Formalization of Properties of Functional Programs

ZOHAR MANNA AND AMIR PNUELI*

Stanford University,† Stanford, California

ABSTRACT. The problems of convergence, correctness, and equivalence of computer programs can be formulated by means of the satisfiability or validity of certain first-order formulas. An algorithm is presented for constructing such formulas for functional programs, i.e. programs defined by LISP-like conditional recursive expressions.

KEY WORDS AND PHRASES: convergence, correctness, equivalence, functional programs, recursive definitions, first-order logic, satisfiability, validity

CR CATEGORIES: 5.21, 5.23, 5.24

1. *Introduction*

Providing methods for proving properties of a given computer program is one of the very first steps toward transforming the art of programming into a science. We would like to be able to prove properties, such as convergence (termination), correctness, and equivalence, in a formal and rigorous manner. One natural approach is to transform these problems to equivalent problems in some well-developed field of mathematics, so that the transformation permits mathematical techniques and results to be applied for solving the original problems. We chose the *predicate calculus* as such a field.

Once the programming language's syntax and the notion of its execution (its semantics) are formally defined, we try to find an algorithm which will construct for a given program P (in the given language) a first-order formula W_P characterizing its execution. Then, for example, the problem of proving the convergence and correctness (with respect to a given assertion) of the program can be reduced to the problem of proving the validity of a formula constructed by the use of W_P.

This approach has been used already by Floyd [3] and Manna [4, 5] for flowchart programs, i.e. for programs that can be expressed by simple flowcharts. In this paper we extend the techniques to functional programs, i.e. LISP-like programs which are described by a system of recursively defined functions (see McCarthy [9]). This enables us to study the effects of adding to our programming languages features of recursiveness, and in particular procedure calls, which were absent in the simple flowchart programs. Since any flowchart program can be expressed as a functional program (see McCarthy [8]), the results of this paper actually include the previous results.

For simplicity, we shall restrict ourselves to functional programs which are defined

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183). The preliminary version of this paper was presented at the ACM Symposium on the Theory of Computing, Marina Del Rey, California, May 1969.

* Present Address: Weizmann Institute of Science, Rehovoth, Israel.

† Computer Science Department.

by a single recursive definition. Extension of the presented results to the general case is straightforward and is described briefly in Section 9.

To minimize the preliminary definitions, we make use of standard conventions regarding first-order theories (see, for example, Mendelson [10] or Manna [4]).

2. *Conditional Terms*

Let N be a fixed nonnegative integer. The symbols from which our conditional terms are constructed are:

(1) Variables:
  (a) individual variables x_i $(1 \leq i \leq N)$, and
  (b) a function variable $\mathfrak{F}$ of N arguments.

(2) Constants:
  (a) individual constants a_i $(i \geq 1)$,
  (b) function constants f_i^n $(i, n \geq 1)$ of n arguments, and
  (c) predicate constants p_i^n $(i, n \geq 1)$ of n arguments.

A *conditional term* $\tau(\bar{x}, \mathfrak{F})$ (τ, for short) is defined recursively as:

(1) Each individual variable x_i $(1 \leq i \leq N)$ and each individual constant a_i is a conditional term;

(2) If $\tau_1, \cdots, \tau_n$, $n \geq 1$, are conditional terms, so is $f_i^n(\tau_1, \cdots, \tau_n)$, where f_i^n is a function constant of n arguments;

(3) If $\tau_1, \cdots, \tau_N$ are conditional terms, so is $\mathfrak{F}(\tau_1, \cdots, \tau_N)$;

(4) If π is a propositional term (see below), and τ_1 and τ_2 are conditional terms, then

$$(\textbf{if } \pi \textbf{ then } \tau_1 \textbf{ else } \tau_2)$$

is a conditional term;

(5) The conditional terms consist exactly of all finite expressions generated by (1), (2), (3), and (4).

A conditional term with no occurrences of the **if-then-else** connective is called a *simple conditional term.*

A *propositional term* $\pi(\bar{x}, \mathfrak{F})$ (π, for short) is defined recursively as:

(1) T (true) and F (false) are propositional terms;

(2) If $\tau_1, \cdots, \tau_n$, $n \geq 1$, are conditional terms then $p_i^n(\tau_1, \cdots, \tau_n)$ is a propositional term, where p_i^n is a predicate constant of n arguments;

(3) If π_1, π_2, and π_3 are propositional terms, so is

$$(\textbf{if } \pi_1 \textbf{ then } \pi_2 \textbf{ else } \pi_3);$$

(4) The propositional terms consist exactly of all finite expressions generated by (1), (2), and (3).

A propositional term with no occurrences of the **if-then-else** connective is called a *simple propositional term.*

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

Example. Let $N = 1$.

$$(\textbf{if } p_1(x) \textbf{ then } a_1 \textbf{ else } \mathfrak{F}(\textbf{if } p_2(x) \textbf{ then } f_1(x) \textbf{ else } f_2(x)))$$

is a conditional term, while

$$(\textbf{if } p_1(\mathfrak{F}(\textbf{if } p_2(x) \textbf{ then } a_1 \textbf{ else } f_1(x))) \textbf{ then } p_3(x)$$
$$\textbf{else } p_3(\textbf{if } p_2(\mathfrak{F}(x)) \textbf{ then } f_2(x) \textbf{ else } f_3(x)))$$

is a propositional term.

An *interpretation $\mathcal{I}$ of a conditional term τ* consists of a nonempty domain D and the following assignments to the constants of τ:

(a) To each individual constant a_i which occurs in τ, we assign some fixed element of D;

(b) To each function constant f_i^n which occurs in τ, we assign a total function mapping D^n into D;

(c) To each predicate constant p_i^n which occurs in τ, we assign a total predicate over D^n (i.e. a total function mapping D^n into {T, F}).

Note that $\mathcal{I}$ does not include assignments to the variables of τ.

3. *Functional Programs*

A *functional program*[1] $\mathcal{P}$ is an ordered pair $(P, \mathcal{I})$, where

(a) P (called a *functional program schema*) is a recursive definition of the form $\mathfrak{F}(\bar{x}) \Leftarrow \tau(\bar{x}, \mathfrak{F})$, where $\tau(\bar{x}, \mathfrak{F})$ is a conditional term, and

(b) $\mathcal{I}$ is an interpretation of τ.

Example. The functional program schema P,

$$\mathfrak{F}(\bar{x}) \Leftarrow \textbf{if } p(x) \textbf{ then } a \textbf{ else } f_2(x, \mathfrak{F}(f_1(x))),$$

defines different functional programs for different interpretations. For example:

1. The *factorial program* (for computing $x!$), $\mathcal{P}_1 = (P, \mathcal{I}_1)$, where $\mathcal{I}_1$ is: $D =$ {the nonnegative integers}, $p(x)$ is $x = 0$, a is 1, $f_2(x_1, x_2)$ is $x_1 \cdot x_2$, and $f_1(x)$ is $x - 1$.

2. The *summation program* (for computing $1 + 2 + \cdots + x$), $\mathcal{P}_2 = (P, \mathcal{I}_2)$, where $\mathcal{I}_2$ is: $D =$ {the nonnegative integers}, $p(x)$ is $x = 0$, a is 0, $f_2(x_1, x_2)$ is $x_1 + x_2$, and $f_1(x)$ is $x - 1$.

3. The *reverse program* (for reversing the order of top level elements in a given list x), $\mathcal{P}_3 = (P, \mathcal{I}_3)$, where $\mathcal{I}_3$ is: $D =$ {all lists}, $p(x)$ is $\textbf{null}(x)$, a is NIL, $f_2(x_1, x_2)$ is $\textbf{cons}(\textbf{car}(x_1), x_2)$, and $f_1(x)$ is $\textbf{cdr}(x)$.

4. The *sort program* (for sorting a list x of integers), $\mathcal{P}_4 = (P, \mathcal{I}_4)$, where $\mathcal{I}_4$ is: $D =$ {lists of integers}, $p(x)$ is $\textbf{null}(x)$, a is NIL, $f_2(x_1, x_2)$ is $\textbf{merge}(\textbf{car}(x_1), x_2)$, and $f_1(x)$ is $\textbf{cdr}(x)$. Here the "primitive" function $\textbf{merge}(y_1, y_2)$ merges an integer y_1 into a sorted list of integers y_2.

