DECIDABLE PROPERTIES OF MONADIC FUNCTIONAL SCHEMAS

BY

EDWARD ASHCROFT
ZOHAR MANNA
AMIR PNEULI

JULY 1971

COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY
Abstract: We define a class of (monadic) functional schemas which properly includes 'IANOV' flowchart schemas. We show that the termination, divergence and freedom problems for functional schemas are decidable. Although it is possible to translate a large class of non-free functional schemas into equivalent free functional schemas, we show that this cannot be done in general. We show also that the equivalence problem for free functional schemas is decidable. Most of the results are obtained from well-known results in Formal Languages and Automata Theory.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183). A preliminary version of this paper was presented at the International Symposium on the Theory of Machines and Computation (Haifa, Israel), August 1971.

I. Monadic Functional Schemas

An alphabet $\mathcal{S}$ of a (monadic) functional schema $S$ consists of one individual variable $x$, a finite set of monadic function variables $\{f_i\}$ (with a designated initial function variable $f_0$), a finite set of monadic function constants $\{t_i\}$, and a finite set of monadic predicate constants $\{p_i\}$. Note that individual constants are not allowed.

A term over $\mathcal{S}$ is any term in the normal sense constructed from the monadic function variables $\{f_i\}$, monadic function constants $\{t_i\}$ and the variable $x$, e.g., $f_1(f_2(f_0(f_2(x))))$. A conditional term over $\mathcal{S}$ is any finite expression of the form

$$\text{if } p_i(x) \text{ then } t_1 \text{ else } t_2,$$

where $p_i$ is any predicate constant of $\mathcal{S}$, and $t_1$ and $t_2$ are any terms or conditional terms over $\mathcal{S}$.

A definition of $f_i$ over $\mathcal{S}$ is of the form

$$f_i(x) \leftarrow \tau,$$

where $\tau$ is any term or conditional term over $\mathcal{S}$. A (monadic) functional schema $S$ (over an alphabet $\mathcal{S}$) consists of a finite set of definitions over $\mathcal{S}$, one for each function variable $f_i$ in $\mathcal{S}$. Whenever the special function variable $f_m$ is used, its definition is considered to be $f_m(x) \leftarrow f_m(x)$. This definition is usually omitted.

Example 1: Let us consider the functional schema $S_1$:

$$f_0(x) \leftarrow \text{if } p_1(x) \text{ then } f_1(x) \text{ else } f_1(f_1(x))$$

$$f_1(x) \leftarrow \text{if } p_2(x) \text{ then } f_0(f_2(x)) \text{ else } f_1(x).$$

Since we are using a very restricted alphabet, parentheses and the individual variable $x$ may be omitted without causing any confusion. Therefore the functional schema $S_1$ can be rewritten as:

$$f_0 \leftarrow \text{if } p_1 \text{ then } f_2 \text{ else } f_1 f_1$$

$$f_1 \leftarrow \text{if } p \text{ then } f_0 f_2 \text{ else } f_1,$$

where $I$ stands for the 'identity function'.

An interpretation $I$ of a functional schema $S$ consists of:

1. a non-empty set of elements $D$ (called the domain),
2. an element $t_0$ of $D$ used as the initial value of $x$, and
3. assignments to the constants of $\mathcal{S}$:

(i) a total monadic function (from $D$ into $D$) for each function constant $f_i$, and

(ii) a total monadic predicate (from $D$ into $\{T,F\}$) for each predicate constant $p_i$.

It is worth noting that most of the results in this paper would be trivial if we did not allow the 'identity function'.
For a given interpretation \( \mathfrak{d} \), the pair \((S, \mathfrak{d})\), called a functional program, can be computed by evaluating \( F_0 \) with input \( t_0 \) in the usual way (see McCarthy [1963]). The computation either (i) terminates yielding an element of \( D \) denoted by \( \text{val}(S, \mathfrak{d}) \), or (ii) diverges (i.e., does not terminate) in which case \( \text{val}(S, \mathfrak{d}) \) is said to be undefined. The method of computation is described more fully later for special types of interpretations called 'Herbrand interpretations'.

