such sets are called *domains*

For choice of the variable from a given domain, there are three possibilities for a predicate $p(x)$:

1. It may be true for *all* values of x from the domain.
2. It may be true for *some, but not all* values from the domain.
3. It may be true for *no* values from the domain.

In case 1, the predicate is said to be *valid* in (or over) the domain; in cases 1 and 2 it is said to be *satisfiable* in the domain; in case 3 it is said to be *unsatisfiable* in the domain.

quantifiers

There is a second way in which a proposition may be obtained from a predicate, namely *quantification*. For example, "$p(x)$ is valid in the domain D of x" is a proposition, since its truth does not depend on any particular value of x being substituted. This proposition is written

$$(\forall x)(p(x))$$

and usually read "for all x, $p(x)$." The part $(\forall x)$ is called a *universal quantifier*.

Again, "$p(x)$ is satisfiable" is a proposition (we shall omit references to a domain except where a particular domain is essential to the argument). This is written

$$(\exists x)(p(x))$$

and usually read "there exists an x such that $p(x)$." The part $(\exists x)$ is called an *existential quantifier*.

Note that quantifiers refer to one specific variable. From a two-variable predicate, $p(x, y)$, we may form the eight propositions:

1. $(\forall x)(\forall y)(p(x, y))$
2. $(\forall y)(\forall x)(p(x, y))$
3. $(\exists x)(\forall y)(p(x, y))$
4. $(\forall y)(\exists x)(p(x, y))$
5. $(\forall x)(\exists y)(p(x, y))$
6. $(\exists y)(\forall x)(p(x, y))$
7. $(\exists x)(\exists y)(p(x, y))$
8. $(\exists y)(\exists x)(p(x, y))$

Note that successive quantifiers are applied from right to left; thus:

$$(\exists x)(\forall y)(p(x, y)) \equiv (\exists x)[(\forall y)(p(x, y))]$$

Also note that 1 and 2 are equivalent, as are 7 and 8, but that the other pairs are not equivalent to each other. (Consider $p(x, y)$ to be $y = x^2$; then 3 is false but 4 is true in the domain of complex numbers, while 5 is true and 6 false.)

We shall write

$(\forall x, y)(p(x, y))$ for $(\forall x)(\forall y)(p(x, y))$ and

$(\exists x, y)(p(x, y))$ for $(\exists x)(\exists y)(p(x, y))$

With predicates of one variable, we have

$$\neg(\forall x)(p(x)) \equiv (\exists x)(\neg p(x)) \quad \text{and}$$

$$\neg(\exists x)(p(x)) \equiv (\forall x)(\neg p(x))$$

Note that the definition of wff is extended to allow the inclusion of propositions formed in this way.

Frequently in deductions we shall omit universal quantifiers. An expression such as:

$$q_1(x) \supset q_2(x)$$

in which the x is not quantified we shall take to stand for the proposition:

$$(\forall x)(q_1(x) \supset q_2(x))$$

and similarly for expressions with more variables. Such unquantified variables are called *free* in the expression and may be substituted for in a deduction step. Note also that substitution for explicitly quantified variables does not alter the meaning of the expression; for example,

$$(\forall x)(p(x)) \equiv (\forall y)(p(y))$$

However, we do not allow quantification of predicate symbols, although in the above sense they are free in expressions. Predicate symbols are to be understood as "place holders," into which particular predicate definitions are eventually to be inserted, and not as variables. It is only under this condition that the completeness theorem applies; the calculus restricted in this way is called *first-order predicate calculus.*

predicates and sets

A predicate $p(x)$ determines a set of values of x, namely those for which $p(x)$ is true. This set is written:

$$\{x \mid p(x)\}$$

i.e., "the set of x's such that $p(x)$." If $p(x)$ is valid in a domain D, then

$$\{x \mid p(x)\} = D \qquad \text{and}$$

$$x \in D \supset p(x)$$

If $p(x)$ is unsatisfiable in a domain D, then

$$\{x \mid p(x)\} = \{\ \} \quad \text{i.e., the } \textit{empty set}, \text{ and}$$

$$x \in D \supset \neg p(x)$$

Note that for any set P[12] of elements in a domain, there is a corresponding predicate; at worst we may write it as:

$$p(x) = x \in P$$

but, in most cases of interest, we can give an alternative definition of the predicate, not involving the set itself.

The empty set, $\{\ \}$, corresponds to the "constant" predicate F (false), and the whole domain, D, to T (true).

If:

$$p(x) \equiv x \in P, \qquad q(x) \equiv x \in Q, \quad \text{and} \quad p(x) \supset q(x),$$

then P is a subset of Q, and conversely. (Consider the truth-values of p and q for elements of P.) Also, if

$$(\forall x)(p(x) \equiv q(x)), \qquad \text{then}$$

$$P = Q$$

and conversely.

partial functions and predicates

If a variable x is restricted to a particular domain, D, a function $f(x)$ may or may not be defined for all values of x in the domain. If it is, it is said to be *total*; if not, it is said to be *partial*, over or in the domain D. Thus, for example, the reciprocal function, $1/y$, is partial over the real numbers, being undefined for $y = 0$. However, it is total over the natural numbers. Note that by defining

$$f(y) = 1/y \quad \text{if} \quad y \neq 0$$
$$= 10 \quad \text{if} \quad y = 0$$

we produce a function total over the real numbers, which may serve the same purpose as $1/y$ in many circumstances. Predicates are functions of their variables producing values of T or F. Thus they too may be total or partial. (Consider: $1/y < 0$.)

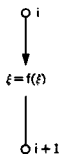
Proofs of program properties

In this section, we examine the application of logic to programs.[5-7,9,10]

the basic theory

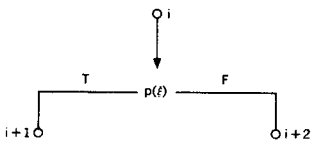
We first consider programs described by flow charts and/or functional definitions. The units of a flow chart are of two types: function boxes, as shown in Figure 1, and test boxes, as shown in Figure 2.

Figure 1 Sample function box



In Figure 1, ξ is a collection of variables on which the program acts and/or which may affect the program's action. Thus, all quantities, such as the contents of machine registers, control blocks, etc., which can affect the results of program execution at any point must be included, as must those stored values that may be changed by execution of the program. This collection of values describes completely that part of the system in which the program is being executed that is relevant to the program. It must contain enough information to determine the succeeding action of the program. We refer to this set of values as "the state." The function f takes part or all of the state as argument and gives new values for the state as result. It need not be defined for all possible values of its argument, i.e., it may be partial over the domain of states.

Figure 2 Sample test box



In Figure 2, p is a predicate on states (assumed total), and exit from the box is to node $i + 1$ if $p(\xi)$ is true for the state on entry and to node $i + 2$ if $p(\xi)$ is false for this state.

(The test of a partial predicate, defined only over part of the domain of states, can be replaced by evaluation of the predicate followed by a test on its truth-value, stored as an extra variable in the state.)

We shall sometimes refer to paths in such a flow chart: a function box provides a path from its entry node to its exit node; a test box provides two paths from its entry node to its exit nodes.

Functions may be defined by conditional expressions in the manner of McCarthy.[1] They may be nested, and their definitions may be recursive. Such a definition indicates an algorithm for evaluation of the function; e.g., the definition:

$$f(n) = (n = 0 \rightarrow 1, n \neq 0 \rightarrow n.f(n - 1))$$

(consisting of two clauses each giving one case of the definition) gives the value 1 for $n = 0$ directly. For other positive integral values of n, it indicates how the value may be obtained by recursive use of the definition. Thus, for $n = 5$, we calculate the sequence:

$5.f(4)$, $5.4.f(3)$, $5.4.3.f(2)$, $5.4.3.2.f(1)$,

$5.4.3.2.1.f(0)$, $5.4.3.2.1.1$

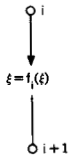
which terminates when the first clause of the definition is used.

We use the term *point* to mean a point in execution of a program which control may be said to have reached. Thus it includes all nodes of a flow chart program, and also the points where a function definition such as that above is entered or left. There is a set of identifiable values of program variables at each such point. The term *path* is used to denote a possible flow of control between points, including points within function definitions. Considering two adjacent points in a program, the operations on the state that take place between them are designed to produce specific properties of the state variables; e.g., a multiplication produces the property that one variable is the product of two others. The operations between particular points are chosen by the programmer on the assumption that the state already has some properties; e.g., that one of the variables is positive and another nonzero. These properties can be expressed formally as predicates, $q(\xi)$, of the state. The determinate action of the program between the nodes sets up a relation between such properties: *if* the earlier state has the assumed properties summarized in $q_i(\xi)$, *then* the latter will have the derived properties summarized in $q_{i+1}(\xi)$. This "if $\cdots$ then" relation is formalized as implication. Other conditions may be included in this implication; in particular, the conditions that must hold in order that a function evaluation shall succeed with a defined value $f(x)$ for input x may be summarized in a predicate $t(x, f(x))$ and included with q_i.

Thus we may set up wff's of the predicate calculus giving general expression to each such relation, and attempt to deduce formally (and thus verifiably) from this basis the properties of the program. Hence with each node of a program, we associate a predicate symbol q_i (with a distinct subscript), assumed to take a state as an argument. With each function that may be evaluated in the course of execution, we associate a predicate symbol t_i (with a distinct subscript) assumed to take two states as arguments. Using these symbols,

and free state variables ξ, ξ', $\cdots$, $\xi^{(n)}$, we construct a set of wff's of the first-order predicate calculus. The complete rules for such constructions are very lengthy and detailed, and we do not attempt to state them here. Later we shall make explicit the objectives of the construction; when these are understood, the rules can be easily generated for every specific case. We give some of the simpler rules now. They may be intuitively justified by taking the predicate $q_i(\xi)$ to mean "the associated node is reached during execution with state ξ," and the predicate $t_i(\xi, \xi')$ to mean "if the associated function evaluation is started with state ξ, then it will terminate with state ξ'."