$\mathcal{P}$ defines $\mathfrak{F}$ as a partial function mapping D^N into D as follows.

For every $\bar{\xi} \in D^N$, one can generate a sequence of conditional terms, $\tau_0 \rightarrow \tau_1 \rightarrow \tau_2 \rightarrow \cdots$, called the *computation sequence of* $\mathfrak{F}(\bar{\xi})$, according to the following computational rules:

(1) τ_0 is $\tau(\bar{x}, \mathfrak{F})$, after all possible simplifications (see below);

(2) τ_{i+1}, $i \geq 0$, is obtained from τ_i by replacing in τ_i the *leftmost* occurrence of

[1] Restricted to a single recursive definition (see Introduction).

$\mathfrak{F}$ with arguments free of $\mathfrak{F}$, $\mathfrak{F}(\bar{t})$ say, by $\tau(\bar{t}, \mathfrak{F})$, and then applying all possible simplifications (see below);

(3) The sequence is finite and τ_k is the final conditional term in the sequence if and only if τ_k does not contain any occurrence of $\mathfrak{F}$. Thus, τ_k must be (see the simplification rules below) an individual constant which denotes therefore by $\mathcal{I}$ a unique element of the domain D, say η. In this case we say that $\mathfrak{F}(\bar{\xi})$ *converges to* η or that $\mathfrak{F}(\bar{\xi})$ *is defined* (notation: $*\mathfrak{F}(\bar{\xi})$) *and* $\mathfrak{F}(\bar{\xi}) = \eta$. Otherwise, i.e. if the sequence is infinite, we say that $\mathfrak{F}(\bar{\xi})$ *is undefined.*

The simplification rules are:

1. Using the assignment of $\bar{\xi}$ to $\bar{x}$ and the interpretation $\mathcal{I}$:

(a) Replace each occurrence of $f_i^n(\tau_1, \cdots, \tau_n)$, where $\tau_1, \cdots, \tau_n$ are simple conditional terms that do not contain occurrences of $\mathfrak{F}$, by an individual constant representing its value.[2]

(b) Replace each occurrence of $p_i^n(\tau_1, \cdots, \tau_n)$, where $\tau_1, \cdots, \tau_n$ are simple conditional terms that do not contain occurrences of $\mathfrak{F}$, by its value (i.e. T or F).

2. Replace "**if** T **then** A **else** B" by "A" and "**if** F **then** A **else** B" by "B". Note that B is not computed in the first case, and A is not computed in the second case.

Example. In the sequel we shall refer to the functional program $\hat{\mathcal{P}}$ which is represented by (see Manna and McCarthy [6]):

$$\hat{\mathfrak{F}}(x) \Leftarrow \textbf{if } x > 100 \textbf{ then } x - 10 \textbf{ else } \hat{\mathfrak{F}}(\hat{\mathfrak{F}}(x + 11))$$

where $D = I$ (the integers).

The computation sequence of $\hat{\mathfrak{F}}(99)$ can be represented as[3]

$$\hat{\mathfrak{F}}(\hat{\mathfrak{F}}(110)) \to \hat{\mathfrak{F}}(100) \to \hat{\mathfrak{F}}(\hat{\mathfrak{F}}(111)) \to \hat{\mathfrak{F}}(101) \to 91.$$

Therefore, $\hat{\mathfrak{F}}(99)$ is defined and $\hat{\mathfrak{F}}(99) = 91$.

Note that for the functional program $\mathcal{P}$ represented by

$$\mathfrak{F}(x) \Leftarrow \textbf{if } x > 100 \textbf{ then } x - 11 \textbf{ else } \mathfrak{F}(\mathfrak{F}(x + 11)),$$

$\mathfrak{F}(99)$ is undefined, since the computation sequence of $\mathfrak{F}(99)$ represented by

$$\mathfrak{F}(\mathfrak{F}(110)) \to \mathfrak{F}(99) \to \mathfrak{F}(\mathfrak{F}(110)) \to \mathfrak{F}(99) \to \cdots$$

is infinite.

In defining propositional terms we have restricted ourselves to one connective, the ternary **if-then-else** connective, whose truth values are defined by:

$$\textbf{if } A \textbf{ then } B \textbf{ else } C \quad \text{is} \quad \begin{cases} B & \text{if } A \text{ is T} \\ C & \text{if } A \text{ is F.} \end{cases}$$

[2] This may require extending the interpretation $\mathcal{I}$ to include assignments for new individual constants.

[3] Here the substitution and simplifications involved, for example in getting τ_0, are:

if 99 > 100 **then** 99 − 10 **else** $\hat{\mathfrak{F}}(\hat{\mathfrak{F}}(99 + 11)) \to$
if F **then** 89 **else** $\hat{\mathfrak{F}}(\hat{\mathfrak{F}}(110)) \to \hat{\mathfrak{F}}(\hat{\mathfrak{F}}(110))$.

However, this is not a real restriction, since every connective of the propositional calculus can be expressed by the **if-then-else** connective together with the constants T and F (see Church [1, Sec. 24] and McCarthy [9]). In particular,

$\sim A$ is **if** A **then** F **else** T,

$A \supset B$ is **if** A **then** B **else** T,

$A \vee B$ is **if** A **then** T **else** B,

$A \wedge B$ is **if** A **then** B **else** F, and

$A \equiv B$ is **if** A **then** B **else** (**if** B **then** F **else** T).

4. *Well-Formed Formulas*

The symbols from which our well-formed formulas are constructed are:

(1) Variables

(a) individual variables x_i, z_i $(i \geq 1)$, and

(b) a predicate variable Q of $N + 1$ arguments.

(2) Constants—the same as for conditional terms (see Section 2).

A *term* t is defined recursively as:

(1) Each individual variable and each individual constant a_i is a term;

(2) If $t_1, \cdots, t_n$ $(n \geq 1)$ are terms, so is $f_i^n(t_1, \cdots, t_n)$, where f_i^n is a function constant of n arguments;

(3) The terms consist exactly of all finite expressions generated by (1) and (2).

A *well-formed formula* (wff) is defined recursively as:

(1) T and F are wffs;

(2) If $t_1, \cdots, t_n$ $(n \geq 1)$ are terms, then $p_i^n(t_1, \cdots, t_n)$ is a wff, where p_i^n is a predicate constant of n arguments;

(3) If $t_1, \cdots, t_{N+1}$ are terms, then $Q(t_1, \cdots, t_{N+1})$ is a wff;

(4) If W_1 and W_2 are wffs, so are $\sim W_1$, $W_1 \wedge W_2$, $W_1 \vee W_2$, $W_1 \supset W_2$, and $(\forall z)W_1$ where z is any free individual variable in W_1;

(5) The wffs consist exactly of all finite expressions generated by (1), (2), (3), and (4).

The ternary connective IF W_1 THEN W_2 ELSE W_3 will be used as an abbreviation for $(W_1 \supset W_2) \wedge (\sim W_1 \supset W_3)$.[4] We write $W(\bar{x}, Q)$ to indicate that there are no free individual variables in W other than $\bar{x}$, and to emphasize that W may contain the predicate variable Q.

Since we have used the same constant symbols for the construction of both conditional terms and wffs, an *interpretation* $\mathcal{I}$ *of a wff* W is defined exactly as for conditional terms. Note again that $\mathcal{I}$ does not include assignments to the variables of W. The ordered pair $\mathfrak{W} = (W, \mathcal{I})$ is called an *interpreted wff*.

Let $\mathfrak{W} = (W, \mathcal{I})$ be an interpreted wff, where W is a wff *with no free individual*

[4] The reader should be aware that we are using the three words "if-then-else" for three different meanings:

(a) if ··· then ··· else in the metalanguage,

(b) **if** ··· **then** ··· **else** in the definition of conditional terms, and

(c) IF ··· THEN ··· ELSE here in the definition of wffs.

variables, and $\mathcal{I}$ is an interpretation of W. Let D be the domain of $\mathcal{I}$. By assigning, in addition, a total predicate over D^{N+1} to the predicate variable Q, $\mathfrak{W}$ has the value T or F. This value is obtained by interpreting T as true and F as false, using the usual truth values of $\sim$, $\wedge$, $\vee$, and $\supset$, and interpreting the universally quantified variables in the standard way.

