Some Completeness Results in the Mathematical Theory of Computation

DONALD M. KAPLAN

Stanford University, Stanford, California*

ABSTRACT. A formal theory is described which incorporates the "assignment" function $a(i, k, \xi)$ and the "contents" function $c(i, \xi)$. The axioms of the theory are shown to comprise a complete and consistent set.

KEY WORDS AND PHRASES: axioms, formal theory, logic, completeness, formal logic consistency, theory of computation, mathematical logic

CR CATEGORIES: 5.21, 5.24, 5.29

In [1], McCarthy introduces computer program state vectors along with two functions used to compute explicitly with them. Certain facts about these functions are given in [1], and in this paper it is shown that these facts constitute a complete set of axioms in an appropriate formal theory $\mathfrak{I}$.

Before developing this formal theory, we discuss state vectors and functions on them so that the relevance of formulas in $\mathfrak{I}$ is apparent.

A *state vector* is simply a family of quantities $\{x_i\}, i \in I, \quad I = \{1, 2, \cdots\}$, where for each i, $x_i \in D$, where D is the set of all possible quantities. The exact nature of D is left unspecified here, but many applications would find D containing such subsets as the integers, certain character strings, finite bit sequences, the real numbers, etc.

Distinct from state vectors as defined above is the *program state vector* which, for a program at a given point during its execution, indicates the set of current assignments of values to the variables of the program. In the case of a machine language program, the program state vector indicates the set of current contents of those registers whose contents change during the course of program execution. Specifically, the program state vector for a program π is defined as the ordered pair (M_π, ξ) where ξ is a state vector in the general sense described above and M_π is a 1-1 mapping of program variables (or register addresses) into the positive integers. Thus, if v is a variable of π then the $M_\pi(v)$-th term in ξ is the quantity currently assigned to v.

In order to compute explicitly with state vectors, we introduce the function c: $I, V \to D$, where $V = \{x : x \text{ is a state vector}\}$, and write $c(i, \xi)$ to denote the ith term (quantity) of ξ. We also introduce the function a: $I, D, V \longrightarrow V$ and write $a(i, k, \xi)$ to denote the state vector that results when the ith term in ξ is replaced with the quantity k and the other terms in ξ are left unchanged. In [1], an example is given showing the transformation of a simple computer program into expressions utilizing this formalism.

We now define a formal theory $\mathfrak{I}' = (Fm', Ax', \mathbf{R1}, \mathbf{R2})$. We follow [2] here.

(1) $\mathfrak{I}'$ has the following countable set of symbols: parentheses, brackets, the

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

comma, individual constant letters $k_1, k_2, \cdots$, ordinary function letters $f_1^1, f_1^2, \cdots, f_k^n, \cdots$ where f_k^n is the kth function of n arguments, index letters $i_{11}, i_{12}, \cdots, i_{nk}, \cdots$, state letters $\xi_1, \xi_2, \cdots$, the special function letters a and c, and the predicate symbol "$=$".

A finite sequence of symbols of $\mathfrak{J}'$ is called an expression of $\mathfrak{J}'$. If p and q are expressions of $\mathfrak{J}'$, then $p =_E q$ means p and q are symbol for symbol the same expression of $\mathfrak{J}'$.

(2) There is an effectively decidable subset Fm' of the expressions of $\mathfrak{J}'$ called the set of well-formed formulas (wfs) of $\mathfrak{J}$. Wfs are defined recursively as follows.

First we define *terms*:

(a) Individual constant letters k_i are terms.

(b) If f_k^n is a function letter, and $t_1, \cdots, t_n$ are terms, then $f_k^n[t_1, \cdots, t_n]$ is a term.

(c) If i_{nk} is an index letter and ξ is an n-state (see definition below), then $c(i_{nk}, \xi)$ is a term.

(d) An expression is a term only if it can be shown to be a term on the basis of clauses (a), (b), and (c).

Next, we define *n-states*:

(a) The state letter ξ_n is an n-state.

(b) If i_{nk} is an index letter, and t is a term, and ξ is an n-state, then $a(i_{nk}, t, \xi)$ is an n-state.

(c) An expression is an n-state only if it can be shown to be an n-state on the basis of clauses (a) and (b).

The definitions given above for terms and n-states are potentially infinitely recursive, so that the restriction must be added that terms and n-states be finite expressions of $\mathfrak{J}'$.

Finally, the predicate symbol "$=$" applied to terms yields the wfs of $\mathfrak{J}'$. Specifically, an expression of $\mathfrak{J}'$ is a *well-formed formula* if and only if it is of the form $t_1 = t_2$ where t_1 and t_2 are terms.

(3) A set $Ax' \subseteq Fm'$ is set aside and called the set of axioms of $\mathfrak{J}'$. The set Ax' is defined by two axiom schemas:

$$\textbf{A1} \quad c(i, a(j, t, \xi)) = c(i, \xi),$$
$$\textbf{A2} \quad c(i, a(i, t, \xi)) = t,$$

where i and j are any distinct index letters of the form i_{nk}, t is any term, and ξ is any n-state, for any $n = 1, 2, \cdots$. Note that i, j, t, ξ are used throughout with the above meanings. Since Ax' is an effectively decidable subset of Fm', $\mathfrak{J}'$ is therefore an axiomatic theory.