Figure 3 Function box



Thus, for a function box of a flow chart, as shown in Figure 3, the wff is:

$$q_i(\xi) \ \& \ t_i(\xi, \xi') \supset q_{i+1}(\xi')$$

For a test box of a flow chart, as shown in Figure 2, two wff's are formed, one for each path:

$$q_i(\xi) \ \& \ p(\xi) \supset q_{i+1}(\xi)$$

$$q_i(\xi) \ \& \ \neg p(\xi) \supset q_{i+2}(\xi)$$

For a function definition of the form:

$$f_1(\xi) =$$
$$p_2(\xi) \rightarrow f_2(\xi)$$
$$p_3(\xi) \rightarrow f_3(\xi)$$
$$\vdots$$
$$p_n(\xi) \rightarrow f_n(\xi)$$

the wff's are:

$$q_1(\xi) \ \& \ p_2(\xi) \supset q_2(\xi)$$

$$q_1(\xi) \ \& \ \neg p_2(\xi) \ \& \ p_3(\xi) \supset q_3(\xi)$$
$$\vdots$$

$$q_1(\xi) \ \& \ \neg p_2(\xi) \ \& \ \neg p_3(\xi) \ \& \ \cdots \ p_n(\xi) \supset q_n(\xi)$$

and:

$$q_1(\xi) \ \& \ p_2(\xi) \ \& \ t_2(\xi, \xi') \supset t_1(\xi, \xi')$$

$$q_1(\xi) \ \& \ \neg p_2(\xi) \ \& \ p_3(\xi) \ \& \ t_3(\xi, \xi') \supset t_1(\xi, \xi')$$
$$\vdots$$

$$q_1(\xi) \ \& \ \neg p_2(\xi) \ \& \ \neg p_3(\xi) \ \& \ \cdots \ p_n(\xi) \ \& \ t_n(\xi, \xi') \supset t_1(\xi, \xi')$$

For a function definition of the form:

$$f_1(\xi) = f_n \circ f_{n-1} \circ \cdots f_2(\xi) = f_n(f_{n-1}(\cdots (f_2(\xi)) \cdots))$$

the wff's are:

$$q_1(\xi) \supset q_2(\xi)$$

$$q_1(\xi) \ \& \ t_2(\xi, \xi') \supset q_3(\xi')$$

$$\vdots$$

$$q_1(\xi) \ \& \ t_2(\xi, \xi') \ \& \cdots t_n(\xi^{(n-1)}, \xi^{(n)}) \supset t_1(\xi, \xi^{(n)})$$

where, in the last two examples, the subscripts have been chosen to match those of the function definition. In practice, all the q's and t's in the W formulas for a complete program must be given distinct subscripts.

In the last two cases, we have omitted some q predicates, on the grounds that the formulas as given are immediately deducible from those including additional q's. For example, for the function f_1 in the last example, we could have written:

$$q_1(\xi) \supset q_2(\xi)$$

$$q_2(\xi) \ \& \ t_2(\xi, \xi') \supset q_3(\xi')$$

$$q_3(\xi') \ \& \ t_3(\xi', \xi'') \supset q_4(\xi'')$$

etc. From these we may deduce the formulas given; as we shall see later, the set given is adequate for our purposes and somewhat briefer.

Note that we must, in the process of definition, ultimately reach "basic" functions, i.e., functions that are defined by axioms, or left undefined. If $f(\xi)$ is such a function, under the intuitive meaning of the t predicates given above, we have that

$$q(\xi) \supset t(\xi, f(\xi))$$

which is the W formula for such functions. Thus the formation of W formulas terminates when such functions are reached. If f is a partial function, and the predicate $p(\xi)$ is true if and only if ξ is in the domain of f, the formula is modified to

$$q(\xi) \ \& \ p(\xi) \supset t(\xi, f(\xi))$$

We call the full set of wff's obtained in this way for a program "the W formulas" for the program.

We claim that the formal system, based on appropriately constructed

W formulas as axioms, is related to execution of the program in a specific and useful way.

In what follows, we shall be considering the use of specific predicates in place of the symbols q_i and t_i. To keep the description general, we shall use ϕ_i and τ_i to denote specific predicates being substituted for q_i and t_i, respectively.

We shall also be concerned with sets of executions of the program determined as follows. We assume that, for the programs we are considering, one execution is completely determined by the initial state ξ with which execution starts at node 1. Given a specific predicate, $\phi_1(\xi)$, this determines a set of states ξ for which $\phi_1(\xi)$ is true; each of these determines a single execution of the program. The whole set of such executions we call "executions of the program with initial condition ϕ_1."

We now turn to the question of the properties we wish to consider of the state at the various points. To determine exactly what is happening in execution of a program, the properties one needs are such as to distinguish between values that do arise in execution and values that do not. Thus if the domain of x is the integers and x has only the values 1, 2, 4, 5 when execution reaches point i, an appropriate predicate would be:

$$q_i(x) = 0 < x \leq 5 \ \& \ x \neq 3$$

Such a predicate, true only of the values that do arise in execution, is called a "minimal valid" predicate, and tells only the truth.

However, we may need to know merely that at this point x is positive—e.g., to ensure success of the square root function that follows. A predicate:

$$q_i(x) = x \geq 0$$

would thus be adequate, even though it is true for values other than those arising in execution. Such a predicate is called a "valid" predicate, and tells us more than the truth. (Minimal valid predicates are minimal in the sense that they are satisfied by a smaller set of values than valid predicates.)

More precise definitions of these terms follow.

Definition 1: The set $\phi_2, \phi_3, \cdots, \phi_n, \phi_o, \tau_1, \tau_2, \cdots$ of predicates is called "a set of valid (ϕ's) and convergence (τ's) predicates" or "VC predicates" for the program for initial condition ϕ_1 if they satisfy the following criterion. In the set of executions of the program with initial condition ϕ_1, $\phi_i(\xi)$ is true for all states with which the associated node is reached, and $\tau_i(\xi, \xi')$ is true for all pairs of states

ξ and ξ' with which an evaluation of the associated function starts and finishes, respectively.

Note that VC predicates may be true for states other than those mentioned in the definition.

Definition 2: The set $\phi_2, \phi_3, \cdots, \phi_n, \phi_o, \tau_1, \tau_2, \cdots$ of predicates is called "a set of minimal valid (ϕ's) and convergence (τ's) predicates" or "MVC predicates" for the program with initial condition ϕ_1 if they satisfy the following criterion. In the set of executions of the program with initial condition ϕ_1, $\phi_i(\xi)$ is true *if and only if* the associated node is reached with state ξ, and $\tau(\xi, \xi')$ is true *if and only if* an evaluation of the associated function starts and terminates with states ξ, ξ', respectively.

Note that, under these definitions, a set of MVC predicates is a set of VC predicates, but the converse may not be true.

Examples and theorems

example 1

A trivial example may help to make these ideas clear. Consider the program shown in Figure 4, in which

$$f_5(x) = 2x$$

The W formulas for this program are:

W1: $q_1(x)\ \&\ (x \geq 5) \supset q_2(x)$

W2: $q_1(x)\ \&\ (x < 5) \supset q_3(x)$

W3: $q_2(x) \supset q_4(7)$

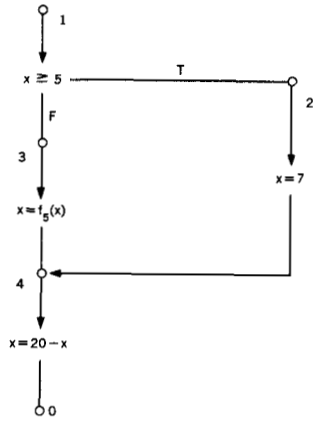
W4: $q_3(x) \supset q_5(x)$

W5: $q_3(x)\ \&\ t_5(x, x') \supset q_4(x')$

W6: $q_4(x) \supset q_o(20 - x)$

W7: $q_5(x) \supset t_5(x, 2x)$

Figure 4 Example 1



Here we have assumed that $20-x$, and $2x$ are total functions of x, so that no t predicates for them are required. (For we have, for example:

$$q_4(x) \supset t_{20}(x, 20 - x)$$

$$q_4(x)\ \&\ t_{20}(x, y) \supset q_o(y)$$

from which we may deduce immediately:

$$q_4(x) \supset q_o(20 - x)$$

which we have used as W6. If there exist predicates ϕ_4, τ_{20}, and ϕ_o that satisfy the first two equations, they also satisfy the third, since it is deducible from them. Conversely, if ϕ_4 and ϕ_o satisfy the third equation, then

$$\tau_{20}(x, y) = \phi_4(x) \ \& \ \phi_o(y)$$

together with ϕ_4 and ϕ_o satisfies the first two. Thus the elimination of t in this manner does not affect the following arguments.)