$\mathfrak{W}$ is said to be:

(a) *satisfiable*, if we can assign to Q a total predicate over D^{N+1} such that the value of $\mathfrak{W}$ is true;

(b) *unsatisfiable*, if it is not satisfiable;

(c) *valid*, if for every assignment of a total predicate over D^{N+1} to Q, the value of $\mathfrak{W}$ is true.

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

W is said to be:

(a) *satisfiable*, if there exists an interpretation $\mathcal{I}$ of W such that $(W, \mathcal{I})$ is satisfiable;

(b) *unsatisfiable*, if it is not satisfiable;

(c) *valid*, if for every interpretation $\mathcal{I}$ of W, $(W, \mathcal{I})$ is valid.

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

5. *The Algorithm to Construct* $\mathfrak{W}_{\mathcal{P}}$

Let $\mathcal{P} = (P, \mathcal{I})$ be a functional program, where P is the recursive definition $\mathfrak{F}(\bar{x}) \Leftarrow \tau(\bar{x}, \mathfrak{F})$, and $\mathcal{I}$ is an interpretation of τ.

In this section we describe an algorithm to construct from $\mathcal{P} = (P, \mathcal{I})$ an interpreted wff $\mathfrak{W}_{\mathcal{P}} = (W_P, \mathcal{I})$, characterizing $\mathcal{P}$. The function variable $\mathfrak{F}$ itself is represented in W_P by the predicate variable Q in the following *intuitive* sense: For every $\bar{x}$ and y, $y = \mathfrak{F}(\bar{x})$ implies $Q(\bar{x}, y) = \mathrm{T}$ (but not *necessarily* conversely!).

In order to gain clarity, we first illustrate the interpreted wffs $\mathfrak{W}_{\mathcal{P}} = (W_P, \mathcal{I})$ for several simple functional programs.

1. The "91 function":

$\mathcal{P}$: $\mathfrak{F}(x) \Leftarrow$ **if** $x > 100$ **then** $x - 10$ **else** $\mathfrak{F}(\mathfrak{F}(x + 11))$,
$D = \{\text{the integers}\}$.

$\mathfrak{W}_{\mathcal{P}}$: $\forall x\{\mathrm{IF}\ x > 100\ \mathrm{THEN}\ Q(x, x - 10)$
$\mathrm{ELSE}\ \forall z_1 \forall z_2[Q(x + 11, z_1) \wedge Q(z_1, z_2) \supset Q(x, z_2)]\}$.

Roughly speaking, the meaning of $\mathfrak{W}_{\mathcal{P}}$ is: "For every integer x, if $x > 100$ then $\mathfrak{F}(x) = x - 10$; else for every integer z_1 and for every integer z_2, if $\mathfrak{F}(x + 11) = z_1$ and $\mathfrak{F}(z_1) = z_2$ then $\mathfrak{F}(x) = z_2$."

Note that equivalently we could construct the interpreted wff

$\forall x \forall y\{\mathrm{IF}\ x > 100\ \mathrm{THEN}\ y = x - 10$
$\mathrm{ELSE}\ \forall z_1 \forall z_2[Q(x + 11, z_1) \wedge Q(z_1, z_2) \supset y = z_2]$
$\supset Q(x, y)\}$,

but we prefer the first formulation which does not use equality relations not occurring in the program itself.

2. The factorial function:

$\mathcal{P}$: $\mathfrak{F}(x) \Leftarrow$ **if** $x = 0$ **then** 1 **else** $x \cdot \mathfrak{F}(x - 1)$,
$D = \{\text{the nonnegative integers}\}$.

$\mathfrak{W}_\mathfrak{P}$: $\forall x\{\text{IF } x = 0 \text{ THEN } Q(x, 1)$
$\text{ELSE } \forall z[Q(x - 1, z) \supset Q(x, x \cdot z)]\}$.

3. Ackermann's function:

$\mathfrak{P}$: $\mathfrak{F}(x_1, x_2) \Leftarrow \textbf{if } x_1 = 0 \textbf{ then } x_2 + 1$
$\textbf{else if } x_2 = 0 \textbf{ then } \mathfrak{F}(x_1 - 1, 1)$
$\textbf{else } \mathfrak{F}(x_1 - 1, \mathfrak{F}(x_1, x_2 - 1))$,

D = {the nonnegative integers}.

$\mathfrak{W}_\mathfrak{P}$: $\forall x_1 \forall x_2\{\text{IF } x_1 = 0 \text{ THEN } Q(x_1, x_2, x_2 + 1)$
$\text{ELSE IF } x_2 = 0 \text{ THEN } \forall z_1[Q(x_1 - 1, 1, z_1) \supset Q(x_1, x_2, z_1)]$
$\text{ELSE } \forall z_2 \forall z_3[Q(x_1, x_2 - 1, z_2) \wedge Q(x_1 - 1, z_2, z_3) \supset Q(x_1, x_2, z_3)]\}$.

The reader should be aware that the formula must be constructed with special care when the function variable $\mathfrak{F}$ occurs in a propositional term. For example:

4. The remainder function:

$\mathfrak{P}$: $\mathfrak{F}(x_1, x_2) \Leftarrow \textbf{if } x_1 = 0 \textbf{ then } 0$
$\textbf{else if } \mathfrak{F}(x_1 - 1, x_2) + 1 = x_2 \textbf{ then } 0$
$\textbf{else } \mathfrak{F}(x_1 - 1, x_2) + 1$,

D = {the nonnegative integers}.

$\mathfrak{W}_\mathfrak{P}$: $\forall x_1 \forall x_2\{\text{IF } x_1 = 0 \text{ THEN } Q(x_1, x_2, 0) \text{ ELSE}$
$\forall z_1\{Q(x_1 - 1, x_2, z_1) \supset \text{IF } z_1 + 1 = x_2 \text{ THEN } Q(x_1, x_2, 0) \text{ ELSE}$
$\forall z_2[Q(x_1 - 1, x_2, z_2) \supset Q(x_1, x_2, z_2 + 1)]\}\}$.

We are now ready to proceed with the detailed algorithm. The algorithm replaces successively $\mathfrak{F}$'s by Q's in τ, obtaining finally the wff W_P (with no free individual variables).

(1) For a *simple conditional term* $\tau(\bar{x}, \mathfrak{F})$ the wff $W_\tau(\bar{x}, Q)$ is defined, by means of an (antecedent) wff R_τ and a (consequent) term $\tilde{\tau}$, as follows:

(a) if τ is an individual variable or an individual constant, then define
R_τ as T and $\tilde{\tau}$ as τ;

(b) if τ is of the form $f_i^n(\tau_1, \cdots, \tau_n)$, then define
R_τ as $\wedge_{i=1}^N R_{\tau_i}$ and $\tilde{\tau}$ as $f_i^n(\tilde{\tau}_1, \cdots, \tilde{\tau}_n)$;

(c) if τ is of the form $\mathfrak{F}(\tau_1, \cdots, \tau_N)$, then define
R_τ as $(\wedge_{i=1}^N R_{\tau_i}) \wedge Q(\tilde{\tau}_1, \cdots, \tilde{\tau}_N, z)$ and $\tilde{\tau}$ as z,
where z is any new individual variable that has not been used before in the algorithm.

Now, for every simple conditional term $\tau(\bar{x}, \mathfrak{F})$, we can define $W_\tau(\bar{x}, Q)$ as
$\forall z_1 \cdots \forall z_k[R_\tau \supset Q(\bar{x}, \tilde{\tau})]$, where $z_1, \cdots, z_k$ are all individual variables introduced in step (c).

Example. If $\tau(x, \mathfrak{F})$ is of the form $f_1(\mathfrak{F}(f_2(x)), f_3(\mathfrak{F}(f_4(x))))$, then $W_\tau(x, Q)$ is
$\forall z_1 \forall z_2\{[Q(f_2(x), z_1) \wedge Q(f_4(x), z_2)] \supset Q(x, f_1(z_1, f_3(z_2)))\}$.