(4) The rules of inference of $\mathfrak{J}'$ are

$$\textbf{R1} \quad (t_1 = t_2, t_2 = t_1),$$

$$\textbf{R2} \quad (\mathfrak{A}(t_1, t_1), t_1 = t_2, \mathfrak{A}(t_1, t_2)),$$

where t_1 and t_2 are terms and where the wf $\mathfrak{A}(t_1, t_2)$ arises from $\mathfrak{A}(t_1, t_1)$ by replacing some, but not necessarily all, occurrences of t_1 by t_2. Obviously, these rules characterize the reflexivity and substitutivity properties of equality; in a first-order theory, the axioms associated with equality would be introduced along with the predicate and *modus ponens* would then be the operative rule of inference. However, framing of the completeness derived in this paper within a full propositional and quantificational structure would only add more complexity, not more significance, to the results obtained. We write $\vdash \mathfrak{A}$ if and only if $\mathfrak{A} \in Fm'$ and is derivable using Ax', **R1** and **R2**.

Of course, a wf $\mathfrak{A}$ has meaning only when an interpretation is given for the symbols in $\mathfrak{A}$. An *interpretation* consists of

(a) a nonempty set D (the domain) of quantities,

(b) a mapping of ordinary function letters f_k^n into n-place operations in D,

(c) a mapping of individual constant letters k_i into D,

(d) 1-1 mapping of the index letters i_{nk}, $k = 1, 2, \cdots$, into I for $n = 1, 2, \cdots$,

(e) a mapping of state letters into families with index set I and terms chosen from D,

(f) an assignment of the equality relation in D to the predicate symbol "=". Thus, we write $x =_D y$ if and only if x and y are the same element of D.

The notion of truth for wfs of $\mathfrak{I}'$ is intuitively clear, but can be precisely defined as follows. Let there be given an interpretation with domain D. We now define what it means for a wf $\mathfrak{A}$ to be *true* in the given interpretation. As a preliminary step, we define a total function v^* of one argument, which maps terms into quantities in D.

(a) If k_i is an individual constant letter, then $v^*(k_i)$ is the interpretation in D of this constant.

(b) If f_k^n is an ordinary function letter and g is the corresponding n-place operation in D given by the interpretation, and $t_1, \cdots, t_n$ are terms, then $v^*(f_k^n[t_1, \cdots, t_n]) =_D g(v^*(t_1), \cdots, v^*(t_n))$.

(c) If i_{nk} is an index letter and ξ an n-state, and if the integer $p \in I$ is the interpretation of i_{nk}, then $v^*(c(i_{nk}, \xi)) =_D p$th term of $u^*(\xi)$, where u^* is defined below.

The total function u^* of one argument maps n-states into families with elements chosen from D.

(a) If ξ_n is a state letter, then $u^*(\xi_n)$ is the family, with index set I and terms chosen from D, given for ξ_n by the interpretation.

(b) If i_{nk} is an index letter, t a term, and ξ an n-state, and if the integer $p \in I$ is the interpretation of i_{nk}, then $u^*(a(i_{nk}, t, \xi))$ is defined as having the same terms as $u^*(\xi)$ except for the pth term which is $v^*(t)$.

From the definitions above we can easily obtain the following two results, which are needed later on.

(a) $v^*(c(i_{nk}, a(i_{nl}, t, \xi))) =_D v^*(c(i_{nk}, \xi))$ if $k \neq l$.

(b) $v^*(c(i_{nk}, a(i_{nk}, t, \xi))) =_D v^*(t)$.

The definition of truth for wfs is then: a wf $\mathfrak{A}$ of the form $t_1 = t_2$ is *true* (for a given interpretation) if and only if $v^*(t_1) =_D v^*(t_2)$, and a wf $\mathfrak{A}$ is *valid* (according to $\mathfrak{I}'$) if and only if $\mathfrak{A}$ is true "in all interpretations" (abbreviated i.a.i.). We write $\models \mathfrak{A}$ if and only if $\mathfrak{A}$ is a valid wf.

Before moving on to the completeness theorems for $\mathfrak{I}'$, we need a few preliminary definitions, two lemmas, and a proposition.

(a) A *simple term* is a term of the form $c(i, \xi_n)$ or k_i or $f_k^n[t_1, \cdots, t_n]$, where $t_1, \cdots, t_n$ are terms.

(b) A *totally simple term* is a term of the form $c(i, \xi_n)$ or k_i or $f_k^n[t_1, \cdots, t_n]$, where $t_1, \cdots, t_n$ are totally simple terms.

LEMMA 1. *If t_1 and t_2 are totally simple terms and $\models t_1 = t_2$ then $t_1 =_E t_2$.*

PROOF. We give an inductive proof where cases (a) and (b) below constitute the primitive induction level.