If we take as an initial condition

$\phi_1(x)$: $\quad x \geq 0$

then we note that, for such initial x's, at node 4, x must be less than 10. Thus,

$\phi_2(x)$: $\quad x \geq 5$

$\phi_3(x)$: $\quad 0 \leq x < 5$

$\phi_4(x)$: $\quad 0 \leq x < 10$

$\phi_5(x)$: $\quad 0 \leq x < 5$

$\phi_o(x)$: $\quad 10 < x \leq 20$

$\tau_5(x, x')$: $\quad x' = 2x$

is a set of VC predicates for initial condition ϕ_1. Note, for example, that $\phi_4(x)$ is true for $x = 3$, a value that never arises in execution. For the same initial condition:

$\phi_2(x)$: $\quad x \geq 5$

$\phi_3(x)$: $\quad 0 \leq x < 5$

$\phi_4(x)$: $\quad x \in \{0, 2, 4, 6, 7, 8\}$

$\phi_5(x)$: $\quad 0 \leq x < 5$

$\phi_o(x)$: $\quad x \in \{20, 18, 16, 14, 13, 12\}$

$\tau_5(x, x')$: $\quad (x' = 2x) \ \& \ 0 \leq x < 5$

may be seen to be a set of MVC's.

By the definitions, if $\{\phi_i^m, \tau_i^m\}$ is a set of MVC predicates and $\{\phi_i, \tau_i\}$ a set of VC predicates, the set of states or pairs of states for which ϕ_i^m or τ_i^m is true is included in the set of states or pairs of states for which ϕ_i or τ_i is true. Thus:

$$\phi_i^m(\xi) \supset \phi_i(\xi)$$

and

$\tau_i^m(\xi, \xi') \supset \tau_i(\xi, \xi')$

Further, all MVC sets are logically equivalent, since they determine the same sets of states and state pairs.

In the example above, since only a finite number of states could arise in execution, it is not difficult to verify that the predicates chosen are VC and MVC sets (by notionally running all distinct cases). Generally this is not possible; however, a test for VC sets is given by Theorem 1 below. This test is sufficient, but not necessary; as we shall see by example, some VC sets do not satisfy this test. However, Theorem 2 shows that the MVC set must satisfy this test, so that the test is useful.

The basic results of Manna, as extended by Ashcroft, are the following theorems:

Theorem 1: If a set $\{\phi_i, \tau_i\}$ of predicates and an initial predicate ϕ_1 satisfy the W formulas for a program when substituted for the q_i and t_i (i.e., the resulting formulas are true), then the set is a VC set for initial condition ϕ_1.

A sketch of a proof of this takes the following form. First we show that, for a W formula, if the q's and t's on the left are replaced by members of a VC set and if the formula is true, the q or t on the right must be in a VC set. For example, for the formula:

$q_i(\xi)\ \&\ p(\xi) \supset q_{i+1}(\xi)$

obtained from one leg of a test box, we argue as follows. If $\phi_i(\xi)$ is in a VC set and if the test box is entered with a state ξ_1 (in any of the set of executions), then

$\phi_i(\xi_1)$

is true. If $p(\xi_1)$ is true, then the T exit will be taken with state ξ_1. Also if

$\phi_i(\xi)\ \&\ p(\xi) \supset \phi_{i+1}(\xi)$

is true, since both terms on the left are true for ξ_1,

$\phi_{i+1}(\xi_1)$

is true. Since this argument holds for all appropriate ξ_1's, there must be a VC set containing ϕ_i and ϕ_{i+1}.

Using a similar result for each W formula, we see that, since all the W formulas are true, there must be a VC set containing all the ϕ's and τ's. The only exceptional cases are the formulas involving q_1, for example,

$$q_1(\xi) \ \& \ t_1(\xi, \xi') \supset q_2(\xi')$$

In such cases, the argument is modified to say that if

$$\phi_1(\xi) \ \& \ \tau_1(\xi, \xi') \supset \phi_2(\xi')$$

is true and τ_1 is in a VC set for an execution such that $\phi_1(\xi_1)$ is true, then ϕ_2 is in a VC set containing τ_1 for initial condition ϕ_1.

Theorem 2: The set of minimal valid and convergence predicates for the program for initial condition ϕ_1 satisfies the W formulas when substituted for the q's and t's with ϕ_1 substituted for q_1. A similar sketch of a proof may be given. For example, for the formula

$$q_i(\xi) \ \& \ t_j(\xi, \xi') \supset q_k(\xi')$$

(corresponding to nodes i and k being connected through an evaluation of a function f_j), if $\phi_i(\xi)$ and $\tau_j(\xi, \xi')$ are in an MVC set, then by definition $\phi_i(\xi_1)$ and $\tau_j(\xi_1, \xi_2)$ are only true if execution reached node i with state ξ_1 and if the function evaluated with ξ_1 produced ξ_2. In such cases, execution must reach node k with state ξ_2; hence, since $\phi_k(\xi)$ is in the MVC set, $\phi_k(\xi_2)$ must be true. Thus

$$\phi_i(\xi) \ \& \ \tau_i(\xi, \xi') \supset \phi_k(\xi')$$

Again, the argument for formulas involving ϕ_1 depend on ϕ_1 being true rather than being in the MVC set. But the set is for initial condition ϕ_1, i.e., it is an MVC set for executions for which $\phi_1(\xi)$ *is* true for the states at node 1, which gives us what we require.

Returning to Example 1, consider again the W formulas and the set of predicates stated to be a VC set for initial condition $x \geq 0$. Substituting in the W formulas, we have:

W1: $x \geq 0 \ \& \ x \geq 5 \supset x \geq 5$

W2: $x \geq 0 \ \& \ x < 5 \supset 0 \leq x < 5$

W3: $x \geq 5 \supset 0 \leq 7 < 10$

W4: $0 \leq x < 5 \supset 0 \leq x < 5$

W5: $0 \leq x < 5 \ \& \ x' = 2x \supset 0 \leq x' < 10$

W6: $0 \leq x < 10 \supset 10 < 20 - x \leq 20$

W7: $0 \leq x < 5 \supset 2x = 2x$

and these are all true. Thus, by Theorem 1, the predicates are indeed a VC set for initial condition $x \geq 0$.

Note that there may well be VC sets that do not satisfy the W formulas. For example, if we take the above set but change $\phi_3(x)$ to

$x \geq 0$, the set is still a VC set, but does not satisfy W4. Note also that in checking that these predicates satisfy the W formulas, we have used properties of the operators $<$, $=$, etc., between states, i.e., we are using results from the theory of the domain of states.

The objectives of the specific construction of the W formulas from specific paths in the program is that the proofs of Theorem 1 and Theorem 2 can be carried through with them.

In the light of the above theorems, we consider the cases in which F is a member of an MVC set, for some node i. Since a minimal valid predicate is true for precisely those states with which this node is reached, if F is in an MVC set, the node can never be reached. In particular, if F is the predicate associated with the exit node, the program does not terminate. Conversely, if a program does not terminate, F is the minimal valid predicate for the exit node. F is also valid for this node (as indeed is any predicate). Thus we can *prove* termination by proving that the minimal valid predicate for node 0 *cannot* be F. By Theorem 2, this will be so if there is no set of predicates satisfying the W formulas with F substituted for q_0, ϕ_1 for q_1, and the initial condition ϕ_1 assumed to be true. Since these formulas (with substitution for q_1 and q_0 only) are wff's of the first-order predicate calculus, and since this theory is both complete and consistent, this is so if the formulas are not consistent; i.e., from them one can deduce a contradiction.

Theorem 3: If the W formulas, with ϕ_1 substituted for q_1 and F substituted for q_0, are inconsistent with the truth of ϕ_1, then the program terminates for initial condition ϕ_1 (Manna, extended by Ashcroft).

Continuing with our trivial example, with the required substitutions we obtain:

1. $x \geq 0 \;\&\; x \geq 5 \supset q_2(x)$
2. $x \geq 0 \;\&\; x < 5 \supset q_3(x)$
3. $q_2(x) \supset q_4(7)$
4. $q_3(x) \supset q_5(x)$
5. $q_3(x) \;\&\; t_5(x, x') \supset q_4(x')$
6. $q_4(x) \supset F$
7. $q_5(x) \supset t_5(x, 2x)$

and from 6 we obtain immediately:

8. $\neg q_4(x)$

and deduce (from the expressions identified at the right) as follows:

9. $\neg q_2(x)$ 8, 3

10. $x < 0 \vee x < 5$ 9, 1

11. $\neg q_3(x) \vee \neg t_5(x, x')$ 8, 5

12. $\neg q_3(x) \vee \neg q_5(x)$ 11, 7

13. $\neg q_3(x)$ 12, 4

14. $x < 0 \vee x \geq 5$ 13, 2

15. $x < 0$ 14, 10

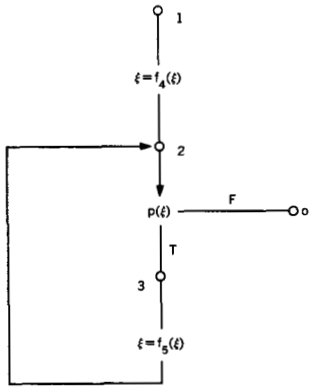
which contradicts $\phi_1(x)$, i.e., $x \geq 0$ in this case. Hence, the program terminates for $x \geq 0$.

example 2

As a more significant example, we take the program in Figure 5.

Without further specification of the functions involved, this represents the general case of a single loop, with $f_4(\xi)$ being the initialization operations and $f_5(\xi)$ the computations in the loop. From this information alone, we can prove that, under some specific assumptions on f_4, f_5, and p, the program will terminate for any input state. We thus obtain a very general result concerning loops. We have formulas as follows:

Figure 5 Example 2



W1: $q_1(\xi) \supset q_4(\xi)$

W2: $q_1(\xi)\ \&\ t_4(\xi, \xi') \supset q_2(\xi')$

W3: $q_2(\xi)\ \&\ p(\xi) \supset q_3(\xi)$

W4: $q_2(\xi)\ \&\ \neg p(\xi) \supset q_o(\xi)$

W5: $q_3(\xi) \supset q_5(\xi)$

W6: $q_3(\xi)\ \&\ t_5(\xi, \xi') \supset q_2(\xi')$

W7: $q_4(\xi) \supset t_4(\xi, f_4(\xi))$

W8: $q_5(\xi) \supset t_5(\xi, f_5(\xi))$

where we have assumed that the functions involved always terminate.