(2) For a *simple propositional term* π, the (antecedent) wff S_π and the (consequent) wff $\tilde{\pi}$ are defined as follows:

(a) if π is T or F, then define
S_π as T and $\tilde{\pi}$ as π;

(b) if π is of the form $p_i^n(\tau_1, \cdots, \tau_n)$, then define
S_π as $\wedge_{i=1}^n R_{\tau_i}$ and $\tilde{\pi}$ as $p_i^n(\tilde{\tau}_1, \cdots, \tilde{\tau}_n)$.

(3) Let $\tau(\bar{x}, \mathfrak{F})$ be any *nonsimple conditional term*, and let ρ be the *leftmost* part of τ of the form

$$\textbf{if } \pi \textbf{ then } \rho_1 \textbf{ else } \rho_2$$

where π is a *simple* propositional term, and ρ_1 and ρ_2 are either both condition terms or both propositional terms.

Let us write τ as τ_ρ to display explicitly ρ's occurrence in τ. So, τ_{ρ_i} is the result of replacing ρ by ρ_i in τ. Define $W_\tau(\bar{x}, Q)$ as

$$\forall z_1 \cdots \forall z_k[S_\pi \supset \text{IF } \bar{\pi} \text{ THEN } W_{\tau_{\rho_1}} \text{ ELSE } W_{\tau_{\rho_2}}]$$

where $z_1, \cdots, z_k$ are all the individual variables introduced in step (1-c) in the construction of S_π and $\bar{\pi}$.

Example. Let $\tau(x, \mathfrak{F})$ be of the form

$$\textbf{if } p_1(x) \textbf{ then } \mathfrak{F}\ (\textbf{if } p_2(\mathfrak{F}(x)) \textbf{ then } f(x) \textbf{ else } a) \textbf{ else } a$$

A detailed derivation of $W_\tau(x, Q)$ is[5]:

(a) $\mathrm{T} \supset \text{IF } p_1(x) \text{ THEN } W_{\mathfrak{F}(\textbf{if } p_2(\mathfrak{F}(x)) \textbf{ then } f(x) \textbf{ else } a)}$
ELSE W_a

(b) IF $p_1(x)$ THEN $\forall z\{Q(x, z) \supset \text{IF } p_2(z) \text{ THEN } W_{\mathfrak{F}(f(x))} \text{ ELSE } W_{\mathfrak{F}(a)}\}$
ELSE $Q(x, a)$

(c) IF $p_1(x)$ THEN $\forall z\{Q(x, z) \supset \text{IF } p_2(z) \text{ THEN } \forall z_1[Q(f(x), z_1) \supset Q(x, z_1)$
ELSE $\forall z_2[Q(a, z_2) \supset Q(x, z_2)]\}$
ELSE $Q(x, a)$.

(4) Define W_P as $\forall \bar{x} W_\tau(\bar{x}, Q)$, and $\mathfrak{W}_\mathcal{P}$ as $(W_P, \mathcal{I})$.

To emphasize that $\mathfrak{W}_\mathcal{P}$ still contains the uninterpreted predicate variable Q, we shall usually denote $\mathfrak{W}_\mathcal{P}$ by $\mathfrak{W}_\mathcal{P}(Q)$. Then $\mathfrak{W}_\mathcal{P}(\delta)$ stands for the truth value obtained by the assignment of the predicate δ to Q in W_P, in addition to the assignments of $\mathcal{I}$ to the constants of W_P.

6. *Lemmas*

The following definitions and lemmas are used in the proofs of the theorems presented in Sections 7 and 8.

Let $\mathcal{P}$ be a functional program, $\mathfrak{W}_\mathcal{P}$ its corresponding interpreted wff, and $\mathfrak{F}(\bar{x})$ the partial function (mapping D^N into D) defined by $\mathcal{P}$. Let $\delta(\bar{x}, y)$ be a total predicate over D^{N+1}.

Definition 1. $\delta(\bar{x}, y)$ is called *a valid predicate for* $\mathcal{P}$ if for every $(\bar{\xi}, \eta) \in D^{N+1}$ if $[*\mathfrak{F}(\bar{\xi})$ and $\eta = \mathfrak{F}(\bar{\xi})]$ then $\delta(\bar{\xi}, \eta)$.

Definition 2. $\delta(\bar{x}, y)$ is called *the minimal valid predicate for* $\mathcal{P}$ if for every $(\bar{\xi}, \eta) \in D^{N+1}$: $[*\mathfrak{F}(\bar{\xi})$ and $\eta = \mathfrak{F}(\bar{\xi})]$ if and only if $\delta(\bar{\xi}, \eta)$.

LEMMA 1. *If δ is the minimal valid predicate for $\mathcal{P}$, then $\mathfrak{W}_\mathcal{P}(\delta)$ is true.*

LEMMA 2. *If $\mathfrak{W}_\mathcal{P}(\delta)$ is true, then δ is a valid predicate for $\mathcal{P}$.*

We will not present a detailed proof of these lemmas here.

To prove Lemma 1 consider first the case that P is defined by $\mathfrak{F}(\bar{x}) \Leftarrow \tau(\bar{x}, \mathfrak{F})$, where τ is a simple conditional term. Then using the fact that δ is the minimal valid predicate for $\mathcal{P}$, show that for every $\bar{\xi} \in D^N$, $\mathfrak{W}_\tau(\bar{\xi}, \delta)$ is true. An inductive argument

[5] Note that W_τ can usually be simplified, since for every wff A, $T \wedge A$ and $T \supset A$ are logically equivalent to A.

on the number of the **if-then-else** connectives will extend the result to the case of τ being a general conditional term.

To prove Lemma 2, consider any $\bar{\xi} \in D^N$ for which $\mathfrak{F}(\bar{\xi})$ is defined. (Suppose $\mathfrak{F}(\bar{\xi}) = \eta \in D$.) Let $\tau_0 \to \tau_1 \to \tau_2 \to \cdots \to \tau_k$ be the computation sequence of $\mathfrak{F}(\bar{\xi})$. By an inductive argument on the length of the computation sequence of $\mathfrak{F}(\bar{\xi})$, one can verify that for every j, $0 \leq j < k$, there exists an m, $j < m \leq k$, such that if $\mathfrak{W}_{\tau_j}(\bar{\xi}, \delta)$ is true then $\mathfrak{W}_{\tau_m}(\bar{\xi}, \delta)$ is also true. Therefore, since it is given that $\mathfrak{W}_{\tau_0}(\bar{\xi}, \delta)$ is true, it follows that $\mathfrak{W}_{\tau_k}(\bar{\xi}, \delta)$, which is $\delta(\bar{\xi}, \eta)$, is true.

Let us consider for illustration the computation sequence of $\hat{\mathfrak{F}}(99)$ (see Section 3), which is:

$$\hat{\mathfrak{F}}(\hat{\mathfrak{F}}(110)) \to \hat{\mathfrak{F}}(100) \to \hat{\mathfrak{F}}(\hat{\mathfrak{F}}(111)) \to \hat{\mathfrak{F}}(101) \to 91.$$

It is given that $\mathfrak{W}_{\hat{\mathcal{F}}}(\delta)$, i.e.

(a) $\forall x\{\text{IF } x > 100 \text{ THEN } \delta(x, x - 10)$
$\text{ELSE } \forall z_1 \forall z_2[\delta(x + 11, z_1) \wedge \delta(z_1, z_2) \supset \delta(x, z_2)]\}$

is true.

The sequence of wffs corresponding to the computation of $\hat{\mathfrak{F}}(99)$ is: Substitute $x = 99$ in (a), to get

(b) $\forall z_1 \forall z_2[\delta(110, z_1) \wedge \delta(z_1, z_2) \supset \delta(99, z_2)]$,

which is $\mathfrak{W}_{\hat{\mathfrak{F}}(\hat{\mathfrak{F}}(110))}(99, \delta)$.

Substitute $x = 110$ in (a) to get $\delta(110, 100)$, and substitute $z_1 = 100$ in (b) to get

(c) $\forall z_2[\delta(100, z_2) \supset \delta(99, z_2)]$,

which is $\mathfrak{W}_{\hat{\mathfrak{F}}(100)}(99, \delta)$.