(a) $t_1 =_E k_i$ for some i.

$$\models t_1 = t_2 \Rightarrow v^*(k_i) =_D v^*(t_2) \text{ i.a.i.}$$
$$\Rightarrow t_2 =_E k_i$$
$$\Rightarrow t_1 =_E t_2 .$$

(b) $t_1 =_E c(i_{nk}, \xi_n)$ for some n and k.

$$\models t_1 = t_2 \Rightarrow v^*(c(i_{nk}, \xi_n)) =_D v^*(t_2) \text{ i.a.i.}$$
$$\Rightarrow t_2 =_E c(i_{nk}, \xi_n)$$
$$\Rightarrow t_1 =_E t_2 .$$

(c) $t_1 =_E f_k^n[t_{11}, t_{12}, \cdots, t_{1n}]$ for some n, k, and terms $t_{11}, t_{12}, \cdots, t_{1n}$.

$$\models t_1 = t_2 \Rightarrow v^*(f_k^n[t_{11}, t_{12}, \cdots, t_{1n}]) =_D v^*(t_2) \text{ i.a.i.}$$
$$\Rightarrow t_2 =_E f_k^n[t_{21}, t_{22}, \cdots, t_{2n}]$$
$$\text{and } v^*(t_{1j}) =_D v^*(t_{2j}), \qquad j = 1, 2, \cdots, n \text{ i.a.i.}$$

But, t_1 and t_2 are totally simple terms, therefore $t_{11}, t_{12}, \cdots, t_{1n}, t_{21}, t_{22}, \cdots, t_{2n}$ are also totally simple terms. Assume the lemma holds for t_{11} and t_{21}, t_{12} and t_{22}, $\cdots$, t_{1n} and t_{2n}. Then $t_{11} =_E t_{21}$, $t_{12} =_E t_{22}$, $\cdots$, $t_{1n} =_E t_{2n}$. Thus, $t_1 =_E t_2$. This completes the induction and the proof of the lemma. The essential point here is simply that two totally simple terms cannot be equal in *all* interpretations unless they are the same term.

LEMMA 2. *If* $\models t_1 = t_2$, $\models t_3 = t_4$, $\models t_1 = t_3$, *then* $\models t_2 = t_4$ *where* t_1, t_2, t_3, *and* t_4 *are terms.*

PROOF. From the given information we have

$$v^*(t_1) =_D v^*(t_2) \text{ i.a.i.,}$$
$$v^*(t_3) =_D v^*(t_4) \text{ i.a.i.,}$$
$$v^*(t_1) =_D v^*(t_3) \text{ i.a.i.}$$

In any particular interpretation, we have certainly the three equalities above. Hence $v^*(t_2) =_D v^*(t_4)$ in this interpretation. Since any interpretation gives this result, we have $v^*(t_2) =_D v^*(t_4)$ i.a.i., that is $\models t_2 = t_4$.

PROPOSITION 1 (**P1**). $\vdash t = t$ *where* t *is any term.*

PROOF.

(1)	$c(i, a(i, t, \xi)) = t$,	**A2**
(2)	$c(i, a(i, t, \xi)) = t$,	**A2**
(3)	$t = t$.	**R2** on (1) and (2)

THEOREM 1 (**T1**). $\vdash \mathfrak{A} \Rightarrow \models \mathfrak{A}$; *that is, all theorems of* $\mathfrak{T}'$ *are valid.*

PROOF. First we show that the axioms are valid.

A1: $c(i, a, (j, t, \xi)) = c(i, \xi)$.

$v^*(c(i, a(j, t, \xi))) =_D v^*(c(i, \xi))$ i.a.i. from the definition of v^*.

So, $\models c(i, a(j, t, \xi)) = c(i, \xi)$.

A2: $c(i, a(i, t, \xi)) = t$.

$v^*(c(i, a(i, t, \xi))) =_D v^*(t)$ i.a.i. from the definition of v^*.

So, $\models c(i, a(i, t, \xi)) = t$.

Next we show that the rules of inference **R1** and **R2** preserve validity. Consider $\mathbf{R1}(t_1 = t_2, t_2 = t_1)$ and assume $\models t_1 = t_2$. Then, by the definition of validity, $v^*(t_1) =_D v^*(t_2)$ i.a.i. or alternatively we can write $v^*(t_2) =_D v^*(t_1)$ i.a.i., that is $\models t_2 = t_1$. Thus, **R1** is seen to preserve validity.

Consider $\mathbf{R2}(\mathfrak{A}(t_1, t_1), t_1 = t_2, \mathfrak{A}(t_1, t_2))$ and assume $\models \mathfrak{A}(t_1, t_1)$ and $\models t_1 = t_2$. The validity of $\mathfrak{A}(t_1, t_1)$ depends on the quantities in D into which v^* maps the terms in $\mathfrak{A}(t_1, t_1)$. Clearly, if $v^*(t_1) =_D v^*(t_2)$ i.a.i., then substitution of the term t_2 for occurrences of the term t_1 in $\mathfrak{A}(t_1, t_1)$ to generate $\mathfrak{A}(t_1, t_2)$ will not affect this validity, so that $\models \mathfrak{A}(t_1, t_2)$ as well.

Since the axioms are valid and the rules of inference preserve validity, then all theorems of $\mathfrak{T}'$ are valid.

THEOREM 2. COMPLETENESS OF $\mathfrak{T}'$ (**T2**). $\models \mathfrak{A} \Rightarrow\ \vdash \mathfrak{A}$; *i.e., all valid wfs are theorems of* $\mathfrak{T}'$.