If we substitute ϕ_1 for q_1 and F for q_o in W1, W2, and W4, we obtain:

1. $\phi_1(\xi) \supset q_4(\xi)$

2. $\phi_1(\xi)\ \&\ t_4(\xi, \xi') \supset q_2(\xi')$

3. $\neg q_2(\xi) \vee p(\xi)$

From these and the remaining W formulas, we deduce:

4. $q_2(\xi)\ \&\ p(\xi) \supset q_5(\xi)$ W3, W5

5. $\supset t_5(\xi, f_5(\xi))$ 4, W8

6. $\supset q_3(\xi)\ \&\ t_5(\xi, f_5(\xi))$ — 5, W3

7. $\supset q_2(f_5(\xi))$ — 6, W6

and by repeated use of 7:

8. $q_2(\xi)\ \&\ (0 \leq r < n \supset p(f_5^r(\xi))) \supset q_2(f_5^n(\xi))$

where we write $f_5^r(\xi)$ for $f_5 \circ f_5 \circ \cdots \circ f_5(\xi)$ with r applications of the function, and $f_5^0(\xi) = \xi$.

Now suppose that, for some k:

A9. $\neg p(f_5^k(\xi_1))$

A10. $0 \leq r < k \supset p(f_5^r(\xi_1))$

then

11. $q_2(\xi_1) \supset q_2(f_5^k(\xi_1))$ — 8, A10

12. $\neg q_2(f_5^k(\xi_1))$ — 3, A9

13. $\neg q_2(\xi_1)$ — 11, 12

14. $\phi_1(\xi) \supset t_4(\xi, f_4(\xi))$ — 1, W7

15. $\supset q_2(f_4(\xi))$ — 14, 2

Hence by 13 with $\xi_1 = f_4(\xi)$

16. $\neg\phi_1(\xi)$

Thus, under assumptions A9, A10, with $\xi_1 = f_4(\xi)$, this program terminates for input condition $\phi_1(\xi)$. Since this holds for all ϕ_1, assumptions A9 and A10 ensure termination of the program for *all* inputs. As a simple application of this result, consider the program in which

$\xi = (n, X)$

n being an integer variable and X the remainder of the state variables. Let:

$f_4(n, X) = (n_1, X)$

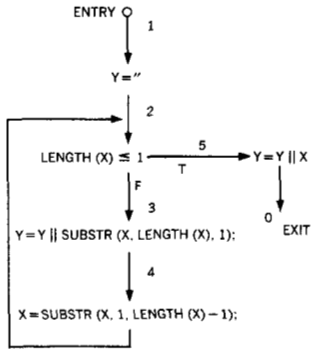
where n_1 is a constant value ≥ 0,

$f_5(n, X) = (n - 1, X)$

$p(n, X) = (n \neq 0)$

Then with:

Figure 6 Example 3



$\xi_1 = f_4(n, X) = (n_1, X)$

$f_5^k(\xi_1) = (n_1 - k, X)$

and A9 and A10 become:

$n_1 - k = 0$

$0 \leq r < k \supset n_1 - r \neq 0$

Remembering that $n_1 \geq 0$, each of these is satisfied by $k = n_1$. Hence the loop terminates.

The functions f_4 and f_5 could include other operations on X that do not alter n without affecting the above argument. Thus we have shown that setting a counter positive or zero, testing for zero at the head of the loop, and counting down within the loop will ensure termination.

Practical applications

example 3

As another practical example, consider the (PL/I) program shown in Table 2 written to reverse a string. The flow chart for this program is shown in Figure 6.

This program uses variables X, Y, which are varying length character strings, the constants '' and 1, functions LENGTH, SUBSTR, and operations || (concatenation), $\leq$, $-$, and assignment. We use the following notations:

N for '' (the null string)

$b(A)$ for LENGTH (A), where A is a string

$X[p{:}q]$ for SUBSTR $(X, p, q-p+1)$, where X is a string and p and q are such that SUBSTR is defined for these arguments

Table 2 Example 3 listing

```
REVERSE: PROC(X);
         DCL X CHAR(*) VARYING,
             Y CHAR(LENGTH(X)) VARYING;
         Y = '';
L2:      IF LENGTH (X) < = 1
         THEN GO TO L5;
         Y = Y || SUBSTR (X, LENGTH(X), 1);
         X = SUBSTR (X, 1, LENGTH (X) −1);
         GO TO L2;
L5:      Y = Y || X;
         END;
```

$X[p{:}q] = S$, an arbitrary string, when p and q are outside the ranges for which SUBSTR $(X, p, q-p+1)$ is defined

$X[p]$ for $X[p : p]$

Note that LENGTH, and hence b, are total functions over strings.

The W formulas for this program are as follows:

W1: $q_1(X, Y) \supset q_2(X, N)$

W2: $q_2(X, Y) \;\&\; b(X) \leq 1 \supset q_5(X, Y)$

W3: $q_2(X, Y) \;\&\; b(X) > 1 \supset q_3(X, Y)$

W4: $q_3(X, Y) \supset q_F(X, b(X), 1)$

W5: $q_3(X, Y) \;\&\; t_F(X, b(X), 1, Z) \supset q_4(X, Y \,||\, Z)$

W6: $q_4(X, Y) \supset q_F(X, 1, b(X) - 1)$

W7: $q_4(X, Y) \;\&\; t_F(X, 1, b(X) - 1, Z) \supset q_2(Z, Y)$

W8: $q_5(X, Y) \supset q_o(X, Y \,||\, X)$

and, from the definition of SUBSTR in PL/I:

W9: $q_F(X, i, j) \;\&\; (1 \leq i \leq b(X)) \;\&\; (0 \leq j \leq b(X))$
$\;\&\; (i + j - 1 \leq b(X)) \supset t_F(X, i, j, X[i{:}\, i + j - 1])$

To prove termination, we take the domains of X, Y, Z to be strings, and of i, j to be integers. We set:

1. $q_1(X, Y) = \phi_1(X, Y)$ and
2. $q_o(X, Y) = \phi_o(X, Y) = F$

and deduce as follows:

3. $q_2(X, Y) \;\&\; b(X) > 1 \supset q_F(X, b(X), 1)$ W3, W4
4. $q_F(X, b(X), 1)$
$\;\&\; (1 \leq b(X) \leq b(X)) \;\&\; (0 \leq 1 \leq b(X))$
$\;\&\; (b(X) \leq b(X)) \supset t_F(X, b(X), 1, X[b(X){:}\, b(X)])$ W9
5. $q_F(X, b(X), 1) \;\&\; b(X) \geq 1$
$\;\supset t_F(X, b(X), 1, X[b(X)])$ 4
6. $q_2(X, Y) \;\&\; b(X) > 1 \supset t_F(X, b(X), 1, X[b(X)])$ 3, 5
7. $\supset q_4(X, Y \,||\, X[b(X)])$ W3, 6, W5
8. $\supset q_F(X, 1, b(X) - 1)$ W6

9. $q_F(X, 1, b(X) - 1)\ \&\ (1 \leq 1 \leq b(X))$
$\&\ (0 \leq b(X) - 1 \leq b(X))\ \&\ (b(X) - 1 \leq b(X))$
$\supset t_F(X, 1, b(X) - 1, X[1: b(X) - 1])$ W9

10. $q_F(X, 1, b(X) - 1)\ \&\ b(X) \geq 1$
$\supset t_F(X, 1, b(X) - 1, X[1: b(X) - 1])$ 9

11. $q_2(X, Y)\ \&\ b(X) > 1$
$\supset q_2(X[1: b(X) - 1], Y \,||\, X[b(X)])$ 7, 10, W7

We now prove by induction

$q_2(X, Y)\ \&\ b(X) > r + 1 \supset p(r + 1, X, Y)$, where

12. $p(r, X, Y)$
$= q_2(X[1: b(X) - r], Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[b(X) - r + 1])$

as follows:

13. $q_2(X[1: b(X) - r],$
$Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[b(X) - r + 1])\ \&$
$b(X) - r > 1 \supset q_2(X[1: b(X) - r - 1],$
$Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[b(X) - r])$

by 11, noting that:

$b(X[1: n]) = n$, and if
$Z = X[1: b(X) - r]$ then
$Z[1: b(Z) - 1] = Z[1: b(X) - r - 1]$
$= X[1: b(X) - r - 1]$ and
$Z[b(Z)] = Z[b(X) - r] = X[b(X) - r]$

14. $p(r, X, Y)\ \&\ b(X) > r + 1 \supset p(r + 1, X, Y)$ 12, 13

15. $q_2(X, Y)\ \&\ b(X) > 1 \supset p(1, X, Y)$ 11

16. $q_2(X, Y)\ \&\ b(X) > r + 1 \supset p(r + 1, X, Y)$

by induction from 14 and 15.