Substitute $x = 100$ in (a) to get

(d) $\forall z_1 \forall z_2[\delta(111, z_1) \wedge \delta(z_1, z_2) \supset \delta(100, z_2)]$.

Combine (d) with (c) to get

(e) $\forall z_1 \forall z_2[\delta(111, z_1) \wedge \delta(z_1, z_2) \supset \delta(99, z_2)]$,

which is $\mathfrak{W}_{\hat{\mathfrak{F}}(\hat{\mathfrak{F}}(111))}(99, \delta)$.

Substitute $x = 111$ in (a) to get $\delta(111, 101)$, and substitute $z_1 = 101$ in (e) to get

(f) $\forall z_2[\delta(101, z_2) \supset \delta(99, z_2)]$,

which is $\mathfrak{W}_{\hat{\mathfrak{F}}(101)}(99, \delta)$.

Substitute $x = 101$ in (a) to get $\delta(101, 91)$, and substitute $z_2 = 91$ in (f) to get

(g) $\delta(99, 91)$,

which is $\mathfrak{W}_{91}(99, \delta)$.

7. *Correctness*

Given:

(a) a functional program $\mathcal{P} = (P, \mathcal{I})$, where D is the domain of $\mathcal{I}$ and $\mathfrak{F}(\bar{x})$ is the partial function (mapping D^N into D) defined by $\mathcal{P}$;

(b) a total predicate $\varphi(\bar{x})$ over D^N (called the *input predicate*); and

(c) a total predicate $\psi(\bar{x}, y)$ over D^{N+1} (called the *output predicate*).

We say that

1. *$\mathcal{P}$ is partially correct with respect to φ and ψ* if for every $\bar{\xi}$, such that $\varphi(\bar{\xi})$, if $*\mathfrak{F}(\bar{\xi})$ then $\psi(\bar{\xi}, \mathfrak{F}(\bar{\xi}))$.

2. *$\mathcal{P}$ is correct with respect to φ and ψ* if for every $\bar{\xi}$, such that $\varphi(\bar{\xi})$, $*\mathfrak{F}(\bar{\xi})$ and $\psi(\bar{\xi}, \mathfrak{F}(\bar{\xi}))$.

Let Φ be a predicate constant of N arguments and let Ψ be a predicate constant of $N+1$ arguments. Let $\mathcal{I}^+$ stand for the interpretation of P, Φ, and Ψ, i.e. $\mathcal{I}^+$ consists of the assignments of $\mathcal{I}$ to the constants of P, φ to Φ, and ψ to Ψ.

We shall make use of the following interpreted wffs:

1. $[\mathcal{P}, \varphi, \psi] = ([P, \Phi, \Psi], \mathcal{I}^+)$ where the wff $[P, \Phi, \Psi]$ is
$$W_P(Q) \wedge \forall\bar{x}\forall y\{[\Phi(\bar{x}) \wedge Q(\bar{x}, y)] \supset \Psi(\bar{x}, y)\}, \text{ and}$$
2. $\{\mathcal{P}, \varphi, \psi\} = (\{P, \Phi, \Psi\}, \mathcal{I}^+)$ where the wff $\{P, \Phi, \Psi\}$ is
$$W_P(Q) \supset \forall\bar{x}\{\Phi(\bar{x}) \supset \exists y[Q(\bar{x}, y) \wedge \Psi(\bar{x}, y)]\}.$$

THEOREM 1. *$\mathcal{P}$ is partially correct with respect to φ and ψ if and only if $[\mathcal{P}, \varphi, \psi]$ is satisfiable.*

PROOF. $\Rightarrow$ *Satisfiability.* Let δ be the minimal valid predicate for $\mathcal{P}$. By Lemma 1, $\mathfrak{W}_{\mathcal{P}}(\delta)$ is true. Since δ is the minimal valid predicate for $\mathcal{P}$, i.e.

$$\forall(\bar{\xi}, \eta) \in D^{N+1}\text{: if } \delta(\bar{\xi}, \eta) \text{ then } [*\mathfrak{F}(\bar{\xi}) \text{ and } \eta = \mathfrak{F}(\bar{\xi})],$$

and since $\mathcal{P}$ is partially correct with respect to φ and ψ, i.e.

$$\forall\bar{\xi} \in D^N\text{: if } [\varphi(\bar{\xi}) \text{ and } *\mathfrak{F}(\bar{\xi})] \text{ then } \psi(\bar{\xi}, \mathfrak{F}(\bar{\xi})),$$

it follows that

$$\forall\bar{x}\forall y\{[\varphi(\bar{x}) \wedge \delta(\bar{x}, y)] \supset \psi(\bar{x}, y)\}$$

is also true.

$\Leftarrow$ *Satisfiability.* Since $\mathfrak{W}_{\mathcal{P}}(\delta)$ is true (for some δ), it follows by Lemma 2 that δ is a valid predicate for $\mathcal{P}$, i.e.

$$\forall\bar{\xi} \in D^N\text{: if } *\mathfrak{F}(\bar{\xi}) \text{ then } \delta(\bar{\xi}, \mathfrak{F}(\bar{\xi})),$$

which implies

$$\forall\bar{\xi} \in D^N\text{: if } [\varphi(\bar{\xi}) \text{ and } *\mathfrak{F}(\bar{\xi})] \text{ then } [\varphi(\bar{\xi}) \text{ and } \delta(\bar{\xi}, \mathfrak{F}(\bar{\xi}))].$$

Therefore, by the second part of the conjunction, we obtain

$$\forall\bar{\xi} \in D^N\text{: if } [\varphi(\bar{\xi}) \text{ and } *\mathfrak{F}(\bar{\xi})] \text{ then } \psi(\bar{\xi}, \mathfrak{F}(\bar{\xi})).$$ Q.E.D.

THEOREM 2. *P is correct with respect to φ and ψ if and only if $\{P, \varphi, \psi\}$ is valid.*

PROOF. $\Rightarrow$ *Validity.* Let δ be any predicate over D^{N+1} for which $\mathfrak{W}_{\mathcal{P}}(\delta)$ is true. Consider an arbitrary $\bar{\xi} \in D^N$ such that $\varphi(\bar{\xi})$. Since $\mathfrak{W}_{\mathcal{P}}(\delta)$ is true it follows by Lemma 2 that δ is a valid predicate for $\mathcal{P}$, i.e.

$$\text{if } *\mathfrak{F}(\bar{\xi}) \text{ then } \delta(\bar{\xi}, \mathfrak{F}(\bar{\xi})).$$

Moreover, since $\mathcal{P}$ is correct with respect to φ and ψ, we have that

$$*\mathfrak{F}(\bar{\xi}) \text{ and } \psi(\bar{\xi}, \mathfrak{F}(\bar{\xi})).$$

Thus $\exists y[\delta, (\bar{\xi}, y) \wedge \psi(\bar{\xi}, y)]$ is true.

