JOURNAL OF COMPUTER AND SYSTEM SCIENCES 6, 88–101 (1972)

On Program Schemata Equivalence

V. E. ITKIN

Computing Center of the Siberian Branch of the USSR Academy of Sciences, Novosibirsk 90, USSR

AND

Z. ZWINOGRODZKI

*Department of Logic, Institute of Philosophy and Sociology of the Polish Academy of Sciences, Warsaw, Poland**

Received September 18, 1970

Some problems related to the class of program schemata introduced by R. M. Karp and R. M. Miller in paper [1] are solved in this paper.

The method of the proof implemented in the paper allows us to obtain a negative solution of the existence problem of an algorithm for equivalence recognition for a wide class of program schemata which can be reduced to the above class, and for a wide class of equivalence definitions.

INTRODUCTION

Investigation of program properties which are invariant with respect to equivalence transformations of programs (when a definition is given) is of great interest for programming practice. In this connection a general "equivalence problem" arises, i.e., the problem of existence of an algorithm which for two arbitrary programs recognizes if they are equivalent or not (see survey paper [2]).

The papers concerned with this problem may be divided into two classes. Concrete *programs*, in which the operators and logical conditions are general recursive functions, are considered in the first class. A partial recursive function, which transforms the input data into the output values produced by a program execution, is associated with the program. Two programs are *equivalent* if the partial recursive functions associated with these programs in such a way are equal. The equivalence problem in this case

* Now at Computing Center of the Siberian Branch of the USSR Academy of Sciences, Novosibirsk 90, USSR.

is unsolvable [3]. The equivalence problems in the cases of different restrictions placed on the class of considered programs are investigated in [4].

Program schemata, in which some abstract symbols are used instead of concrete functions and logical conditions, are considred in the second class of papers. A concept of *interpretation* of the program schemata is introduced here. The interpretation associates some concrete program with a schema. Two program schemata are *functionally equivalent* (this term is introduced in [5]) if they give rise to equivalent programs for every interpretation. Unsolvability of the functional equivalence problem for some classes of program schemata is proved in papers [5–7].

Other definitions of the program schemata equivalence are, in a sense, stronger than the functional equivalence (see [8–16]).

The definition of equivalence in [1] is of the same kind as in [8–16]. In [1], the authors prove the unsolvability of the equivalence problem for the widest class of program schemata treated in [1], as well as the solvability of this problem for a certain very narrow class of those program schemata. In contrast to the solvable cases obtained in [8–16], the equivalence problem in [1] appears to be unsolvable in general for "determinate schemata." This fact is established in the corollary to our Theorem 3.

The general definition of program schema introduced in [1] is given in our Section 1. In Section 2, this definition is compared with conventional definitions of program schemata in [10, 5, 7]. In Section 3, unsolvability of the problem of accessibility of states for automata which represent the program schemata is proved. Also the reasons for solvability or unsolvability of an equivalence problem are shown to depend on whether a given definition of equivalence uses or does not use the conception of "interpretation" and "a set of all interpretations."

The authors wish to express their appreciation of Prof. A. P. Ershov, V. E. Kotov and A. S. Narinyani for stimulating discussions. They also thank the referee for calling their attention to [19, 20], where analogous problems are treated. The reset idea used in our proof of Theorem 2 is also used in [20]. These papers were not known to us at the time of submission of this paper. Finally, we thank E. K. Blum for his personal assistance in the revision of this paper.

1. The Karp–Miller Program Schemata

A *K–M schema* (A Karp-Miller program schemata) $\mathbf{S} = (\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}, \mathbf{q_0}, \mathbf{t})$ is an automaton without outputs specified by a set of states $\mathbf{Q}$ (finite or infinite), an initial state $\mathbf{q_0} \in \mathbf{Q}$, an input alphabet $\mathbf{X}$ and a partial recursive transition function $\mathbf{t} : \mathbf{Q} \times \mathbf{X} \to \mathbf{Q}$.

Moreover:

1. U is a finite set of *operators*. A set $\mathbf{D(u)} \subset M$, whose elements are *domain loca-*

tions, and a set $\mathbf{R(u)} \subset M$, whose elements are *range locations*, are associated with any operator $\mathbf{u} \in \mathbf{U}$. The set $\mathbf{M}$ is called *memory*.