17. $q_2(X, Y)\ \&\ b(X) > b(X) - 1 \supset p(b(X) - 1, X, Y)$ 16

18. $q_2(X, Y) \supset q_2(X[1: 1], Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[2])$ 17, 12

19. $\phi_1(X, Y) \supset q_2(X, N)$ W1, 1

20. $\supset q_2(X[1], N \,||\, X[b(X)] \,||\, \cdots \,||\, X[2])$ 18, 19

21. $q_2(X[1]; Z)\ \&\ b(X[1]) \leq 1 \supset q_5(X[1], Z)$ W2

22. $\phi_1(X, Y) \supset q_5(X[1], N \,||\, X[b(X)] \,||\, \cdots \,||\, X[2])$ 20, 21

23. $\neg q_5(X, Y)$ W8, 2

24. $\neg\phi_1(X, Y)$ 22, 23

Thus we have a contradiction with $\phi_1(X, Y)$, and by Theorem 3 the program therefore terminates for initial condition ϕ_1. In particular, it terminates for $\phi_1 = T$, i.e., for all inputs (restricted only to the domain of X, Y, i.e., strings).

the function computed

The program was constructed to produce the reverse of an input string X (in Y on termination). In our notation, the reverse of A may be written as:

$$A[b(A)] \,||\, A[b(A) - 1] \,||\, \cdots \,||\, A[1]$$

Thus, to show that the program is correct, we need to show that:

1. $Y = A[b(A)] \,||\, A[b(A) - 1] \,||\, \cdots \,||\, A[1]$

is a valid predicate for the exit node, where A is the initial value of X, i.e., the initial condition $\phi_1(X, Y)$ is:

2. $X = A$

Thus we require to show that a complete set of predicates, including those above for ϕ_0 and ϕ_1, respectively, when substituted for the q's and t's in the W formulas for the program, make them true. From an examination of the program, we make intelligent guesses for the remaining predicates of the set as follows.

Firstly, we define some notational abbreviations:

3. $a = b(A)$

4. $p_2(X, Y) = (\exists n)(X = A[1{:}\, n]$
$\quad \& \; Y = A[a] \,||\, A[a - 1] \,||\, \cdots \,||\, A[n + 1] \;\&\; 1 \leq n < a)$

5. $p_3(X, Y) = (\exists n)(X = A[1{:}\, n]$
$\quad \& \; Y = A[a] \,||\, A[a - 1] \,||\, \cdots \,||\, A[n + 1] \;\&\; 1 < n < a)$

6. $p_4(X, Y) = (\exists n)(X = A[1{:}\, n]$
$\quad \& \; Y = A[a] \,||\, A[a - 1] \,||\, \cdots \,||\, A[n] \;\&\; 1 < n < a)$

Then take:

7. $\phi_1(X, Y) = (X = A)$

8. $\phi_2(X, Y) = p_2(X, Y) \vee X = A \;\&\; Y = N$

9. $\phi_3(X, Y) = p_3(X, Y) \vee X = A \;\&\; Y = N \;\&\; a > 1$

10. $\phi_4(X, Y) = p_4(X, Y) \vee X = A \;\&\; Y = A[a] \;\&\; a > 1$

11. $\phi_5(X, Y) = (X = A[1]$
$\quad \&\; Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[2]$
$\quad \vee\; X = A \;\&\; Y = N \;\&\; a \leq 1)$

12. $\phi_6(X, Y) = (Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[1]$
$\quad \vee\; Y = N \;\&\; a = 0)$

13. $\phi_F(X, Y) = 1 \leq i \leq b(X)$
$\quad \&\; 0 \leq j \leq b(X) \;\&\; i + j - 1 \leq b(X)$

14. $\tau_F(X, i, j, Y) = (Y = X[i: i + j - 1])$

Then, substituting in the W formulas, we show that we obtain true formulas. For example, for W5 we have:

15. $(p_3(X, Y) \vee X = A \;\&\; Y = N \;\&\; a > 1) \;\&\; Z = Z[b(X)]$
$\quad \supset p_4(X, Y \,||\, Z) \vee X = A \;\&\; Y \,||\, Z = A[a] \;\&\; a > 1$

Proof: Assume that $p_3(X, Y)$ is true, i.e.,

16. $(\exists n)(X = A[1: n]$
$\quad \&\; Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[n+1] \;\&\; 1 < n < a)$

and let n_1 be such an n, then:

17. $X = A[1: n_1]$

18. $Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[n_1 + 1]$

19. $1 < n_1 < a$, whence:

20. $Z = X[b(X)]$
$\quad \supset Z = A[n_1]$ 17

21. $\quad \supset Y \,||\, Z = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[n_1]$ 18, 20

22. $\quad \supset p_4(X, Y \,||\, Z)$ 17, 21, 19

n_1 being such an n. Thus

23. $p_3(X, Y) \;\&\; Z = X[b(X)]$
$\quad \supset p_4(X, Y \,||\, Z)$ 22

24. $\supset p_4(X, Y \,||\, Z) \vee X = A \;\&\; Y \,||\, Z = A[a] \;\&\; a > 1$ 23

Also we have:

25. $Y = N \;\&\; Z = X[b(X)] \supset Y \,||\, Z = X[b(X)]$

26. $X = A \;\&\; Y = N \;\&\; Z = X[b(X)]$

$\supset Y \,||\, Z = A[b(A)]$ 25

27. $\supset Y \,||\, Z = A[a]$ 26, 3

28. $X = A \;\&\; Y = N \;\&\; a > 1 \;\&\; Z = X[b(X)]$

$\supset X = A \;\&\; Y \,||\, Z = A[a] \;\&\; a > 1$ 27

29. $(p_3(X, Y) \vee X = A \;\&\; Y = N \;\&\; a > 1)$

$\&\; Z = X[b(X)] \supset p_4(X, Y \,||\, Z) \vee X = A$

$\&\; Y \,||\, Z = A[a] \;\&\; a > 1$ 23, 28

Q.E.D.

Again, taking W2 we have:

30. $(p_2(X, Y) \vee X = A \;\&\; Y = N)$

$\&\; b(X) \leq 1 \supset X = A[1]$

$\&\; Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[2]$

$\vee X = A \;\&\; Y = N \;\&\; a \leq 1$

Proof: Assume that $p_2(X, Y)$ is true, i.e.

31. $(\exists n)(X = A[1{:}\, n]$

$\&\; Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[n+1]$

$\&\; 1 \leq n < a)$

and let n_1 be such an n. Then:

32. $X = A[1{:}\, n_1]$

33. $Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[n_1 + 1]$

34. $1 \leq n_1 < a$

35. $b(X) \leq 1 \supset n_1 \leq 1$ 32

36. $\supset n_1 = 1$ 34, 35

37. $\supset Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[2]$ 36, 33

38. $\supset X = A[1]$ 32, 36

39. $\supset X = A[1]$

$\&\; Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[2]$ 37, 38

40. $p_2(X, Y) \;\&\; b(X) \leq 1$

$\supset X = A[1]$

$\&\; Y = A[a] \,||\, A[a-1] \,||\, \cdots \,||\, A[2]$ 31, 39

41. $\supset C$ 40

where C is the right-hand side of 30.

42. $X = A \;\&\; Y = N \;\&\; b(X) \leq 1$

$\supset a \leq 1$

43. $\supset X = A \;\&\; Y = N \;\&\; a \leq 1$ 42

44. $\supset C$ 43

45. $(p_2(X, Y) \vee X = A \;\&\; Y = N) \;\&\; b(X) \leq 1 \supset C$ 41, 44

Q.E.D.

Similarly, we may prove that the remaining W formulas, when the ϕ and τ are substituted for the q and t, are true. Thus, by Theorem 1, $\phi_0(X, Y)$ is a valid predicate for the exit node for input $\phi_1(X, Y)$. Hence for inputs $X = A$ (and any Y), the program computes the reverse of A, in Y on termination.

minimal valid predicate

So far, it appears that the proof of termination can tell us nothing concerning the function actually computed when and if the program terminates. We are interested particularly in discovering a *minimal* valid predicate for the output point, since it is this that excludes all values that are not computed. However, the methods presented so far only tell us whether a given predicate is valid, not whether it is minimal, and they only do this if we can construct a whole set of predicates for the program.

If a program successfully computes $f(\xi_1)$ from an input state ξ_1, the minimal predicate for the output point is:

$$\phi_0(\xi) = (\xi = f(\xi_1))$$

for input condition:

$$\phi_1(\xi) = (\xi = \xi_1)$$

In this section, we show how a proof of termination may be constructed that gives $f(\xi)$ explicitly if the construction succeeds, without any need to guess predicates for the intermediate points.

We now consider in more detail the sets of states corresponding to the predicates in a VC or MVC set, and the relations within and between VC sets and MVC sets.

Firstly, a given input condition $\phi_1(\xi)$ determines a set:

1. $S_1 = \{\xi \mid \phi_1(\xi)\}$

of input states; each of these determines a single execution of the program. The set of executions so determined gives rise to a set of states at each point of the program; the minimal valid predicates for input condition $\phi_1(\xi)$ correspond precisely with these sets. The minimal convergence (t) predicates determine sets of pairs of states (ξ, ξ') such that the corresponding function is entered with state ξ at least once in the set of executions, and then terminates with state ξ'. If T_i is the set of pairs for some function and if S_i and S_{i+1} are the minimal valid sets for the nodes on either side of the function, then we have:

2. $\xi \in S_i \ \& \ (\xi, \xi') \in T_i \supset \xi' \in S_{i+1}$

which is the set-theoretic form of the W formula:

3. $q_i(\xi) \ \& \ t_i(\xi, \xi') \supset q_{i+1}(\xi')$

Thus the minimal convergence predicates express the mapping between the input and output states of a function; the W formulas relating the q predicates express the mapping between the minimal valid sets at the appropriate nodes, set up by execution of the function or the program between the nodes. It is this property that ensures that the W formulas reflect the action of the program.