$\Leftarrow$ *Validity.* Let δ be the minimal valid predicate for $\mathcal{P}$. By Lemma 1, $\mathfrak{W}_{\mathcal{P}}(\delta)$ is true. Consider an arbitrary $\bar{\xi} \in D^N$ such that $\varphi(\bar{\xi})$. The validity of $\{\mathcal{P}, \varphi, \psi\}$ implies that

$$\exists\eta \in D \text{ such that } \delta(\bar{\xi}, \eta) \text{ and } \psi(\bar{\xi}, \eta).$$

Since δ is the minimal valid predicate for $\mathcal{P}$, i.e.

$$\text{if } \delta(\bar{\xi}, \eta) \text{ then } [*\mathcal{F}(\bar{\xi}) \text{ and } \eta = \mathcal{F}(\bar{\xi})],$$

it follows that

$$*\mathcal{F}(\bar{\xi}) \text{ and } \psi(\bar{\xi}, \mathcal{F}(\bar{\xi})). \qquad \text{Q.E.D.}$$

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

COROLLARY 1. *$\mathcal{P}$ is undefined for φ (i.e. $\forall \bar{\xi} \in D^N$: if $\varphi(\bar{\xi})$ then $\mathcal{F}(\bar{\xi})$ is undefined) if and only if $[\mathcal{P}, \varphi, \mathrm{F}]$ is satisfiable.*

COROLLARY 2. *$\mathcal{P}$ is defined for φ (i.e. $\forall \bar{\xi} \in D^N$: if $\varphi(\bar{\xi})$ then $\mathcal{F}(\bar{\xi})$ is defined) if and only if $\{\mathcal{P}, \varphi, \mathrm{T}\}$ is valid.*

Example. Let us consider the functional program $\hat{P}$, which was defined in Section 3 as:

$$\hat{\mathcal{F}}(x) \Leftarrow \textbf{if } x > 100 \textbf{ then } x - 10 \textbf{ else } \hat{\mathcal{F}}(\hat{\mathcal{F}}(x + 11)),\ D = I \text{ (the integers)},$$

and the predicate $\hat{\psi}$, defined as:

$$\textbf{if } x > 100 \textbf{ then } y = x - 10 \textbf{ else } y = 91.$$

I. Using Theorem 1 we prove that $\hat{\mathcal{P}}$ is partially correct with respect to $\varphi : \mathrm{T}_i$ and $\hat{\psi}$, i.e. for every integer ξ : if $*\hat{\mathcal{F}}(\xi)$ then $\hat{\psi}(\xi, \hat{\mathcal{F}}(\xi))$. Thus, we have to show that $[\hat{\mathcal{P}}, \mathrm{T}, \hat{\psi}]$ is satisfiable.

Let $Q(x, y)$ be $\hat{\psi}(x, y)$. Since the second part of the conjunction, i.e.

$$\forall x \exists y \{Q(x, y) \supset \hat{\psi}(x, y)\},$$

is clearly true, we have only to show that $\mathfrak{W}_{\hat{\mathcal{P}}}(\hat{\psi})$ is true, where $\mathfrak{W}_{\hat{\mathcal{P}}}(\hat{\psi})$ is:

$$\forall x \{\textbf{if } x > 100 \textbf{ then } \hat{\psi}(x, x - 10) \textbf{ else } \forall z_1 \forall z_2 [\hat{\psi}(x + 11, z_1) \wedge \hat{\psi}(z_1, z_2) \supset \hat{\psi}(x, z_2)]\},$$

i.e.

$$\begin{aligned}\forall x \{\textbf{if } x > 100 \textbf{ then } &[\textbf{if } x > 100 \textbf{ then } x - 10 = x - 10 \textbf{ else } x - 10 = 91] \\ \textbf{else } \forall z_1 \forall z_2 &[\textbf{if } x + 11 > 100 \textbf{ then } z_1 = x + 11 - 10 \textbf{ else } z_1 = 91 \\ &\wedge \textbf{ if } z_1 > 100 \textbf{ then } z_2 = z_1 - 10 \textbf{ else } z_2 = 91 \\ &\supset \textbf{ if } x > 100 \textbf{ then } z_2 = x - 10 \textbf{ else } z_2 = 91]\}.\end{aligned}$$

The reader can verify easily that this expression is true by considering four cases: $x > 100$, $x = 100$, $89 < x < 100$, and $x \leq 89$.

II. Using Theorem 2 we prove that $\hat{\mathcal{P}}$ is correct with respect to $\varphi \equiv \mathrm{T}$ and $\hat{\psi}$, i.e. for every integer ξ : $*\hat{\mathcal{F}}(\xi)$ and $\hat{\psi}(\xi, \hat{\mathcal{F}}(\xi))$. Thus we have to show that $\{\hat{\mathcal{P}}, \mathrm{T}, \hat{\psi}\}$, i.e.

$$\mathfrak{W}_{\hat{\mathcal{P}}}(Q) \supset \forall x \exists y [Q(x, y) \wedge \hat{\psi}(x, y)],$$

is valid.

Suppose $\mathfrak{W}_{\hat{\mathcal{P}}}(Q)$ is true.

1. For $x > 100$, choose $y = x - 10$. $\mathfrak{W}_{\hat{\mathcal{P}}}(Q)$ implies $Q(x, x - 10)$ and $\hat{\psi}(x, x - 10)$ is clearly true.

2. For $x \leq 101$, choose $y = 91$. $\hat{\psi}(x, 91)$ is clearly true. We proceed to prove by induction on x that $Q(x, 91)$. For $x = 101$ we know already that $Q(101, 91)$. Suppose $Q(x', 91)$ for every x' such that $x < x' \leq 101$; show that $Q(x, 91)$. We dis-

tinguish between two cases:

(a) $89 < x \leq 100$.

$\mathfrak{W}_{\hat{\mathcal{P}}}(Q)$ implies $\forall z_1 \forall z_2[Q(x + 11, z_1) \wedge Q(z_1, z_2) \supset Q(x, z_2)]$.

Take $z_1 = x + 1$ and $z_2 = 91$. Since $x + 11 > 100$, we know $Q(x + 11, x + 1)$. Thus we have $Q(x + 1, 91) \supset Q(x, 91)$, which implies by the induction hypothesis $Q(x, 91)$.

(b) $x \leq 89$.

$\mathfrak{W}_{\hat{\mathcal{P}}}(Q)$ implies $\forall z_1 \forall z_2[Q(x + 11, z_1) \wedge Q(z_1, z_2) \supset Q(x, z_2)]$. Take $z_1 = z_2 = 91$. Since $x + 11 \leq 100$, $Q(x + 11, 91)$ by the induction hypothesis. Moreover, we know $Q(91, 91)$ by case (a). Thus we have $Q(x, 91)$.

The interested reader can find an extension of this example in Manna and Pnueli [7]. In Appendix B of that paper we proved that the functional program $\mathcal{P}$ is correct with respect to $\varphi(x) \equiv T$ and $\psi(x, y)$, where:

(i) $\mathcal{P}$ is defined as:

$\mathcal{F}(x) \Leftarrow$ **if** $x > a$ **then** $x - b$ **else** $\mathcal{F}(\mathcal{F}(x + b + c))$,

$D = I$ (the integers), and a, b, and c are integers ($b, c > 0$); and

(ii) the predicate $\psi(x, y)$ is defined as:

if $x > a$ **then** $y = x - b$ **else** $y = a - b + c - \mod(a - x, c)$.

Note that for $a = 100$, $b = 10$, and $c = 1$, we obtain the above example as a special case.

The results of this section can easily be extended to functional program schemas.

We say that P *is [partially] correct with respect to* Φ *and* Ψ if for every interpretation $\mathcal{I}^+$, the functional program $(P, \mathcal{I})$ is [partially] correct with respect to φ and ψ.

From Theorem 1 and 2, respectively, there follows immediately:

THEOREM 3. *(a) P is partially correct with respect to Φ and Ψ if and only if $[P, \Phi, \Psi]$ is satisfiable,*

(b) P is correct with respect to Φ and Ψ if and only if $\{P, \Phi, \Psi\}$ is valid.

8. *Equivalence*

Given:

(a) functional programs $\mathcal{P}_1 = (P_1, \mathcal{I}_1)$ and $\mathcal{P}_2 = (P_2, \mathcal{I}_2)$, where the domain of $\mathcal{I}_i$ is D_i and $\mathcal{F}_i(\bar{x}_i)$ is the partial function (mapping D^{N_i} into D) defined by $\mathcal{P}_i$ (for $i = 1, 2$);

(b) a total predicate $\varphi(\bar{x}_1, \bar{x}_2)$ over $D_1^{N_1} \times D_2^{N_2}$ (called the *input predicate*); and

(c) a total predicate $\psi(\bar{x}_1, \bar{x}_2, y_1, y_2)$ over $D_1^{N_1} \times D_2^{N_2} \times D_1 \times D_2$ (called the *output predicate*).

We say that:

1. $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *are partially equivalent with respect to* φ *and* ψ if for every $\bar{\xi}_1 \in D_1^{N_1}$ and $\bar{\xi}_2 \in D_2^{N_2}$: if $[\varphi(\bar{\xi}_1, \bar{\xi}_2)$ and $*\mathcal{F}_1(\bar{\xi}_1)$ and $*\mathcal{F}_2(\bar{\xi}_2)]$ then $\psi(\bar{\xi}_1, \bar{\xi}_2, \mathcal{F}_1(\bar{\xi}_1), \mathcal{F}_2(\bar{\xi}_2))$.