PROOF. We give a constructive proof so that given any valid wf $\mathfrak{A}$, we show how to construct a proof of $\mathfrak{A}$.

First, we define a total function r^* of one argument which maps terms into other terms.

(a) If t_1 is a simple term, then $r^*(t_1) =_E t_1$.
(b) If $t_1 =_E c(i, a(j, t, \xi))$, then $r^*(t_1) =_E c(i, \xi)$.
(c) If $t_1 =_E c(i, a(i, t, \xi))$, then $r^*(t_1) =_E t$.

We now define two proof-generating procedures, each accepting a single term as parameter. An ALGOL-like representation is used for clarity and brevity. Boolean procedures *simpleterm*(t) and *totallysimpleterm*(t) are assumed available, an assumption warranted by the decidability of the set of all terms. The procedure *proofstep*(x, y) emits a proof step x with justification y.

```
term procedure reduce(t);   value t;   term t;
begin comment  this procedure constructs a proof of ⊢t = q where q is a simple term. q is
  returned as the value of the procedure;
  term p, q;
  proofstep(t = t, P1);
  q ← t;
  while ¬ simpleterm(q) do
    begin
      p ← r*(q);
      proofstep(q = p, A1 or A2);
      proofstep(t = p, R2);
      q ← p
    end;
  reduce ← q
end procedure reduce;
```

To verify that the sequence of proof steps generated by procedure *reduce*(t) constitutes a valid proof, it is only necessary to observe that $\vdash x = r^*(x)$ for all terms x and that the applications of **R2** are correct. In the next procedure a certain looseness of notation is introduced, but the meaning should be apparent.

```
term procedure reducetotally(t);  value t;  term t;
begin comment  this procedure constructs a proof of ⊢t = t' where t' is a totally simple term.
  t' is returned as the value of the procedure;
  term t;
  integer j, n;
  t' ← t;
  if ¬ simpleterm(t) then t' ← reduce(t)
                     else proofstep(t = t, P1);
  if ¬ totallysimpleterm(t') then
  comment  t' =_E f_k^n[t_1 , ··· , t_n] where at least one of the terms t_1 , ··· , t_n is not totally
    simple;
  for j ← 1 step 1 until n do
    begin
      t' ← f_k^n[t_1 , ··· , t_{j-1} , reducetotally(t_j), t_{j+1} , ··· , t_n];
      proofstep(t = t', R2)
    end;
  reducetotally ← t'
end procedure reducetotally;
```

Consider now any wf $\mathcal{A}$ where $\models \mathcal{A}$. Suppose $\mathcal{A}$ is of the form $t_1 = t_2$ where t_1 and t_2 are terms. Construct proofs of $\vdash t_1 = t_1^*$ and $\vdash t_2 = t_2^*$ using procedures *reduce*(t) and *reducetotally*(t) where both t_1^* and t_2^* are totally simple terms.

From **T1** we then have that $\models t_1 = t_1^*$ and $\models t_2 = t_2^*$. But, we are given that $\models t_1 = t_2$; from Lemma 2 we then obtain $\models t_1^* = t_2^*$. Since t_1^* and t_2^* are totally simple terms, we can apply Lemma 1 and obtain $t_1^* =_E t_2^*$. That is, t_1^* and t_2^* are the same term, symbol for symbol. That being so, **P1** gives $\vdash t_1^* = t_2^*$.

So, from the proof constructions we have $\vdash t_1 = t_1^*$ and $\vdash t_2 = t_2^*$ and we write $\vdash t_1^* = t_2^*$ as an instance of **P1**. Using **R2** on $\vdash t_1 = t_1^*$ and $\vdash t_1^* = t_2^*$ gives $\vdash t_1 = t_2^*$, which on applying **R1** gives $\vdash t_2^* = t_1$. This last result together with $\vdash t_2 = t_2^*$ and **R2** gives $\vdash t_2 = t_1$, which on applying **R1** gives $\vdash t_1 = t_2$, i.e., $\vdash \mathcal{A}$, which completes the proof of completeness for $\mathfrak{I}'$.

We now have a theory $\mathfrak{I}'$ which is complete and so can be used to deduce all valid statements about equality of terms. However, in a potentially more powerful theory we should also be able to deduce all valid statements about equality of n-states (i.e., state vectors). To form a new theory $\mathfrak{I} = (Fm, Ax, \textbf{R1}, \textbf{R2}, \textbf{R3}, \textbf{R4})$ that will allow such deductions, we modify $\mathfrak{I}'$ in the manner outlined below.

(1) $\mathfrak{I}$ has the same countable set of symbols as $\mathfrak{I}'$ with the addition of the predicate symbol "$\equiv$".

(2) $Fm = Fm' \cup \{x : x$ is an expression of $\mathfrak{I}$ of the form $\theta_1 \equiv \theta_2$ where θ_1 and θ_2 are n-states for any $n = 1, 2, \cdots\}$. The expressions in Fm and only those are the wfs of $\mathfrak{I}$.

(3) $Ax = Ax' \cup \{x : x$ is an instance of axiom schema **A3**, **A4**, or **A5**$\}$.