The sets corresponding to a VC set of predicates are supersets of those for the MVC set, that is, they contain the minimal valid sets, but more states besides. The fact that a set of predicates satisfies the W formulas corresponds to the correct mappings holding between these enlarged sets. Since implication corresponds to the inclusion relation between sets, the relations expressed by the W formulas allow the sets to become larger (become valid rather than minimal) but never to get smaller—thus they must always contain the minimal sets.

These relations between sets of states are the fundamental features of a program. By appealing directly to them, we may obtain some rather deeper results concerning the logic.

Thus, suppose that we can deduce from the W formulas a formula of the form:

4. $q_1(\xi) \ \& \ p(\xi) \supset q_0(f(\xi))$

where p is some predicate and f some function. (The deduction may involve properties of the states and the functions and operations used in the program.) Consider an input condition $\xi = \xi_1$, where ξ_1 is a single state such that:

5. $p(\xi_1)$

and let $\phi^m_o(\xi)$ be the member of the MVC set for the output node and this input condition. Then, by Theorem 2, the MVC set satisfies the W formulas; hence it satisfies 4 above, since this is deducible from the W formulas.

Thus:

6. $\xi = \xi_1 \;\&\; p(\xi) \supset \phi^m_o(f(\xi))$ i.e.,

7. $p(\xi_1) \supset \phi^m_o(f(\xi_1))$, whence

8. $\phi^m_o(f(\xi_1))$

since $p(\xi_1)$ is true. Now the set S_1 of inputs we are considering contains just the state ξ_1. We assume that the program is determinate, i.e., that for a single input state, there is not more than one possible output state. By 8 above, since ϕ^m_o is minimal, there is one output state, $f(\xi_1)$, hence

9. $S_o = \{f(\xi_1)\}$

Thus we have Theorem 4.

Theorem 4: If $q_1(\xi) \;\&\; p(\xi) \supset q_o(f(\xi))$ can be deduced from the W formulas, then for any input ξ satisfying p, the program terminates and computes $f(\xi)$.

If the whole program, regarded as a single function, is embedded in a larger program, it would be useful to have a single W formula for it. Theorem 4 shows that an appropriate formula is:

10. $q_i(\xi) \;\&\; p(\xi) \supset t_i(\xi, f(\xi))$

where $p(\xi)$ is that appearing in 4. This derivation allows us to treat programs piecemeal and to prove properties of parts independently, later carrying through a study of the whole program.

the examples concluded

Continuing with Example 2, from the W formulas for the loop program, we may deduce as before:

1. $q_2(\xi) \;\&\; (0 \leq r < n \supset p(f^r_5(\xi))) \supset q_2(f^n_5(\xi))$

Also:

2. $q_1(\xi) \supset t_4(\xi, f_4(\xi))$ — W1, W7

3. $\supset q_2(f_4(\xi))$ — 2, W2

4. $q_1(\xi) \;\&\; (0 \leq r < n \supset p(f^r_5 \circ f_4(\xi))) \supset q_2(f^n_5 \circ f_4(\xi))$ — 1, 3

5. $q_1(\xi) \;\&\; (0 \leq r < n \supset p(f^r_5 \circ f_4(\xi)))$
$\;\&\; \neg p(f^n_5 \circ f_4(\xi)) \supset q_o(f^n_5 \circ f_4(\xi))$ — 4, W4

and this formula has the required form. Hence, for initial states satisfying:

$$(0 \leq r < n \supset p(f_5^r \circ f_4(\xi_1))) \ \& \ \neg p(f_5^n \circ f_4(\xi_1))$$

the program terminates with state

$$f_5^n \circ f_4(\xi_1).$$

Note that if for some k

$$\neg p(f_5^k \circ f_4(\xi_1))$$

then the least number principle[3] applied to

$$P(r) = \neg p(f_5^r \circ f_4(\xi_1))$$

guarantees that there exists an n such that the condition for termination is true. Hence, a wider condition for termination is

$$(\exists k)(\neg p(f_5^k \circ f_4(\xi_1))$$

for the input state ξ_1, and the n in the formula for the output state is the least such k.

For the string reverser in Example 3, we may deduce from the W formulas a formula of the required form as below. Noting that there is a loop from node 2, we start there and deduce the function computed around the loop.

1.	$q_2(X, Y) \ \& \ b(X) > 1 \supset q_F(X, b(X), 1)$	W3, W4
2.	$b(X) > 1 \supset 1 \leq b(X) \leq b(X)$ $\& \ 0 \leq 1 \leq b(X) \ \& \ b(X) + 1 - 1 \leq b(X)$	
3.	$b(X) > 1 \ \& \ q_F(X, b(X), 1)$ $\supset t_F(X, b(X), 1, X[b(X): b(X)])$	2, W9
4.	$q_2(X, Y) \ \& \ b(X) > 1 \supset t_F(X, b(X), 1, X[b(X)])$	1, 3
5.	$q_2(X, Y) \ \& \ b(X) > 1 \supset q_4(X, Y \mid\mid X[b(X)])$	W3, 4, W5
6.	$\supset q_F(X, 1, b(X) - 1)$	5, W6
7.	$b(X) > 1 \supset 1 \leq 1 \leq b(X)$ $\& \ 0 \leq b(X) - 1 \leq b(X) \ \& \ b(X) - 1 \leq b(X)$	
8.	$q_F(X, 1, b(X) - 1) \ \& \ b(X) > 1$ $\supset t_F(X, 1, b(X) - 1, X[1: b(X) - 1])$	7, W9
9.	$q_2(X, Y) \ \& \ b(X) > 1$ $\supset q_2(X[1: b(X) - 1], Y \mid\mid X[b(X)])$	5, 6, 8, W7

10. $q_2(X[1: b(X) - r],$

$Y \,||\, X[b(X)] \,||\, X[b(X) - 1] \,||\, \cdots \,||\, X[b(X) - r + 1])$

$\& \; b(X) - r > 1 \supset q_2(X[1: b(X) - r - 1],$

$Y \,||\, X[b(X)] \,||\, X[b(X) - 1] \,||\, \cdots \,||\, X[b(X) - r])$ 9

By induction from 9 ($r = 1$) and 10:

11. $q_2(X, Y) \; \& \; b(X) > 1$

$\& \; b(X) > r + 1 \supset q_2(X[1: b(X) - r - 1],$

$Y \,||\, X[b(X)] \,||\, X[b(X) - r] \,||\, \cdots \,||\, X[b(X) - r])$

With $r = b(X) - 2$ this gives:

12. $q_2(X, Y) \; \& \; b(X) > 1$

$\supset q_2(X[1], Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[2])$

13. $q_2(X[1], Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[2]) \; \& \; b(X[1]) \leq 1$

$\supset q_5(X[1], Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[2])$ W2

14. $q_2(X, Y) \; \& \; b(X) > 1$

$\supset q_5(X[1], Y \,||\, X[b(X)] \,||\, \cdots \,||\, X[2])$ 12, 13

15. $q_1(X, Y) \; \& \; b(X) > 1$

$\supset q_5(X[1], X[b(X)] \,||\, \cdots \,||\, X[2])$ 14, W1

16. $\supset q_0(X[1], X[b(X)] \,||\, \cdots \,||\, X[1])$ 15, W8

17. $q_1(X, Y) \; \& \; b(X) \leq 1 \supset q_5(X, N)$ W1, W2

18. $\supset q_0(X, X)$ 17, W8

If we now define rev(X) by

19. $b(X) \geq 1 \supset \text{rev}\,(X) = X[b(X)] \,||\, \cdots \,||\, X[1]$

Table 3 Example 4 listing

```
MMULT: PROCEDURE (A, B, C);
       DECLARE (A(*, *), B(*, *), C(*, *)) FLOAT DECIMAL,
               (I, J, K) FIXED BINARY (31, 0);
       DO I = LBOUND (C, 1) TO HBOUND (C, 1);
          DO J = LBOUND (C, 2) TO HBOUND (C, 2);
          C (I, J) = 0;
            DO K = LBOUND (A, 2) TO HBOUND (A, 2);
            C (I, J) = C(I, J) + A(I, K) * B(K, J);
            END;
          END;
       END;
       END MMULT;
```

20. $b(X) = 0 \supset \text{rev}(X) = N$

and allow $N[1] = N$, then 16 and 18 give

21. $q_1(X, Y) \supset q_0(X[1], \text{rev}(X))$

Thus the program terminates with $Y = \text{rev}(X)$ for all inputs. (Since there is no $p(X)$ term on the right, there is no restriction on the inputs.)

Some further techniques

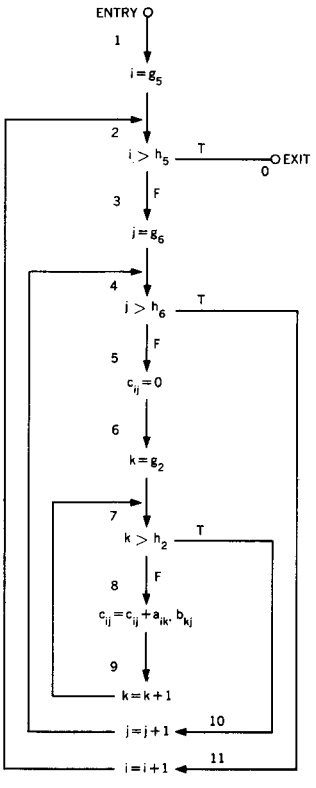
example 4

As our final example, to demonstrate some techniques in producing the required deductions, we take the PL/I program shown in Table 3.