2. $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *are equivalent with respect to* φ *and* ψ if for every $\bar{\xi}_1 \in D_1^{N_1}$ and $\bar{\xi}_2 \in D_2^{N_2}$: if $\varphi(\bar{\xi}_1, \bar{\xi}_2)$ then $[*\mathcal{F}_1(\bar{\xi}_1)$ and $*\mathcal{F}_2(\bar{\xi}_2)$ and $\psi(\bar{\xi}_1, \bar{\xi}_2, \mathcal{F}_1(\bar{\xi}_1), \mathcal{F}_2(\bar{\xi}_2))]$.

Let Φ be a predicate constant of $N_1 + N_2$ arguments and let Ψ be a predicate constant of $N_1 + N_2 + 2$ arguments. Let $\mathcal{I}^+$ stand for the interpretation of P_1, P_2, Φ, and Ψ, i.e. $\mathcal{I}^+$ consists of the assignments of $\mathcal{I}_1$ to the constants of P_1, the assignments of $\mathcal{I}_2$ to the constants of P_2, φ to Φ, and ψ to Ψ.

We make use of the following interpreted wffs:

1. $[\mathcal{P}_1, \mathcal{P}_2, \varphi, \psi] = ([P_1, P_2, \Phi, \Psi], \mathcal{J}^+)$, where the wff $[P_1, P_2, \Phi, \Psi]$ is

$$W_{P_1}(Q_1) \wedge W_{P_2}(Q_2) \wedge$$
$$\forall \bar{x}_1 \forall \bar{x}_2 \forall y_1 \forall y_2 \{[\Phi(\bar{x}_1, \bar{x}_2) \wedge Q_1(\bar{x}_1, y_1) \wedge Q_2(\bar{x}_2, y_2)] \supset \Psi(\bar{x}_1, \bar{x}_2, y_1, y_2)\},$$

and

2. $\{\mathcal{P}_1, \mathcal{P}_2, \varphi, \psi\} = (\{P_1, P_2, \Phi, \Psi\}, \mathcal{J}^+)$, where the wff $\{P_1, P_2, \Phi, \Psi\}$ is

$$W_{P_1}(Q_1) \wedge W_{P_2}(Q_2) \supset$$
$$\forall \bar{x}_1 \forall \bar{x}_2 \{\Phi(\bar{x}_1, \bar{x}_2) \supset \exists y_1 \exists y_2 [Q_1(\bar{x}_1, y_1) \wedge Q_2(\bar{x}_2, y_2) \wedge \Psi(\bar{x}_1, \bar{x}_2, y_1, y_2)]\}.$$

THEOREM 4. *$\mathcal{P}_1$ and $\mathcal{P}_2$ are partially equivalent with respect to φ and ψ if and only if $[\mathcal{P}_1, \mathcal{P}_2, \varphi, \psi]$ is satisfiable.*

PROOF. $\Rightarrow$ *Satisfiability.* Let δ_1 be the minimal valid predicate for $\mathcal{P}_1$ and δ_2 be the minimal valid predicate for $\mathcal{P}_2$. By Lemma 1, both $\mathfrak{W}_{\mathcal{P}_1}(\delta_1)$ and $\mathfrak{W}_{\mathcal{P}_2}(\delta_2)$ are true.

Since δ_1 and δ_2 are the minimal valid predicates for $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, i.e.

$$\forall (\bar{\xi}_1, \eta_1) \in D_1^{N_1+1} : \text{if } \delta_1(\bar{\xi}_1, \eta_1) \text{ then } [*\mathcal{F}_1(\bar{\xi}_1) \text{ and } \eta_1 = \mathcal{F}_1(\bar{\xi}_1)],$$

and

$$\forall (\bar{\xi}_2, \eta_2) \in D_2^{N_2+1} : \text{if } \delta_2(\bar{\xi}_2, \eta_2) \text{ then } [*\mathcal{F}_2(\bar{\xi}_2) \text{ and } \eta_2 = \mathcal{F}_2(\bar{\xi}_2)],$$

it follows that the last part of the conjunction is also true.

$\Leftarrow$ *Satisfiability.* Since both $\mathfrak{W}_{\mathcal{P}_1}(\delta_1)$ and $\mathfrak{W}_{\mathcal{P}_2}(\delta_2)$ are true (for some δ_1 and δ_2), it follows by Lemma 2 that δ_1 and δ_2 are valid predicates for $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, i.e.

$$\forall \bar{\xi}_1 \in D_1^{N_1} : \text{if } *\mathcal{F}_1(\bar{\xi}_1) \text{ then } \delta_1(\bar{\xi}_1, \mathcal{F}_1(\bar{\xi}_1)),$$

and

$$\forall \bar{\xi}_2 \in D_2^{N_2} : \text{if } *\mathcal{F}_2(\bar{\xi}_2) \text{ then } \delta_2(\bar{\xi}_2, \mathcal{F}_2(\bar{\xi}_2)).$$

Therefore, by the last part of the conjunction, we obtain that $\mathcal{P}_1$ and $\mathcal{P}_2$ are partially equivalent with respect to φ and ψ. Q.E.D.

THEOREM 5. *$\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent with respect to φ and ψ if and only if $\{\mathcal{P}_1, \mathcal{P}_2, \varphi, \psi\}$ is valid.*

PROOF. $\Rightarrow$ *Validity.* Let δ_1 be any predicate over $D_1^{N_1+1}$ for which $\mathfrak{W}_{\mathcal{P}_1}(\delta_1)$ is true, and δ_2 be any predicate over $D_2^{N_2+1}$ for which $\mathfrak{W}_{\mathcal{P}_2}(\delta_2)$ is true. Consider arbitrary $\bar{\xi}_1 \in D_1^{N_1}$ and $\bar{\xi}_2 \in D_2^{N_2}$ such that $\varphi(\bar{\xi}_1, \bar{\xi}_2)$.

Since both $\mathfrak{W}_{\mathcal{P}_1}(\delta_1)$ and $\mathfrak{W}_{\mathcal{P}_2}(\delta_2)$ are true, it follows by Lemma 2 that δ_1 and δ_2 are valid predicates for $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, i.e.

$$\text{if } *\mathcal{F}_1(\bar{\xi}_1) \text{ then } \delta_1(\bar{\xi}_1, \mathcal{F}_1(\bar{\xi}_1)),$$

and

$$\text{if } *\mathcal{F}_2(\bar{\xi}_2) \text{ then } \delta_2(\bar{\xi}_2, \mathcal{F}_2(\bar{\xi}_2)).$$

Moreover, since $\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent with respect to φ and ψ we have that

$$*\mathcal{F}_1(\bar{\xi}_1) \text{ and } *\mathcal{F}_2(\bar{\xi}_2) \text{ and } \psi(\bar{\xi}_1, \bar{\xi}_2, \mathcal{F}_1(\bar{\xi}_1), \mathcal{F}_2(\bar{\xi}_2)).$$

Thus

$$\exists y_1 \exists y_2 [\delta_1(\bar{\xi}_1, y_1) \wedge \delta_2(\bar{\xi}_2, y_2) \wedge \psi(\bar{\xi}_1, \bar{\xi}_2, y_1, y_2)]$$

is true.

$\Leftarrow$ *Validity.* Let δ_1 be the minimal valid predicate for $\mathcal{P}_1$ and δ_2 be the minimal valid predicate for $\mathcal{P}_2$. By Lemma 1, both $\mathfrak{W}_{\mathcal{P}_1}(\delta_1)$ and $\mathfrak{W}_{\mathcal{P}_2}(\delta_2)$ are true.