A functional schema \( S \) is said to terminate/diverge if for every interpretation \( \mathfrak{d} \), \( \text{val}(S, \mathfrak{d}) \) is defined/undefined. Two functional schemas \( S_1 \) and \( S_2 \) are said to be equivalent if for every interpretation \( \mathfrak{d} \), either both \( \text{val}(S_1, \mathfrak{d}) \) and \( \text{val}(S_2, \mathfrak{d}) \) are undefined or both are defined and \( \text{val}(S_1, \mathfrak{d}) = \text{val}(S_2, \mathfrak{d}) \).

The same class of functional schemas has been discussed by DeBakker and Scott [1969]. It is straightforward to show that every functional schema in which any term contains at most one function variable, occurring on the left-hand side of the term (as in Example 1 above), can be translated to an equivalent 'Ianov' flowchart schema (Ianov [1960], see also Rutledge [1964]). However, such simple functional schemas as
\[
F_0 = \begin{cases} 
\text{if } p \text{ then } i \text{ else } f_1 F_0 f_2 
\end{cases}
\]
have no equivalent Ianov flowchart schema. Hence, the results in this paper generalize known results about Ianov flowchart schemas.

II. Herbrand Interpretations

A Herbrand interpretation \( \mathfrak{d}^* \) of a functional schema \( S \) consists of:

1. The domain \( D^* \), which is the set of all expressions "\( \tau_c \)" where \( \tau_c \) is a constant term constructed from the individual variable \( x \) and the function constants \( f_i \) of \( S \), e.g., "\( x \)", "\( f_1(x) \)", "\( f_2(f_1(x)) \);"
2. The expression "\( x \) in \( D^* \) used as the initial value of \( x \);
3. Assignments to the constants of \( D^* \):
   
   (i) For every function constant \( f_i \) in \( D^* \), we assign the total function mapping every expression "\( \tau_c \) in \( D^* \) into the expression "\( f_i(\tau_c) \) in \( D^* \);
   
   (ii) For every predicate constant \( p_j \) in \( D^* \), we assign same total predicate over \( D^* \), i.e., for every "\( \tau_c \) in \( D^* \) the value of \( p_j(\tau_c) \) is either T or F.

Note that all Herbrand interpretations of a given functional schema differ only in the assignments to the predicate constants. Henceforth we omit the quotation marks whenever this causes no confusion.

The computation of \((S, \mathfrak{d}^*)\) is best described by a (finite or infinite) sequence of terms \( \mathfrak{a}_i \). The first term \( \mathfrak{a}_0 \) of the sequence is \( F_0(x) \). In general, suppose \( \mathfrak{a}_n \) \( (n \geq 0) \) contains the sub-term \( F_i(t) \) where \( t \in D^* \), i.e., \( F_i \) is the right-most function variable in \( \mathfrak{a}_n \). Then \( \mathfrak{a}_{n+1} \) is obtained by substituting a term \( \tau \) for \( F_i(t) \) where \( \tau \) is obtained as follows. First \( x \) is replaced by \( \tau \) in the definition of \( F_i \), and then the term \( \tau \) is chosen from this definition using the values of \( p_j(\tau_c) \) supplied by the interpretation of the predicate constants \( p_j \). The computation terminates as soon as we reach a constant term \( \tau_c \). Then \( \text{val}(S, \mathfrak{d}^*) = \tau_c \).

Herbrand interpretations are important because many properties of functional schemas can be described and proved just by considering Herbrand interpretations rather than all interpretations. For example,

(i) a functional schema terminates/diverges (under every interpretation) if and only if it terminates/diverges under every Herbrand interpretation;

(ii) two functional schemas are equivalent (under every interpretation) if and only if they are equivalent under every Herbrand interpretation.

Herbrand interpretations are identical to the 'free interpretations' of Luckham, Park and Paterson [1970].
III. Termination and Divergence of Functional Schemas

We show that

**Theorem 1.** It is decidable whether or not a functional schema terminates or diverges (for every interpretation).

**Proof:** The proof depends on the well-known results in Formal Language Theory that it is decidable for any context-free grammar whether or not all (rightmost) derivations are finite and whether or not all (rightmost) derivations are infinite.