2. With any operator $\mathbf{u} \in \mathbf{U}$ a set $\mathbf{X_u} = \{\dot{\mathbf{u}}, \mathbf{u_1}, ..., \mathbf{u_{k(u)}}\}$ is associated and $\mathbf{X}$ is a sum of sets $\mathbf{X_u}$ for all $\mathbf{u} \in \mathbf{U}$. Intuitively $\dot{\mathbf{u}}$ denotes an "initiation" of the operator $\mathbf{u}$ and $\mathbf{u_i}$ ($\mathbf{i} = 1, ..., \mathbf{k(u)}$) denotes a "termination" of the operator $\mathbf{u}$.

3. We suppose that the function $\mathbf{t}$ is total on all the pairs containing the "termination" symbols.

In order to give an *interpretation* of a K–M schema, it is necessary to specify an initial content $\mathbf{c_0}$ of the memory $\mathbf{M}$ (i.e., to associate an integer number with every element of $\mathbf{M}$) and to associate with every operator $\mathbf{u} \in \mathbf{U}$ general recursive functions $\mathbf{f_u} : \mathbf{C^r} \to \mathbf{C^l}$, $\mathbf{g_u} : \mathbf{C^r} \to \{\mathbf{u_1}, ..., \mathbf{u_{k(u)}}\}$ ($\mathbf{C}$ is the set of all integer numbers; $\mathbf{r}$, $\mathbf{l}$ are the numbers of elements in $\mathbf{D(u)}$ and $\mathbf{R(u)}$, respectively; if $\mathbf{R(u)} = \varnothing$, then only $\mathbf{g_u}$ is associated with $\mathbf{u}$).

A word $\mathbf{N} = \mathbf{x_1} \cdots \mathbf{x_n} \cdots$ over the alphabet $\mathbf{X}$ ($\mathbf{N}$ can be finite, infinite or empty) is *confirmed* by an interpretation $\mathbf{J}$ of the K–M schema $\mathbf{S} = (\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}, \mathbf{q_0}, \mathbf{t})$ (then $\mathbf{N}$ is called a *computation* for $\mathbf{S}$) if it can be generated in the following way:

i. We fix an initial content $\mathbf{c_0}$ of the memory $\mathbf{M}$, $\mathbf{c_0}$ being defined by $\mathbf{J}$, and an initial state $q_0 \in Q$.

ii. Let a word $\mathbf{N}_n = \mathbf{x_1} \cdots \mathbf{x_n}$ ($\mathbf{n} \geqslant 0$) be built and the automaton $\mathbf{S}$ be in a state $\mathbf{q}$. Then, if $\mathbf{t(q, \dot{u})}$ is not defined for all $\mathbf{u} \in \mathbf{U}$ and the number of "initiations" of every operator $\mathbf{u}$ is equal to the number of its "terminations" in $\mathbf{N_n}$, then $\mathbf{N} = \mathbf{N_n}$ is the word we need and the process is finished. We may suppose $\mathbf{x_{n+1}} = \dot{\mathbf{u}}$ iff $\mathbf{t(q, \dot{u})}$ is defined. If $\mathbf{x_{n+1}} = \dot{\mathbf{u}}$, then the automaton $\mathbf{S}$ turns into the state $\mathbf{t(q, \dot{u})}$. We may suppose $\mathbf{x_{n+1}} = \mathbf{u_i}$ iff the number of "initiations" of an operator $\mathbf{u}$ is more than the number $\mathbf{s}$ of its "terminations" and $\mathbf{g_n(c)} = \mathbf{u_i}$ (where $\mathbf{c}$ is the content of the domain locations $\mathbf{D(u)}$ just before the $(\mathbf{s} + 1)$–th "initiation" of the operator $\mathbf{u}$ in $\mathbf{N_n}$). If we suppose that $\mathbf{x_{n+1}} = \mathbf{u_i}$, then the automaton $\mathbf{S}$ turns into the state $\mathbf{t(q, u_i)}$ and the current content of the range locations $\mathbf{R(u)}$ is changed in accordance with the function $\mathbf{f_u}$, the values of its arguments being taken out of the c, where c is again the contents of $D(u)$ before the $(\mathbf{s} + 1)$–st initiation of u in N_n.

iii. If for every prefix $\mathbf{N_n}$ of the word $\mathbf{N}$ the word $\mathbf{N_n x}$ satisfies the requirements i.–ii. ($\mathbf{n \geqslant k \geqslant 0}$, $\mathbf{x \in X}$), then we suppose that in $\mathbf{N}$ there exists $\mathbf{x_j}$ ($\mathbf{j > k}$) such that $\mathbf{x_j} = \mathbf{x}$. (*Finite delay property of* [1].)