A3 $a(i, t, a(j, s, \xi)) \equiv a(j, s, a(i, t, \xi))$.

A4 $a(i, t, a(i, s, \xi)) \equiv a(i, t, \xi)$.

A5 $a(i, c(i, \xi), \xi) \equiv \xi$.

(4) The rules of inference of $\mathfrak{I}$ are **R1** and **R2** (extended to all wfs of $\mathfrak{I}$) together with

$$\mathbf{R3} \quad (\theta_1 \equiv \theta_2\,,\ \theta_2 \equiv \theta_1)$$

and

$$\mathbf{R4} \quad (\mathfrak{A}(\theta_1\,,\ \theta_1),\ \theta_1 \equiv \theta_2\,,\ \mathfrak{A}(\theta_1,\ \theta_2))$$

where $\mathfrak{A} \in Fm$ and θ_1 and θ_2 are n-states for any $n = 1, 2, \cdots$. The same comments offered earlier about **R1** and **R2** are relevant here regarding **R3** and **R4**. We write $\vdash\mathfrak{A}$ if and only if $\mathfrak{A}$ is a wf and is derivable using Ax and **R1**, **R2**, **R3**, and **R4**.

As before, a wf $\mathfrak{A} \in Fm$ has meaning only when an interpretation is given to the symbols in $\mathfrak{A}$. An interpretation is defined exactly as before with the addition that an assignment of the equality relation in the set of all families with index set I and terms chosen from D is made to the predicate symbol "$\equiv$". Thus, we write $\{x_i\} \equiv_D \{y_i\}$ where $i \in I$ and $x_i\,,\ y_i \in D$, if and only if $x_i =_D y_i\,,\ i = 1, 2, \cdots$.

The definitions of truth and validity for wfs are extended as follows: a wf $\mathfrak{A}$ of the form $\theta_1 \equiv \theta_2$, where θ_1 and θ_2 are n-states is *true* (for a given interpretation) if and only if $u^*(\theta_1) \equiv_D u^*(\theta_2)$, and $\mathfrak{A}$ is *valid* (according to $\mathfrak{I}$) if and only if $\mathfrak{A}$ is true in all interpretations. From the 1-1 nature of the mappings given by the interpretation for index letters and from the definition of u^*, we obtain that $u^*(\theta_1) \equiv_D u^*(\theta_2)$ if and only if $v^*(c(i_{nk}\,,\ \theta_1)) =_D v^*(c(i_{nk}\,,\ \theta_2))$ for each $k = 1, 2, \cdots$; i.e., $\theta_1 \equiv \theta_2$ is true if and only if $c(i_{nk}\,,\ \theta_1) = c(i_{nk}\,,\ \theta_2)$ is true for each $k = 1, 2, \cdots$. Then, $\theta_1 \equiv \theta_2$ is valid if and only if $c(i_{nk}\,,\ \theta_1) = c(i_{nk}\,,\ \theta_2)$ is valid for each $k = 1, 2, \cdots$. As before, we write $\models\mathfrak{A}$ if and only if $\mathfrak{A}$ is a valid wf.

Before moving on to the completeness theorems for $\mathfrak{I}$, we need a lemma, a proposition, and a definition.

LEMMA 3. *If* $\models\theta_1 \equiv \theta_2\,,\ \models\theta_3 \equiv \theta_4\,,\ \models\theta_1 \equiv \theta_3$, *then* $\models\theta_2 \equiv \theta_4$ *where* $\theta_1\,,\ \theta_2\,,\ \theta_3$, *and* θ_4 *are* n-*states for any* $n = 1, 2, \cdots$.

PROOF. From the given information, we have

$$\models c(i_{nk}\,,\ \theta_1) = c(i_{nk}\,,\ \theta_2), \qquad k = 1, 2, \cdots,$$

$$\models c(i_{nk}\,,\ \theta_3) = c(i_{nk}\,,\ \theta_4), \qquad k = 1, 2, \cdots,$$

$$\models c(i_{nk}\,,\ \theta_1) = c(i_{nk}\,,\ \theta_3), \qquad k = 1, 2, \cdots.$$

By applying Lemma 2 to the above three statements for $k = 1, 2, \cdots$, we obtain

$$\models c(i_{nk}\,,\ \theta_2) = c(i_{nk}\,,\ \theta_4), \qquad k = 1, 2, \cdots;$$

i.e., $\models\theta_2 \equiv \theta_4$.

PROPOSITION 2 (**P2**). $\vdash\xi \equiv \xi$ *where* ξ *is any* n-*state for* $n = 1, 2, \cdots$.

PROOF.

(1) $a(i, c(i, \xi), \xi) \equiv \xi$, **A5**

(2) $a(i, c(i, \xi), \xi) \equiv \xi$, **A5**

(3) $\xi \equiv \xi$. **R4** on (1) and (2)

We define the *canonical form* for n-states to be

$$a(i_{nk_1}, t_1\,,\ a(i_{nk_2}, t_2\,,\ a(\cdots a(i_{nk_p}, t_p\,,\ \xi_n)\cdots)))$$

where if $p = 0$ we have simply ξ_n, where $k_1 < k_2 \cdots < k_p$ and where $t_1, t_2, \cdots, t_p$ are totally simple terms such that $t_j \neq_E c(i_{nk_j}, \xi_n)$ for $j = 1, 2, \cdots, p$.