For this purpose we associate with every functional schema $S$ a context-free grammar $G_S$ such that:

there is a one-to-one correspondence between the set of all computations of $S$ for all Herbrand interpretations, and the set of all (rightmost) derivations of $G_S$. Furthermore, a computation of $S$ diverges if and only if the corresponding derivation of $G_S$ is infinite.

Given a functional schema $S$ with $n$ predicates $p_1, p_2, \ldots, p_n$, function variables $\{F_i\}$ and function constants $\{f_i\}$, we define $G_S$ as follows:

1. The non-terminals are of the form  
   
   
   \[ \omega \cdot F_k \cdot \omega' \] 

   where $\omega$ is the set of all strings of length $n$ over $\{F, f\}$. The intuitive meaning of $\omega \cdot F_k \cdot \omega'$ is that it will generate all possible constant terms $\tau_c$ such that $\tau_c$ is computed by $S$ starting from $F_k(x)$, under some Herbrand interpretation for which the values of $p_1, \ldots, p_n$ for $x$ are $\omega$ and for $\tau_c$ are $\omega'$.

   In addition, we have a special initial non-terminal $A$.

   $G_S$ now clearly has the required properties. Q.E.D.
IV. **Functional Schemas in Standard Form**

We introduce now an interesting subclass of functional schemas which has a special syntactic form. This form is relevant to our later discussion.

A functional schema $S$ is said to be in standard form if

1. Every conditional term in $S$ is of the form
   
   $\text{if } p_j \text{ then } \tau_1 \text{ else } \tau_2$
   
   where $\tau_1$ and $\tau_2$ are distinct (i.e., there are no 'redundant' tests) and each of $\tau_1$ and $\tau_2$ is of one of the following forms:
   
   (a) a conditional term not containing $p_j$ (i.e., there are no 'redundant' terms),
   
   (b) $F = \top$, or
   
   (c) a term in which $F = \not F$ does not occur and the rightmost symbol is a function constant.

2. Every definition in $S$ (except for $F = \not F$) contains a conditional term.

3. No function variable in $S$ always diverges (i.e., no matter which function variable in $S$ is taken as the initial function, the schemas do not diverge), except for $F = \not F$ if $F = \not F \subseteq F = \not F$.

Note that our definition of standard form has the flavor of Greibach Normal Form in Formal Languages Theory.

The interest in this form arises from the fact that every functional schema can be effectively transformed to an equivalent functional schema in standard form. This is done easily by first recognizing all 'divergent' function variables (using the technique described in the previous section) and replacing all terms in which they occur by $F = \not F$.

Then we repeatedly replace every function variable occurring as the rightmost symbol of a term by its definition, applying straightforward simplifications, as illustrated in Example 3 below.

**Example 1:** The functional schema $S_2$ is

$$F_0 \leftarrow \text{if } p \text{ then } F_0 F_1 \text{ else } F_2$$
$$F_1 \leftarrow \text{if } p \text{ then } F_1 F_3 \text{ else } I,$$

is clearly in standard form.

**Example 2:** The functional schema $S_3$ where

$$F_0 \leftarrow \text{if } p \text{ then } F_1 F_3 \text{ else } F_2$$
$$F_1 \leftarrow \text{if } p \text{ then } F_1 F_3 \text{ else } F_4$$

is not in standard form because of the term $F_1 F_3$. This term can be removed by first replacing $F_1$ in $F_1 F_3$ by its definition to obtain:

$$F_0 \leftarrow \text{if } p \text{ then if } p \text{ then } F_1 F_3 \text{ else } F_2$$

and then simplifying it to

$$F_0 \leftarrow \text{if } p \text{ then } F_1 F_3 \text{ else } F_2.$$

Thus, we obtain the functional schema $S_3'$, which is clearly in standard form and equivalent to $S_3$:

$$F_0 \leftarrow \text{if } p \text{ then } F_1 F_3 \text{ else } F_2$$
$$F_1 \leftarrow \text{if } p \text{ then } F_1 F_3 \text{ else } F_4.$$
A functional schema S is said to be free if for every Herbrand interpretation \( \mathcal{J}^* \) of \( S \) the computation of \((S, \mathcal{J}^*)\) does not test 8 predicate with the same term (from \( D^* \)) more than once.