The set off all the computations of $\mathbf{S}$ confirmed by $\mathbf{J}$ is denoted as $\mathbf{S(J)}$.

A K–M schema $\mathbf{S}$ is called *one-valued* if for every interpretation $\mathbf{J}$ the set $\mathbf{S(J)}$ involves not more than one computation (i.e., every interpretation confirms not more than one computation).

Two K–M schemata $\mathbf{S}_i = (\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}_i, \mathbf{q}_0{}^i, \mathbf{t}_i)_{i=1,2}$ are 0–*equivalent* iff for every interpretation $\mathbf{J}$ the intersection of sets $\mathbf{S}_1(\mathbf{J})$ and $\mathbf{S}_2(\mathbf{J})$ is not empty.

THEOREM 1. *There exists an algorithm which for an arbitrary* K–M *schema constructs a* 0*-equivalent one-valued* K–M *shcema.*

Proof. Let us build the K–M schema $\mathbf{S}' = (\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}', \mathbf{q}_0', \mathbf{t}')$ for an arbitrary K–M schema $\mathbf{S} = (\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}, \mathbf{q}_0, \mathbf{t})$, where $\mathbf{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^n\}$, in the following way:

1. $\mathbf{Q}' = \mathbf{Q} \times \{0, 1, \ldots, n\} \times \{0, 1\}$.

2. $\mathbf{q}_0' = (\mathbf{q}_0, 0, 0)$.

3. $\mathbf{t}'((\mathbf{q}, \mathbf{m}_1, \mathbf{m}_2), \dot{\mathbf{u}}^i)$, where $(\mathbf{q}, \mathbf{m}_1, \mathbf{m}_2) \in \mathbf{Q}'$ is defined iff the following conditions are fulfilled simultaneously:

3.1. $\mathbf{t}(\mathbf{q}, \mathbf{u}^i)$ is defined.

3.2. $\mathbf{m}_2 = 0$.

3.3. If there exists $\mathbf{k}$ such that $\mathbf{k} > \mathbf{m}_1$ and $\mathbf{t}(\mathbf{q}, \dot{\mathbf{u}}^k)$ is defined, then $\mathbf{i}$ is the least of such $\mathbf{k}$; if such $\mathbf{k}$ does not exist, then $\mathbf{i}$ is the least of such $\mathbf{j}$ ($\mathbf{j} = 1, \ldots, \mathbf{n}$) that $\mathbf{t}(\mathbf{q}, \dot{\mathbf{u}}^j)$ is defined.

4. If $\mathbf{t}'((\mathbf{q}, \mathbf{m}_1, \mathbf{m}_2), \dot{\mathbf{u}}^i)$ is defined, then it is equal to $(\mathbf{t}(\mathbf{q}, \dot{\mathbf{u}}^i), \mathbf{m}_1 + 1 \pmod{\mathbf{n}}, 1)$.

5. $\mathbf{t}'((\mathbf{q}, \mathbf{m}_1, \mathbf{m}_2), \mathbf{u}_j{}^i) = (\mathbf{t}(\mathbf{q}, \mathbf{u}_j{}^i), \mathbf{m}_1, 0)$.

We may say informally that the initiations and terminations of operators (when $\mathbf{m}_2$ equals 0 and 1 accordingly) occur in turn in the schema $\mathbf{S}$. In every moment of initiation, if some operators of the set $\{\mathbf{u}^{m_1+1}, \ldots, \mathbf{u}^n\}$ can be initiated in the K–M schema $\mathbf{S}$ by a proper state $\mathbf{q}$, then in $\mathbf{S}$ that state is initiated, which has the least order number. Otherwise such a "least" operator is sought in the whole set $\mathbf{U}$. This construction satisfies the condition of one-valued and satisfies the condition iii of the definition of a computation.

(This proof is a version of the proof of Theorem 4.10 in [1].)

2. FLOWCHARTS

Let us take a subclass $\mathscr{M}_0$ of one-valued K–M schemata $\mathbf{S}(\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}, \mathbf{q}_0, \mathbf{t})$ in which one of the following two conditions is true for every operator $\mathbf{u} \in \mathbf{U}$.