THEOREM 3 (**T3**). $\vdash \mathfrak{A} \Rightarrow \models \mathfrak{A}$; *that is, all theorems of* $\mathfrak{I}$ *are valid.*

PROOF. First we show that the new axioms given by **A3**, **A4**, and **A5** are valid.

A3: $a(i, t, a(j, s, \xi)) \equiv a(j, s, a(i, t, \xi))$.

Consider the following deduction and the result obtained; m, i, and j are distinct.

(1)	$c(m, a(i, t, a(j, s, \xi))) = c(m, a(j, s, \xi))$,	**A1**
(2)	$c(m, a(j, s, \xi)) = c(m, \xi)$,	**A1**
(3)	$c(m, a(i, t, a(j, s, \xi))) = c(m, \xi)$,	**R2** on (1) and (2)
(4)	$c(m, a(j, s, a(i, t, \xi))) = c(m, a(i, t, \xi))$,	**A1**
(5)	$c(m, a(i, t, \xi)) = c(m, \xi)$,	**A1**
(6)	$c(m, a(j, s, a(i, t, \xi))) = c(m, \xi)$,	**R2** on (4) and (5)
(7)	$c(m, \xi) = c(m, a(j, s, a(i, t, \xi)))$,	**R1** on (6)
(8)	$c(m, a(i, t, a(j, s, \xi))) = c(m, a(j, s, a(i, t, \xi)))$.	**R2** on (3) and (7)

Then **T1** gives $\models c(m, a(i, t, a(j, s, \xi))) = c(m, a(j, s, a(i, t, \xi)))$ for any m distinct from i and j. Consider now the following deduction and the result obtained; i and j are distinct and $m =_E i$.

(1)	$c(m, a(i, t, a(j, s, \xi))) = t$,	**A2**
(2)	$c(m, a(j, s, a(i, t, \xi))) = c(m, a(i, t, \xi))$,	**A1**
(3)	$c(m, a(i, t, \xi)) = t$,	**A2**
(4)	$c(m, a(j, s, a(i, t, \xi))) = t$,	**R2** on (2) and (3)
(5)	$t = c(m, a(j, s, a(i, t, \xi)))$,	**R1** on (4)
(6)	$c(m, a(i, t, a(j, s, \xi))) = c(m, a(j, s, a(i, t, \xi)))$.	**R2** on (1) and (5)

Then **T1** gives $\models c(m, a(i, t, a(j, s, \xi))) = c(m, a(j, s, a(i, t, \xi)))$ for distinct i and j and $m =_E i$.

A proof similar to that above gives the same result for distinct i and j and $m =_E j$. Thus, for all m we have $\models c(m, a(i, t, a(j, s, \xi))) = c(m, a(j, s, a(i, t, \xi)))$, and so by the definition of validity $\models a(i, t, a(j, s, \xi)) = a(j, s, a(i, t, \xi))$. Hence, the axioms given by **A3** are all valid.

Proofs structured like the one above can be given to show that the axioms given by **A4** and **A5** are all valid.

As well, straightforward arguments like those in **T1** can be given to show that **R2** extended to wfs of $\mathfrak{I}$ and **R3** and **R4** lead from valid wfs to other valid wfs.

Thus, since the axioms of $\mathfrak{I}$ are valid and since the rules of inference of $\mathfrak{I}$ preserve validity, then all theorems of $\mathfrak{I}$ are valid.

THEOREM 4. COMPLETENESS OF $\mathfrak{I}$ (**T4**). $\models \mathfrak{A} \Rightarrow \vdash \mathfrak{A}$, *i.e., all valid wfs are theorems of* $\mathfrak{I}$.

PROOF. This theorem has already been proved (**T2**) for wfs of the form $t_1 = t_2$

where t_1 and t_2 are terms. Thus, we consider here only wfs $\mathfrak{A}$ of the form $\theta_1 \equiv \theta_2$. Once again a constructive proof is given. Several proof-constructing procedures are used in addition to *reduce*(t) and *reducetotally*(t) defined above. For the description of the new procedures, an ALGOL-like representation is again used.

The first three procedures are used to manipulate n-states s and construct proofs of $\vdash s = s'$, where s' is the result of the manipulation. In these procedures, the the argument s is of the form

$$a(j_1\,,\, t_1\,,\, a(j_2\,,\, t_2\,,\, a(\cdots a(j_m\,,\, t_m\,,\, a(j_{m+1}\,,\, t_{m+1}\,,\, a(\cdots a(j_p\,,\, t_p\,,\, \xi_n)\cdots)))\cdots)))$$