Examples:
1. The functional schema \( S_1 \) (Example 1 above) is clearly free.
2. The functional schema \( S_2 \) (Example 2 above) is not free, since there is a Herbrand interpretation \( \mathcal{J}^* \) for which the computation of \((S, \mathcal{J}^*)\) is of the form
   \[
   P_0(x) \rightarrow P_0(f_1(f_2(x))) \rightarrow P_0(f_1(x)) \rightarrow f_2(f_1(x)) \rightarrow \ldots .
   \]
   Predicate \( p \) tests term \( f_1(x) \) at both steps (a) and (b).
3. The functional schema \( S_3 \) (Example 3 above) is not free, since there is a Herbrand interpretation \( \mathcal{J}^* \) for which the computation of \((S, \mathcal{J}^*)\) is of the form
   \[
   P_0(x) \rightarrow f_1(P_1(x)) \rightarrow f_1(f_2(x)) \rightarrow \ldots .
   \]
   Predicate \( p \) tests term \( x \) at both steps (a) and (b).

There is a crucial difference between the non-freedom demonstrated in the functional schema \( S_2 \) and that of \( S_3 \). In Example 3 the non-freedom results from the substitutions at step (b) for the function variable \( F_1 \) which resulted from the substitution at step (a). In Example 2 the non-freedom results from the substitution at step (b) for the function variable \( F_0 \) which was already present before step (8). Note that in this case the function variable \( F_2 \) substituted for at step (8) produced the identity function. This will always be the case when non-freedom of the second type occurs.

Functional schemas in standard form cannot have non-freedom of the first type but may have non-freedom of the second type. Thus, by bringing a functional schema to standard form we always eliminate any non-freedom of the first type (but unfortunately preserving any non-freedom of the second type). Since the second type of non-freedom involves the identity function, it follows that any functional schema not containing \( I \) can be made free by putting it into standard form.

Although a large class of functional schemas containing \( I \) can still be translated to equivalent free functional schemas, this cannot be done in general as follows from

**Theorem 2**: The non-free schema \( S_4 \) where
\[
P_0 \triangleq \text{if } p \text{ then } F_0 \text{ else } I, \quad P_1 \triangleq \text{ if } q \text{ then } F_1 \text{ else } I .
\]
has no equivalent free functional schema.

Proof: Consider the following two families \( [\mathcal{J}_n^P] \) and \( [\mathcal{J}_n^Q] \), \( n \geq 1 \), of Herbrand interpretations, where
\[
\mathcal{J}_n^P = \begin{cases} 
  p(f^I(x)) = T \& \text{iff } i < n, \\
  q(f^I(x)) = T \& \text{iff } i \neq n,
\end{cases}
\]
\[
\mathcal{J}_n^Q = \begin{cases} 
  p(f^I(x)) = T \& \text{iff } i < n, \\
  q(f^I(x)) = T \& \text{for all } i.
\end{cases}
\]

Note that two corresponding interpretations \( \mathcal{J}_n^P \) and \( \mathcal{J}_n^Q \) differ by only one value, that of \( q(f^I(x)) \). It is clear that \( \text{val}(\mathcal{J}_n^P) = P(x) \) and \( \text{val}(\mathcal{J}_n^Q) = f^P(x) \) for all \( n \geq 1 \).
Suppose there exists a free functional schema $G$ in standard form that is equivalent to $S_k$. We shall derive a contradiction by showing that there exists a positive integer $n_0$ for which $\text{val}(G, n_0^2) \neq f^{n_0}(x)$.

Let $n$ be an arbitrary positive integer. If we apply interpretations $\alpha_n^m$ and $\gamma_n^m$ to $G$ we must reach in both cases a term of the form $\alpha_n^m(f(x))$, since for $f(x)$, $1 \leq n$, $G$ cannot distinguish between $\alpha_n^m$ and $\gamma_n^m$.