We shall use the following abbreviations throughout this section:

g_1 = LBOUND (A, 1) h_1 = HBOUND (A, 1)
g_2 = LBOUND (A, 2) h_2 = HBOUND (A, 2)
g_3 = LBOUND (B, 1) h_3 = HBOUND (B, 1)
g_4 = LBOUND (B, 2) h_4 = HBOUND (B, 2)
g_5 = LBOUND (C, 1) h_5 = HBOUND (C, 1)
g_6 = LBOUND (C, 2) h_6 = HBOUND (C, 2)
A' = the transpose of the matrix A
A_i = the ith row of A (a vector)
A'_j = the jth column of A
a_{ij} = the element of A in the ith row and jth column of A and similar notations for B and C.

Figure 7 Example 4



The flow chart for the program is shown in Figure 7.

The DECLARE statements tell us the domain of the respective variables. We shall treat FIXED BINARY as integers, and FLOAT DECIMAL as real numbers, ignoring the differences from these domains. In PL/I, LBOUND and HBOUND are defined for the uses we make of them, so we treat them as total functions.

We first treat the section of program between nodes 5 and 10, noting that this section has one entry and one exit. Thus we attempt to derive the equivalent function, so that we may replace it by a single function box. The variables on which it operates are c_{ij} and k; i, j and A and B are parameters that are unchanged. Thus we take the "state" to be c_{ij} and k, and write W formulas as follows:

W1: $q_5(c_{ij}, k) \supset q_6(0, k)$

W2: $q_6(c_{ij}, k) \supset q_7(c_{ij}, g_2)$

W3: $q_7(c_{ij}, k) \;\&\; k > h_2 \supset q_{10}(c_{ij}, k)$

W4: $q_7(c_{ij}, k) \;\&\; k \leq h_2 \supset q_8(c_{ij}, k)$

W5: $q_8(c_{ij}, k) \supset q_9(c_{ij} + a_{ik}.b_{kj}, k)$

W6: $q_9(c_{ij}, k) \supset q_7(c_{ij}, k + 1)$

note 1 (In W5, we use the point to indicate multiplication.)

In constructing these formulas, we have ignored any possibility that references to elements of A, B, and C may fail because subscripts are outside of the declared range.

From the W formulas, we deduce the initial state at q_7 as follows:

1. $q_5(c_{ij}, k) \supset q_7(0, g_2)$ W1, W2

and the general function around the loop:

2. $q_7(c_{ij}, k)\ \&\ k \leq h_2 \supset q_9(c_{ij} + a_{ik}.b_{kj}, k)$ W4, W5

3. $\supset q_7(c_{ij} + a_{ik}.b_{kj}, k + 1)$ 2, W6

Thus, once around the loop adds $a_{ik}.b_{kj}$ to c_{ij} and 1 to k. From 1 and 3, it is evident that the general form of c_{ij} will be

$$\sum_{r=g_2}^{s} a_{ir}.b_{rj}$$

for some s. However, 0 (in formula 1) is not of this form, so we deduce one more turn around the loop:

4. $q_5(c_{ij}, k)\ \&\ g_2 \leq h_2 \supset q_7(a_{ig_2}.b_{g_2j}, g_2 + 1)$ 1, 3

and this gives the case $s = g_2$. From 3 we now deduce the recurrence relation:

5. $$q_7\left(\sum_{r=g_2}^{s} a_{ir}.b_{rj}, s + 1\right)$$
$$\&\ s + 1 \leq h_2 \supset q_7\left(\sum_{r=g_2}^{s+1} a_{ir}.b_{rj}, s + 2\right) \quad 3$$

whence, by induction from 4 and 5:

6. $$q_5(c_{ij}, k)\ \&\ g_2 \leq h_2\ \&\ s + 1 \leq h_2 \supset q_7\left(\sum_{r=g_2}^{s+1} a_{ir}.b_{rj}, s + 2\right)$$

We come out of the loop, by W3, when the second argument of q_7 is $> h_2$. Hence, in 6 we put $s = h_2 - 1$ and use W3 to get:

7. $q_5(c_{ij}, k)\ \&\ g_2 \leq h_2 \supset q_{10}\left(\sum_{r=g_2}^{h_2} a_{ir}.b_{rj}, h_2 + 1\right)$ 6, W3

We now have a formula of the desired form. To simplify it, we may recognize that the sum on the right-hand side is the dot product $A_i.B'_j$ if $g_2 = g_3$ and $h_2 = h_3$, i.e.

8. $g_2 = g_3 \;\&\; h_2 = h_3 \supset \sum_{r=g_2}^{h_2} a_{ir}.b_{rj} = A_i.B'_j$

9. $q_5(c_{ij}, k) \;\&\; g_2 \leq h_2 \;\&\; g_2 = g_3$
$\;\&\; h_2 = h_3 \supset q_{10}(A_i.B'_j, h_2 + 1)$ 7, 8

and by Theorem 4 we may write:

10. $q_5(c_{ij}, h) \;\&\; g_2 \leq h_2 \;\&\; g_2 = g_3$
$\;\&\; h_2 = h_3 \supset t_{5-10}(c_{ij}, k, A_i.B'_j, h_2 + 1)$

as a single W formula for this section of the program.

note 2

The termination of this section of the program is given by 7, subject only to the condition $g_2 \leq h_2$. The additional conditions in 10 are required so that it will compute the function $A_i.B'_j$ and not some other function. (Remembering Note 1, a deeper treatment of the logic of the statement:

$C(I, J) = C(I, J) + A(I, K) * B(K, J)$

would bring out in the logic that if, for example, h_3 were less than h_2, we would get an undefined result.)

note 3

The choice of a formula for the general form of c_{ij} while looping is essential to the success of this deduction. It is the problem of selecting a suitable form, which depends greatly on a knowledge of the program's intended operation and the properties of various forms in mathematics generally, which is to date almost impossible to do automatically in any sufficiently general fashion.

the intermediate loop

We now treat the section of program between nodes 3 and 11, replacing the section between nodes 5 and 10 by a function. The W formulas are:

W7: $q_3(C_i, j, k) \supset q_4(C_i, g_6, k)$

W8: $q_4(C_i, j, k) \;\&\; j > h_6 \supset q_{11}(C_i, j, k)$

W9: $q_4(C_i, j, k) \;\&\; j \leq h_6 \supset q_5(c_{ij}, k)$

W10: $q_4(C_i, j, k) \;\&\; j \leq h_6$
$\;\&\; t_{5-10}(c_{ij}, k, x, n) \supset q_{10}(f(C_i, x, j), j, n)$

W11: $q_{10}(C_i, j, k) \supset q_4(C_i, j + 1, k)$

and 10 of the last section:

W12: $q_5(c_{ij}, k)\ \&\ g_2 \leq h_2\ \&\ g_2 = g_3$

$\&\ h_2 = h_3 \supset t_{5-10}(c_{ij}, k, A_i.B'_j, h_2 + 1)$

where we have used

$f(C_i, x, j)$

for a function that gives the new C_i resulting from the substitution of x for its jth element.

note 4 Again we have written as arguments to the q predicates only those things that will be altered by this section of the program. In the case of q_5, we know from the previous section that only c_{ij} and k are altered between nodes 5 and 10, so we retain only these arguments. Their altered values are fitted back into the remainder by W10, using a function defined for this purpose.

From these formulas, we may deduce:

11. $q_4(C_i, j, k)\ \&\ j \leq h_6\ \&\ g_2 \leq h_2\ \&\ g_2 = g_3\ \&\ h_2 = h_3$
$\supset t_{5-10}(c_{ij}, k, A_i.B'_j, h_2 + 1)$ W9, W12

12. $\supset q_{10}(f(C_i, A_i.B'_j, j), j, h_2 + 1)$ 11, W10

13. $\supset q_4(f(C_i, A_i.B'_j, j), j + 1, h_2 + 1)$ 12, W11

The initial condition at q_4 is given directly by W7; thus we see that the general term in the first argument position of q_4 is going to be C_i with its first s (say) elements replaced by $A_i.B'_r$, for $g_6 \leq r \leq s$. Thus we define a function, P_i, to represent this, as:

14. $P_i(g_6) = f(C_i, A_i.B'_{g_6}, g_6)$

15. $P_i(g_6 + r) = f(P_i(g_6 + r - 1), A_i.B'_{g_6+r}, g_6 + r)$

Then we have:

16. $q_3(C_i, j, k)\ \&\ g_6 \leq h_6\ \&\ g_2 \leq h_2\ \&\ g_2 = g_3$
$\&\ h_2 = h_3 \supset q_4(P_i(g_6), g_6 + 1, h_2 + 1)$ W7, 13, 14

and

17. $q_4(P_i(g_6 + r), g_6 + r + 1, k)\ \&\ g_6 + r + 1 \leq h_6$
$\&\ g_2 \leq h_2\ \&\ g_2 = g_3\ \&\ h_2 = h_3$
$\supset q_4(P_i(g_6 + r + 1), g_6 + r + 2, h_2 + 1)$ 13, 15

Thus, by induction from 16 and 17:

18. $q_3(C_i, j, k)\ \&\ g_6 + r + 1 \leq h_6\ \&\ g_6 \leq h_6$
$\&\ g_2 \leq h_2\ \&\ g_2 = g_3\ \&\ h_2 = h_3$
$\supset q_4(P_i(g_6 + r + 1), g_6 + r + 2, h_2 + 1)$

The exit from this loop is taken, by W8, if the second argument to q_4 is $> h_6$. Thus in 18 we put $r = h_6 - g_6 - 1$ and obtain:

19. $q_3(C_i, j, k)\ \&\ g_6 \leq h_6$
$\&\ g_2 \leq h_2\ \&\ g_2 = g_3\ \&\ h_2 = h_3$
$\supset q_4(P_i(h_6), h_6 + 1, h_2 + 1)$ 18