and θ_m is used to represent

$$a(j_m\,,\, t_m\,,\, a(j_{m+1}\,,\, t_{m+1}\,,\, a(\cdots a(j_p\,,\, t_p\,,\, \xi_n)\cdots))).$$

```
state procedure interchange(s, m);
  value s, m;  state s;  integer m;
  begin comment  this procedure constructs a proof of ⊢s ≡ s′ where s′ is like s, except that
     the mth and (m + 1)-st assignments are interchanged. s′ is returned as the value of the
     procedure. Within s, we have j_m ≠_E j_{m+1} ;
     state s′;
     proofstep(θ_m ≡ a(j_{m+1} , t_{m+1} , a(j_m , t_m , θ_{m+2})), A3);
     s′ ← a(j_1 , t_1 , a(··· a(j_{m−1} , t_{m−1} , a(j_{m+1} , t_{m+1} , a(j_m , t_m , θ_{m+2})))···));
     proofstep(s ≡ s′, R4);
     interchange ← s′
  end procedure interchange;
state procedure deleteduplicate(s, m);
  value s, m;  state s;  integer m;
  begin comment this procedure constructs a proof of ⊢s ≡ s′ where s′ is the same as s except
     that the (m + 1)-st assignment (a duplicate one) is deleted. Within s, we have j_m =_E j_{m+1};
     state s′;
     proofstep(θ_m ≡ a(j_m , t_m , θ_{m+2}), A4);
     s′ ← a(j_1 , t_1 , a(··· a(j_m , t_m , θ_{m+2})···));
     proofstep(s ≡ s′, R4);
     deleteduplicate ← s′
  end procedure deleteduplicate;
state procedure deletevacuous(s, m);
  value s, m;  state s;  integer m;
  begin comment this procedure constructs a proof of ⊢s ≡ s′ where s′ is the same as s except
     that the mth assignment (a vacuous one) is deleted. s′ is returned as the value of the
     procedure. Within s, we have t_m =_E c(j_m , ξ_n) and j_1 , ··· , j_p all distinct;
     state s;
     term t;
     t ← reducetotally(c(j_m , θ_{m+1}));
     comment  since the j_m , j_{m+1} , ··· , j_p are distinct, t is now c(j_m , ξ_n) and a proof
       of ⊢c(j_m , θ_{m+1}) = c(j_m , ξ_n) has been constructed;
     proofstep(a(j_m , c(j_m , θ_{m+1}), θ_{m+1}) ≡ θ_{m+1} , A5);
     proofstep(a(j_m , c(j_m , ξ_n), θ_{m+1}) ≡ θ_{m+1} , R2);
     comment  this last step states that ⊢θ_m ≡ θ_{m+1} ;
     s′ ← a(j_1 , t_1 , a(··· a(j_{m−1} , t_{m−1} , θ_{m+1})···));
     proofstep(s ≡ s′, R4);
     deletevacuous ← s′
  end procedure deletevacuous;
```

In the following procedure, the argument s is of the form

$$a(i_{nk_1}\,,\, t_1\,,\, a(i_{nk_2}\,,\, t_2\,,\, a(\cdots a(i_{nk_p}\,,\, t_p\,,\, \xi_n)\cdots))).$$

```
state procedure canonical(s);  value s;  state s;
 begin comment this procedure constructs a proof of ⊢s ≡ s' where s' is the canonical form
   of s. s' is returned as the value of the procedure;
  state s';
  integer j, m;
  proofstep(s ≡ s, P2);
  s' ← s;
  comment  first we do a simple sort on the second subscript of the index letters in s and as
    assignments are rearranged, duplicates are deleted. Proofs of these manipulations are
    contructed;
  for m ← 1 step 1 until p−1 do
  for j ← 1 step 1 until p−m do
  if k_j = k_{j+1} then
    begin
      s' ← deleteduplicate(s', j);
      proofstep(s ≡ s', R4);
      p ← p − 1
    end
  else if k_j > k_{j+1} then
    begin
      s' ← interchange(s', j);
      proofstep(s ≡ s', R4)
    end;
  comment  second, we convert all of the terms t_1, ···, t_p into totally simple terms and
    construct proofs of the conversions;
  for j ← 1 step 1 until p do
    begin
      s' ← a(i_{nk_1}, t_1, a(··· a(i_{nk_j}, reducetotally (t_j), a(··· a(i_{nk_p}, t_p, ξ_n)···))···));
      proofstep(s ≡ s', R2)
    end;
  comment  third, we delete all vacuous assignments which reassign to a variable the
    quantity already there and construct proofs of the deletions. Note that the i_{nk_1}, ···, i_{nk_p}
    are now distinct;
  for j ← 1 step 1 until p do
  if t_j =_E c(i_{nk_j}, ξ_n) then
    begin
      s' ← deletevacuous(s', j);
      proofstep(s ≡ s', R4);
      p ← p − 1
    end;
  canonical ← s'
 end procedure canonical;
```

Consider now any wf $\mathfrak{A}$ of the form $\theta_1 \equiv \theta_2$ where $\models \mathfrak{A}$ and θ_1 and θ_2 are n-states for some $n = 1, 2, \cdots$. First construct proofs of $\theta_1 \equiv \theta_1^*$ and $\theta_2 \equiv \theta_2^*$ using procedure *canonical*(s) so that both θ_1^* and θ_2^* are in canonical form.

From **T3** we then have that $\models \theta_1 \equiv \theta_1^*$ and $\models \theta_2 \equiv \theta_2^*$. But we are given that $\models \theta_1 \equiv \theta_2$; from Lemma 3 we then obtain $\models \theta_1^* \equiv \theta_2^*$ which by definition means that $\models c(i_{nj}, \theta_1^*) = c(i_{nj}, \theta_2^*)$ for $j = 1, 2, \cdots$.