Consider arbitrary $\bar{\xi}_1 \in D_1^{N_1}$ and $\bar{\xi}_2 \in D_2^{N_2}$ such that $\varphi(\bar{\xi}_1, \bar{\xi}_2)$. The validity of $\{\mathcal{P}_1, \mathcal{P}_2, \varphi, \psi\}$ implies that $\exists \eta_1 \in D$ and $\exists \eta_2 \in D$ such that $\delta_1(\bar{\xi}_1, \eta_1)$ and $\delta_2(\bar{\xi}_2, \eta_2)$ and $\psi(\bar{\xi}_1, \bar{\xi}_2, \eta_1, \eta_2)$. Since δ_1 and δ_2 are the minimal valid predicates for $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, i.e.

$$\text{if } \delta_1(\bar{\xi}_1, \eta_1) \text{ then } [*\mathcal{F}_1(\bar{\xi}_1) \text{ and } \eta_1 = \mathcal{F}_1(\bar{\xi}_1)]$$
$$\text{if } \delta_2(\bar{\xi}_2, \eta_2) \text{ then } [*\mathcal{F}_2(\bar{\xi}_2) \text{ and } \eta_2 = \mathcal{F}_2(\bar{\xi}_2)],$$

it follows that

$$*\mathcal{F}_1(\bar{\xi}_1) \text{ and } *\mathcal{F}_2(\bar{\xi}_2) \text{ and } \psi(\bar{\xi}_1, \bar{\xi}_2, \mathcal{F}_1(\bar{\xi}_1), \mathcal{F}_2(\bar{\xi}_2)). \qquad \text{Q.E.D.}$$

Note that Theorems 4 and 5 can be used to prove some properties of a single functional program which cannot be proved by Theorems 1 and 2. For example, the property that for a functional program $\mathcal{P}$ (where $D = \{\text{the integers}\}$) the function $\mathcal{F}(x)$ is always defined and monotonically increasing (i.e. $x > x' \Rightarrow \mathcal{F}(x) > \mathcal{F}(x')$), is exactly the case where the functional programs $\mathcal{P}$ and $\mathcal{P}'$ (where $\mathcal{P}'$ is identical to $\mathcal{P}$) are equivalent with respect to $\varphi(x, x'): T$ and $\psi(x, x', y, y'): x > x' \supset y > y'$.

The results of this section can easily be extended to functional program schemas.

We say that P_1 *and* P_2 *are* [*partially*] *equivalent with respect to* Φ *and* Ψ if for every interpretation $\mathcal{I}^+$, the functional programs $(P_1, \mathcal{I}_1)$ and $(P_2, \mathcal{I}_2)$ are [partially] equivalent with respect to φ and ψ.

From Theorems 4 and 5, respectively, it follows immediately that:

THEOREM 6. *(a) P_1 and P_2 are partially equivalent with respect to Φ and Ψ if and only if $[P_1, P_2, \Phi, \Psi]$ is satisfiable,*

(b) P_1 and P_2 are correct with respect to Φ and Ψ if and only if $\{P_1, P_2, \Phi, \Psi\}$ is valid.

The partial equivalence property introduced in this section is also called "weak equivalence," while the second equivalence property is sometimes called "total equivalence." Usually one is interested in a different type of equivalence, called "strong equivalence." Using our notations it can be defined as follows: $\mathcal{P}_1$ and $\mathcal{P}_2$ are *strongly equivalent with respect to* φ *and* ψ if for every $\bar{\xi}_1 \in D_1^{N_1}$ and $\bar{\xi}_2 \in D_2^{N_2}$: if $\varphi(\bar{\xi}_1, \bar{\xi}_2)$ then either both $\mathcal{F}_1(\bar{\xi}_1)$ and $\mathcal{F}_2(\bar{\xi}_2)$ are undefined, or both are defined and $\psi(\bar{\xi}_1, \bar{\xi}_2, \mathcal{F}_1(\bar{\xi}_1), \mathcal{F}_2(\bar{\xi}_2))$. In general, this property cannot be formalized by first-order formulas, but can be formalized by second-order formulas as was done for flowchart programs by Cooper [2].

For all these definitions (weak equivalence, total equivalence, and strong equivalence) the most interesting case is clearly the one where $N_1 = N_2$, $D_1 = D_2$, φ is $\bar{\xi}_1 = \bar{\xi}_2$, and ψ is $\mathcal{F}_1(\bar{\xi}_1) = \mathcal{F}_2(\bar{\xi}_2)$. The reader should be aware that in considering this special case, among the above three definitions of equivalence, only the last one is really an equivalence relation (i.e. reflexive, symmetric, and transitive). Total equivalence is symmetric and transitive but not reflexive, while weak equivalence is reflexive and symmetric but not transitive.

9. *Extension for Recursive Systems*

So far we have restricted ourselves to functional programs which are defined by a single recursive definition. In general, one may define functional programs $\mathcal{P} = (P, \mathcal{I})$, where P is a system of several recursive definitions of the form:

$$\begin{aligned}
\mathfrak{F}_1(\bar{x}) &\Leftarrow \tau_1(\bar{x}, \mathfrak{F}_1, \mathfrak{F}_2, \cdots, \mathfrak{F}_n),\\
\mathfrak{F}_2(\bar{x}) &\Leftarrow \tau_2(\bar{x}, \mathfrak{F}_1, \mathfrak{F}_2, \cdots, \mathfrak{F}_n),\\
&\vdots\\
\mathfrak{F}_n(\bar{x}) &\Leftarrow \tau_n(\bar{x}, \mathfrak{F}_1, \mathfrak{F}_2, \cdots, \mathfrak{F}_n),
\end{aligned}$$

i.e. each τ_i is a conditional term that may contain the function variables $\mathfrak{F}_1, \mathfrak{F}_2, \cdots, \mathfrak{F}_n$. The extension of our results to the more general case is straightforward.

For each recursive definition P_i, define W_{P_i} exactly as before (see Section 5), except that each function symbol $\mathfrak{F}_i$ is represented in the formula by a predicate symbol Q_i (of $N + 1$ arguments). Then define W_P as $\bigwedge_{i=1}^{n} W_{P_i}$.

Using the modified W_P, all the results of Sections 7 and 8 can be extended to formalize properties of the partial function $\mathfrak{F}_1$ defined by $\mathcal{P}$.

ACKNOWLEDGMENTS. We are indebted to Professor John McCarthy for his encouragement while we worked on this research. We are also grateful to Lockwood Morris, Stephen Ness, and Richard Waldinger for their detailed reading of the manuscript.

REFERENCES

1. CHURCH, A. *Introduction to Mathematical Logic, Vol. 1.* Princeton U. Press, Princeton, N. J., 1956.
2. COOPER, D. C. Program scheme equivalences and second order logic. In *Machine Intelligence, Vol. 4*, Meltzer, B., and Michie, D. (Eds.), Edinburgh U. Press, Edinburgh, 1969, pp. 3–15.
3. FLOYD, R. W. Assigning meaning to programs. In Proc. Symposia in Appl. Math. (1966), Vol. 19 (Schwartz, J. T., Ed.), Amer. Math. Soc., Providence, R. I., 1967, pp. 19–32.
4. MANNA, Z. Properties of programs and the first-order predicate calculus. *J. ACM 16*, 2 (April 1969), 244–255.
5. ——. The correctness of programs. *J. Comput. System Sci. 3*, 2 (May 1969), 119–127.
6. —— AND MCCARTHY, J. Properties of programs and partial function logic. In *Machine Intelligence, Vol. 5*, Michie, D. (Ed.), Edinburgh U. Press, Edinburgh, 1970.
7. —— AND PNUELI, A. The validity problem of the 91-function. Memo No. AI-68, Stanford U. Artificial Intelligence Report, Stanford, Calif., Aug. 1968.
8. MCCARTHY, J. Towards a mathematical science of computation. In Information Processing 1962, Proc. IFIP Congress 62, Popplewell, C. M. (Ed.), North-Holland, Amsterdam, 1963, pp. 21–28.
9. ——. A basis for a mathematical theory of computation. In Computer Programming and Formal Systems, Braffort, P., and Hirschberg, D. (Eds.), North-Holland, Amsterdam, 1963, pp. 33–70.
10. MENDELSON, E. *Introduction to Mathematical Logic.* D. Van Nostrand, Princeton, N. J., 1964.

RECEIVED MARCH, 1969; REVISED JANUARY, 1970