Since for $\alpha_n^m$ this term must produce $f^m(x)$, it follows that every symbol in $\alpha_n^m$ is a function variable that 'collapses' to identity for predicate values determined by $\alpha_n^m$ for the term $f^m(x)$. Thus, since $G$ is free and no predicate in $G$ is ever to occur in $S$, then for any computation under interpretation $\alpha_n^m$ the term $f^m(x)$ twice, the number of symbols in $\alpha_n^m$ cannot exceed the number of distinct predicate constants $\{p_1, \ldots, p_j\}$ in $G$.

Therefore, there must exist two distinct positive integers $n_1$ and $n_2$, $n_1 \neq n_2$, such that $\alpha_{n_1}^m$ is identical to $\alpha_{n_2}^m$. By definition of $\{\gamma_n^m\}$, the continuation of the computation of $\alpha_{n_1}^m(f(x))$ under $\gamma_n^m$ is identical to that of $\alpha_{n_2}^m(f(x))$ under $\gamma_n^m$. Therefore, since $n_1 \neq n_2$, either $\text{val}(G, \gamma_n^m) \neq f^{n_1}(x)$ or $\text{val}(G, \gamma_n^m) \neq f^{n_2}(x)$.

Contradiction.

Q.E.D.

Despite the above result, we still have

**Theorem 5**: It is decidable whether a functional schema is free or not.

**Proof (sketch)**: The proof consists of showing that a functional schema $S$ is non-free if and only if non-free $n$ occurs in any computation under any Herbrand interpretation within a number $K$ of steps, where $K$ depends only on $S$. The decision method is then to explore all the different ways in which computation can proceed for $K$ steps under any Herbrand interpretation. Since there are $8$ finite number of such ways it is possible to decide whether or not non-free will ever occur.

We can assume that no function variable in $S$ is simply defined by the term $I$, since such function variables can be removed without affecting freedom in any way.

If non-free occurs it will be of the first or the second types. It can easily be shown that if non-free of the second type is ever to occur in $S$, then for any Herbrand interpretation the functional schema must have a subcomputation of the form

$$
\ldots \rightarrow \tau F_m F_{m-1} \ldots F_1 \tau_c \rightarrow \tau F_m \ldots F_2 \tau_c \rightarrow \ldots \rightarrow \tau F_m \tau_c \rightarrow \ldots
$$

where $\tau_c$ is a constant term and $m \leq n-1$ ($n$ is the number of distinct predicate constants in $\gamma_n^m$). Non-free occurs between steps (a) and (b), i.e., at (b) same predicate constant is testing $\tau_c$ which also tested $\tau_c$ at step (a). Note that $F_{m-1}, \ldots, F_{m-1}$ all must collapse to identity for $\tau_c$ under this interpretation.

Let us assume that the earliest occurrence of non-free, in any computation, is of the second type, and results from a computation of $\tau F_m \ldots F_1 \tau_c$ as above. If it takes $J$ steps to get $\tau F_m \ldots F_1 \tau_c$, then all computations of $S$ are free for at least their first $J$ steps. These initial computations therefore result from all possible substitutions, analogously to the rightmost derivations of a grammar. We can use the following easily proved result of Formal Language Theory to show that $J$ is bounded.

In any context free grammar, let $S \rightarrow a$ be the shortest rightmost derivation of a sentential form containing a given substring $a$ (of terminals and non-terminals) of length $m$, with some terminal string on its right (i.e., is a terminal string). Then this derivation has no more than $m$ steps where:

- $I$ is the number of non-terminals in $U$,
- $M$ is the maximal number of non-terminals on the right-hand side of any production of $G$,
- $N$ is the upper bound on the minimum number of steps needed for each non-terminal of $G$ to produce a terminal string, if any; (i.e., every non-terminal of $G$ can generate $8$ terminal strings if at all, within $N$ Steps).
Similarly defining L, M, and N for schema S gives us 8 bound on J depending only on S and M. Since m ≤ n1, we have a bound K on the first occurrence of non-freedom if such an occurrence is of the second type. It can be easily shown that any non-freedom of the first type must also be discovered within K steps, and hence the method of exploring the initial segments of all possible computations can decide whether non-freedom (of any type) is present.