20. $\supset q_{11}(P_i(h_6), h_6 + 1, h_2 + 1)$ 19, W8

This is of the form required for Theorem 4, so we get:

21. $q_3(C_i, j, k)\ \&\ g_6 \leq h_6\ \&\ g_2 \leq h_2\ \&\ g_2 = g_3\ \&\ h_2 = h_3$
$\supset t_{3-11}(C_i, j, k, P_i(h_6), h_6 + 1, h_2 + 1)$

the complete program

We now treat the whole program, substituting a function, with W formula 21 above, for the section between nodes 3 and 11. The W formulas are as follows:

W13: $q_1(C, i, j, k) \supset q_2(C, g_5, j, k)$

W14: $q_2(C, i, j, k)\ \&\ i > h_5 \supset q_0(C, i, j, k)$

W15: $q_2(C, i, j, k)\ \&\ i \leq h_5 \supset q_3(C_i, j, k)$

W16: $q_2(C, i, j, k)\ \&\ i \leq h_5\ \&\ t_{3-11}(C_i, j, k, D_i, m, n)$
$\supset q_{11}(G(C, D_i, i), i, m, n)$

W17: $q_{11}(C, i, j, k) \supset q_2(C, i + 1, j, k)$

and from 21 in the previous section:

W18: $q_3(C_i, j, k)$
$\&\ g_3 \leq h_6\ \&\ g_2 \leq h_2\ \&\ g_2 = g_3\ \&\ h_2 = h_3$
$\supset t_{3-11}(C_i, j, k, P_i(h_6), h_6 + 1, h_2 + 1)$

Here we have introduced the function $G(C, D_i, i)$ to denote the result of substituting D_i for the ith row of C. From these formulas we may deduce:

22. $q_2(C, i, j, k)\ \&\ i \leq h_5\ \&\ g_6 \leq h_6$
$\&\ g_2 \leq h_2\ \&\ g_2 = g_3\ \&\ h_2 = h_3$
$\supset t_{3-11}(C_i, j, k, P_i(h_6), h_6 + 1, h_2 + 1)$ W15, W18

23. $\supset q_{11}(G(C, P_i(h_6), i), i, h_6 + 1, h_2 + 1)$ 22, W16

24. $\supset q_2(G(C, P_i(h_6), i), i + 1, h_6 + 1, h_2 + 1)$ 23, W17

Now define:

25. $Q(g_5) = G(C, P_{g_5}(h_6), g_5)$

26. $Q(g_5 + r) = G(Q(g_5 + r - 1), P_{g_5+r}(h_6), g_5 + r)$

then:

27. $q_1(C, i, j, k) \;\&\; g_5 \leq h_5$
$\;\&\; g_6 \leq h_6 \;\&\; g_2 \leq h_2 \;\&\; g_2 = g_3 \;\&\; h_2 = h_3$
$\supset q_2(G(C, P_{g_5}(h_6), g_5), g_5 + 1, h_6 + 1, h_2 + 1)$ W13, 24

28. $\supset q_2(Q(g_5), g_5 + 1, h_6 + 1, h_2 + 1)$ 27, 25

29. $q_2(Q(g_5 + r - 1), g_5 + r, j, k) \;\&\; g_5 + r \leq h_5$
$\;\&\; g_6 \leq h_6 \;\&\; g_2 = g_3 \;\&\; h_2 = h_3$
$\supset q_2(Q(g_5 + r), g_5 + r + 1, h_6 + 1, h_2 + 1)$ 24, 26

and by induction from 28 and 29:

30. $q_1(C, i, j, k) \;\&\; g_5 + r \leq h_5 \;\&\; g_5 \leq h_5$
$\;\&\; g_6 \leq h_6 \;\&\; g_2 \leq h_2 \;\&\; g_2 = g_3 \;\&\; h_2 = h_3$
$\supset q_2(Q(g_5 + r), g_5 + r + 1, h_6 + 1, h_2 + 1)$

The exit from this loop occurs, by W14, when the second argument of q_2 is $> h_5$. Thus we put $r = h_5 - g_5$ and get:

31. $q_1(C, i, j, k) \;\&\; g_5 \leq h_5 \;\&\; g_6 \leq h_6$
$\;\&\; g_2 \leq h_2 \;\&\; g_2 = g_3 \;\&\; h_2 = h_3$
$\supset q_2(Q(h_5), h_5 + 1, h_6 + 1, h_2 + 1)$ 30

32. $\supset q_0(Q(h_5), h_5 + 1, h_6 + 1, h_2 + 1)$ 31, W14

Thus by Theorem 4, the program terminates if

$g_5 \leq h_5 \;\&\; g_6 \leq h_6 \;\&\; g_2 \leq h_2 \;\&\; g_2 = g_3 \;\&\; h_2 = h_3$

and computes $Q(h_5)$.

To see the significance of this, we must investigate the nature of $Q(h_5)$ in detail.

The relevant function definitions are collected below (with their original numbers):

25. $Q(g_5) = G(C, P_{g_5}(h_6), g_5)$

26. $Q(g_5 + r) = G(Q(g_5 + r - 1), P_{g_5+r}(h_6), g_5 + r)$

14. $P_i(g_6) = f(C_i, A_i.B'_{g_6}, g_6)$

15. $P_i(g_6 + r) = f(P_i(g_6 + r - 1), A_i.B'_{g_6+r}, g_6 + r)$

and the functions f and G were defined informally as follows:

$f(C_i, x, j) = C_i$ with x substituted for its jth element
$G(C, D_i, i) = C$ with D_i substituted for its ith row.

Thus from 14 and 15, $P_i(h_6)$ is a vector whose elements indexed by $j = g_6$ to h_6 are $A_i.B'_j$; these elements are exactly those of C_i.

Similarly, by 25 and 26, $Q(h_5)$ is a matrix whose rows, indexed by $i = g_5$ to h_5, are $P_i(h_6)$. These are exactly the rows of C. Thus the elements of $Q(h_5)$ are $A_i.B'_j$ for

$$g_5 \leq i \leq h_5 \ \& \ g_6 \leq i \leq h_6$$

and these are precisely the elements of $A.B$ if

33. $g_1 = g_5 \ \& \ h_1 = h_5 \ \& \ g_4 = g_6 \ \& \ h_4 = h_6$

So we have that the program computes $A.B$ under the conditions (from 32 and 33):

$$g_5 \leq h_5 \ \& \ g_6 \leq h_6 \ \& \ g_2 \leq h_2 \ \& \ g_2 = g_3 \ \& \ h_2 = h_3$$
$$\& \ g_1 = g_5 \ \& \ h_1 = h_5 \ \& \ g_4 = g_6 \ \& \ h_4 = h_6$$

Summary comment

The techniques required and results obtainable when applying logic to programs are well illustrated by Example 4. Firstly, it demonstrates the possibility of treating program segments in isolation, and using the results obtained directly in their condensed forms in a treatment of the containing program. With this technique, the amount of formalism involved at any one stage in the proof can be kept within reasonable bounds. It also shows that any program computing the dot product under the appropriate conditions may be substituted for the inner loop.

The use of recursive function definitions, such as those of P_i and Q, to define a function computed by a program parallels the treatment by McCarthy.[1] The example shows that, where desirable, deductions from such definitions may be postponed until later in the proof.

Again, we may note that the conditions under which the program computes the required result arose in the proof. In this example,

these are not just the termination conditions, but the conditions that guarantee that the required function is computed. These arose partially from our insistence that the inner loop computed $A_i.B'_j$—i.e., to obtain these correctly, we had to know what intermediate results were required. However, had we chosen wrongly here, the overall proof would not have succeeded, so that we have a check that the choice was in fact correct.

REFERENCES

1. McCarthy, J., "Towards a mathematical science of computation," *Proceedings I.F.I.P. Congress,* 1962, North-Holland, Amsterdam.
2. Floyd, R. W., "Assigning meaning to programs," *Proc. Symposium on Applied Maths., American Math. Soc.* **19** (1967).
3. Mendelson, E., *Introduction to mathematical logic,* D. Van Nostrand, Princeton (1963).
4. Kleene, S. C., *Introduction to metamathematics,* North-Holland, Amsterdam (1952).
5. Manna, Z., "Termination of algorithms," Thesis, Carnegie Mellon University (April 1968).
6. Manna, Z., "The correctness of programs," *Journal of Computer and Systems Sciences,* No. 3 (1969).
7. Manna, Z., "Properties of programs and the first-order predicate calculus," *Journal of the Association for Computing Machinery,* **16,** No. 2, April, 1969.
8. Manna, Z., Pneuli, A., "Formalization of properties of recursively defined functions," *A.C.M. Symposium on Theory of Computation,* Marina del Rey, California, May 1969.
9. Ashcroft, E. A., "Functional programs as axiomatic theories," Report No. 9, Centre for Computing and Automation, Imperial College, London.
10. Ashcroft, E. A., "Mathematical logic applied to the semantics of computer programs," thesis submitted to Imperial College.
11. Although "$x^n + y^n = z^n$ has no solution in integers unless $n = 2$" is a proposition, whether it is true or false is not known.
12. We generally use capital letters to stand for sets.

GENERAL REFERENCES

1. Cooper, D. C., "Program schema equivalences and second order logic," *Machine Intelligence* **4,** Edinburgh University Press, Edinburgh (1969).
2. Park, D., "Fixpoint induction and proofs of program properties," *Machine Intelligence* **5,** Edinburgh University Press, Edinburgh (1970).