Suppose that

$$\theta_1^* =_E a(i_{nk_1}, t_1, a(\cdots a(i_{nk_p}, t_p, \xi_n)\cdots)),$$

$$\theta_2^* =_E a(i_{nl_1}, u_1, a(\cdots a(i_{nl_q}, u_q, \xi_n)\cdots)),$$

and let

$$I_1 = \{i_{nk_1}, i_{nk_2}, \cdots, i_{nk_p}\},$$

$$I_2 = \{i_{nl_1}, i_{nl_2}, \cdots, i_{nl_q}\}.$$

For any specific $i \notin I_1$, *reducetotally*$(c(i, \theta_1^*))$ will construct a proof of $\vdash c(i, \theta_1^*) = c(i, \xi_n)$. Now either $i \in I_2$ or $i \notin I_2$. Suppose for the moment that $i \in I_2$ and in fact $i =_E i_{nl_r}$ for some r, $1 \leq r \leq q$. Then *reducetotally*$(c(i, \theta_2^*))$ will construct a proof of $\vdash c(i, \theta_2^*) = u_r$ where $u_r \neq_E c(i, \xi_n)$ since θ_2^* is in canonical form.

These results and **T1** give $\models c(i, \theta_1^*) = c(i, \xi_n)$ and $\models c(i, \theta_2^*) = u_r$, and since we have shown above that $\models c(i_{nj}, \theta_1^*) = c(i_{nj}, \theta_2^*)$ for $j = 1, 2, \cdots,$ then certainly $\models c(i, \theta_1^*) = c(i, \theta_2^*)$ where $i =_E i_{nl_r}$. Then Lemma 2 gives $\models u_r = c(i, \xi_n)$. But, since θ_2^* is in canonical form, u_r is a totally simple term as is $c(i, \xi_n)$. So Lemma 1 gives $u_r =_E c(i, \xi_n)$, a contradiction.

Thus, the supposition that $i \in I_2$ leads to a contradiction. So if $i \notin I_1$ then $i \notin I_2$, or alternatively, if $i \in I_2$ then $i \in I_1$. An argument symmetric to the one above gives that if $i \in I_1$ then $i \in I_2$ so that combined, these results give $I_1 = I_2$. Since θ_1^* and θ_2^* are in canonical form, both I_1 and I_2 have elements ordered on the second subscript letter. Thus, $i_{nk_j} =_E i_{nl_j}$ for $j = 1, 2, \cdots, p$.

Now consider any specific $i \in I_1$, say $i =_E i_{nk_r}$ where $1 \leq r \leq p$. Then *reducetotally*$(c(i, \theta_1^*))$ will construct a proof of $\vdash c(i, \theta_1^*) = t_r$ and **T1** gives $\models c(i, \theta_1^*) = t_r$. Similarly, we obtain $\models c(i, \theta_2^*) = u_r$. But we know $\models c(i, \theta_1^*) = c(i, \theta_2^*)$ so that Lemma 2 gives $\models t_r = u_r$. Since t_r and u_r are totally simple terms, we have by Lemma 1 that $t_r =_E u_r$. Since this is true for any $i =_E i_{nk_r}$, then $t_j =_E u_j$ for $j = 1, 2, \cdots, p$.

An examination of the canonical representation of θ_1^* and θ_2^* reveals that the results obtained in the preceding two paragraphs for the index letters and terms in θ_1^* and θ_2^* give $\theta_1^* =_E \theta_2^*$. That is, θ_1^* and θ_2^* are the same n-state, symbol for symbol. That being so, **P2** gives $\vdash \theta_1^* \equiv \theta_2^*$. But the proof constructions gave $\vdash \theta_1 \equiv \theta_1^*$ and $\vdash \theta_2 \equiv \theta_2^*$. Using **R4** on $\vdash \theta_1 \equiv \theta_1^*$ and $\vdash \theta_1^* \equiv \theta_2^*$ gives $\vdash \theta_1 \equiv \theta_2^*$ which on applying **R3** gives $\vdash \theta_2^* \equiv \theta_1$. This last result together with $\vdash \theta_2 \equiv \theta_2^*$ and **R4** gives $\vdash \theta_2 \equiv \theta_1$ which on applying **R3** gives $\vdash \theta_1 \equiv \theta_2$, that is, $\vdash \mathfrak{A}$.

So we see that the formal theory $\mathfrak{J}$ is both complete and consistent; in addition the constructive proofs of completeness (in **T2** and **T4**) mean that $\mathfrak{J}$ is a decidable theory. These facts can now be utilized whenever state vectors, the functions a and c, and the axioms of $\mathfrak{J}$ are incorporated into other formalisms.

REFERENCES

1. McCarthy, J. Towards a mathematical science of computation. Proc. IFIP Congr. 62 (1962), 21–28.
2. Mendelson, E. *Introduction to Mathematical Logic*. Van Nostrand, Princeton, N. J., 1964.

RECEIVED APRIL, 1967; REVISED JULY, 1967