Q.E.D.

VI. Equivalence of Free Functional Schemas

Theorem 4: It is decidable whether or not two free functional schemas are equivalent (for every interpretation).

Proof: Suppose we are given two free functional schemas $S$ and $S'$. Since the translation of functional schemas into standard form never introduces new non-freedom, we can assume without loss of generality that $S$ and $S'$ are in standard form.

We construct a Deterministic Push-Down Automaton (DPDA) $A$, whose input tape is a representation of some Herbrand interpretation and which simulates the computation of $S$ and $S'$ under this interpretation. $A$ accepts an input tape if and only if $S$ and $S'$ are inequivalent under the corresponding interpretation. Since it is decidable whether or not the language accepted by a DPDA is empty (see, for example, Hopcroft and Ullman [1969]), it follows that the equivalence problem for free functional schemas is decidable.

The construction of $A$ makes use of ideas developed by Rosenkrantz and Stearns [1970] (see also Korenjak and Hopcroft [1966]).

Suppose $S$ and $S'$ are $n$ predicate constants $p_1, p_2, \ldots, p_n$. The input alphabet consists of words of length $n$ over $\{T,F\}$ where we intend that when $A$ reads such a word (input symbol), the $i$-th letter denotes the current truth value of $p_i$. This requires some more explanation.

Each step in the computation sequence of a schema for a given interpretation $J$ consists of taking the current term, $\tau_{F_J^C}$, say, and substituting for $F_J$ according to the truth values of the predicates applied to $C$. Suppose this substitutes some term $\tau_{F_J^k}$ for $F_J$. Then, at the next step, to substitute for $F_J$, we need only know the truth values of the predicates applied to $F_J^k$, i.e., the current truth values. At each step we need only know the current truth values, for example in this case the truth values of the predicates for $F_J^k$, $F_J \neq F_J^k$, will not affect the computation in any way. Each interpretation, by specifying 8 computation, has a corresponding sequence of n-tuples over $\{T,F\}$; and vice-versa, each sequence of n-tuples over $\{T,F\}$, by specifying a computation, gives the truth values of the predicates for certain constant terms and therefore indicate a set of interpretations, all giving this computation. Hence we can represent interpretations by sequences of n-tuples over $\{T,F\}$, which is what we use as input tapes to our automaton $A$.

To simulate the joint action of $S$ and $S'$ for a given Herbrand interpretation, we let $A$ have a two-track push-down stack. Each track will hold 8 modified version of the current term in the computation sequence of the corresponding schema under the given interpretation.

The modification of the computation terms is such that if $S$ and $S'$ are equivalent, both tracks are of the same length during corresponding computations of $S$ and $S'$. This enables us to put both tracks in a single push-down stack. To understand this modification we introduce the notion of the 'thickness' $T(\tau)$ of a term $\tau$ (that does not contain $F_m$), namely the length of the shortest possible constant term computable from $\tau$ for some Herbrand interpretation. For free schemas, $T(\tau_{F_1^C}) = T(\tau_{F_2^C}) = T(\tau_{F_3^C})$, since no decision made while computing $\tau_{F_1}$ may affect the freedom of choice in computing $\tau_{F_2}$. The required modification of a term to give its stack representation is to make $T(\tau_{F_1})$ copies of each function variable $F_1$. Thus the length of stack representing term $\tau$ is $T(\tau)$. Note that if the corresponding terms in the computation of two equivalent
functional schemas are $\tau_1$ and $\tau_2$, then $T(\tau_1) = T(\tau_2)$; so the modified terms have the required property. To erase a function variable $F_i$ from such a stack, the automaton $A$ will in fact erase $T(F_i)$ copies of $F_i$, which is a feasible action of a multi-state DPDA.

The only problem that may arise is that it is possible that $T(F_i) = 0$, which is the case when the definition of $F_i$ contains $I$. In this case we would like to add $F_i$ to the preceding stack position and hence not increase the length of the stack. Note that if, for the next function variable $F_j$, we have again $T(F_j) = 0$, $F_j$ will also be added to the same stack position; and so on. However, for free functional schemas the number of function variables we have to add to the same stack position will never exceed the number $n$ of predicate symbols. This is because there can never be more than $n$ successive function variables collapsing to $I$ (otherwise some predicate would be tested more than once on the same term). We therefore can resolve the above difficulty by making each position in a track wide enough to hold $n+1$ ordered symbols. Then symbols of thickness zero are written on the preceding position, after the other symbols already residing there. In order to make no exception for the bottom position of the stack, we may agree to test the equivalence of $F_0$ and $F_0'$ instead of $F_0$ and $F_0$, .

The behavior of the DPDA $A$ is as follows.

For each input symbol, $A$ simulates all the possible actions of $S$ and $S'$. If the topmost letter of the corresponding track is a function constant, no change is made. Otherwise, it must be a function variable $F_i$ and we modify the top of that track according to the term in the definition of $F_i$ selected by the current input symbol. These actions will terminate either with some new stack-track with a function constant at the top or $F_i$ will be encountered. The crucial point is that for free functional schemas in standard form, the variation that occurs during these actions in the stack is bounded and can be temporarily remembered in a finite control until both operations for $S$ and $S'$ are completed.

Before moving to the next input letter, $A$ now proceeds as follows:

(i) If $F_0$ is encountered for both tracks, we pass to a rejecting state, (i.e., $S$ and $S'$ are equivalent under the given interpretation).

(ii) If we encounter $F_0$ for one track and not in the other, we pass to an accepting state (i.e., $S$ and $S'$ are inequivalent for some Herbrand interpretation under which the other track goes to a constant term).

(iii) If the two tracks are not of the same length, we pass to an accepting state (i.e., $S$ and $S'$ are inequivalent for some Herbrand interpretation under which the shorter track produces its shortest constant term).

(iv) If the two function constants at the top of the two tracks are different, we pass to an accepting state.

(v) Otherwise, (i.e., same length tracks with the same topmost function constant) we remove the topmost letters. If both stacks are still non-empty, we move to the next input symbol, otherwise both tracks are empty and we pass to a rejecting state.

Thus the two given free functional schemas $S$ and $S'$ are equivalent if and only if the DPDA $A$ accepts no input tapes, which is a known decidable problem.

Q.E.D.

**Example 1:** Consider the following two free functional schemas:

\[
\begin{align*}
& F_0 \leftarrow F_0 F_1 F_2 \\
& F_1 \leftarrow \text{if } F_0 \text{ then } F_1 F_2 \text{ else } I \\
& F_2 \leftarrow \text{if } F_2 \text{ then } F_2 F_2 \text{ else } I \\
& F_3 \leftarrow \text{if } F_0 \text{ then } F_3 F_1 F_2 \text{ else } I \\
& F_4 \leftarrow \text{if } F_1 \text{ then } F_0 F_2 \text{ else } F_3
\end{align*}
\]
Here, $P \neq 1$, and $F_1$, $F_2$, $F_3$, and $F_4$ are as in $S_5$.

This implies, for example, that the terms $F_0 F_2 f_2$, $F_4 f_2$, and $f_3 F_3 f_2$ would be stacked, respectively, as follows:

- $f_2$ $f_2$ $f_2$
- $F_4$ $F_4 F_2$
- $f_3 F_3 F_1$
- $F_0$ $F_0$

We illustrate now the behavior of the DPDA $A$ simulating the joint operation of $S_5$ and $S_6$ for any input tape starting with $(T, T)$, $(F, F)$, $(T, T)$, ... .

Since the input element indicates that both $p_1$ and $p_2$ are false at step (3), we have by definition that $F_1 = F_2 = F_3 = 1$ and $F_4 = F_3$. Therefore both terms $f_3 F_3 F_1$ and $F_4 F_2$ reduce to $f_3$. Note that we shall get the same sequence of stacks for any input tape for which the second element is $(F, F)$.

Acknowledgment: We are indebted to Mike Paterson for his critical reading of the manuscript and subsequent helpful suggestions. Mike proved independently that the equivalence problem is decidable for the class of all schemas without $I$.

References