1. $\mathbf{u}$ has exactly one symbol of termination and $\mathbf{R}(\mathbf{u}) \neq \varnothing$. In this case we refer to $\mathbf{u}$ as a *transformer*.

2. $\mathbf{u}$ has exactly two symbols of termination (denoted by $+\mathbf{u}$ and $-\mathbf{u}$) and $\mathbf{R}(\mathbf{u}) = \varnothing$. In this case we refer to $\mathbf{u}$ as a *resolver*.

The K–M schemata of the class $\mathscr{M}_0$ are known usually as *flowcharts*. This definition

differs from the definitions in [5, 7, and 10] only in that there are sending operators, i.e., the operators $\mathbf{u}$ whose sets $\mathbf{D(u)}$ and $\mathbf{R(u)}$ contain only one element and $\mathbf{f_u}$ is an identity in all interpretations. The definition also differs from that in [1] for parallel flowchart.

The transition graph of an arbitrary K–M schema $\mathbf{S(M, U, X, Q, q_0, t)}$ is a graph whose vertices are in a one-to-one correspondence with the set of states $\mathbf{Q}$ and there exists an arc from $\mathbf{q}$ to $\mathbf{q'}$ assigned by a symbol $\mathbf{x(q, q' \in Q, x \in X)}$ iff $\mathbf{t(q, x) = q'}$. Sometimes some arcs of this graph are omitted for convenience.

In order to receive a clearer representation of the flowchart $\mathbf{S(M, U, X, Q, q_0, t)}$ one uses usually not a transition graph but a graph which can easily be obtained from the transition graph by transformation shown in Fig. 1. Here, $\mathbf{q}$ and $\mathbf{q'}$ are states in $\mathbf{Q}$; $\mathbf{i}$, $\mathbf{j}$ are those states to which the proper arcs lead, 1 is a set of arcs leading to vertex $\mathbf{q}$, $\mathbf{a}$ is an operator, and $\mathbf{p}$ is a resolver.

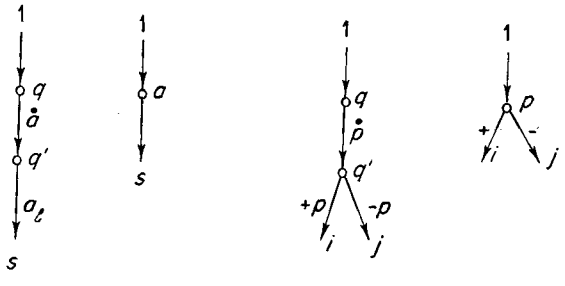

FIGURE 1

By a code of a one-valued arbitrary K–M schema $\mathbf{S(M, U, X, Q, q_0, t)}$ we mean a flowchart $\mathbf{S'(M, U, X, Q', q_0, t)}$ that will be defined by means of a following transformation of the transition graph of the K–M schema $\mathbf{S}$:

1. If $\mathbf{u \in U}$, $\mathbf{R(u) \neq \varnothing}$, $\mathbf{X_u = \{\dot{u}, u_1, ..., u_{k(u)}\}}$, where $k > 1$, then we replace every fragment of the transition graph of the K–M schema $\mathbf{S}$, which is similar to the fragment in Fig. 2 (on the left), by the corresponding fragment, which is similar to the fragment in Fig. 2 (on the right), where $\mathbf{a}$ is a transformer, $\mathbf{p^1, ..., p^{k(u)-1}}$ are resolvers which are not included in $\mathbf{U}$ and which encode the operator $\mathbf{u}$,

$$\mathbf{D(a) = D(p) = \cdots = D(p^{k(u)-1}) = D(u), \qquad R(a) = R(u).}$$

2. If $\mathbf{u \in U}$, $\mathbf{R(u) = \varnothing}$, $\mathbf{X_u = \{\dot{u}, u_1, ..., u_{k(u)}\}}$, where $\mathbf{k > 2}$, then all remains as in the previous point except the following: We do not introduce the transformer, and the arcs, assigned by $+\mathbf{p^1}$, $+\mathbf{p^2}, ..., +\mathbf{p^{k(u)-1}}$, $-\mathbf{p^{k(u)-1}}$, are directed to the vertices $1, ..., \mathbf{k(u)} - 1, \mathbf{k(u)}$ accordingly.

3. If $\mathbf{p(u) = \varnothing}$, $\mathbf{X_u = \{\dot{u}, u_1\}}$, then instead of $\mathbf{u}$ we introduce the resolver $\mathbf{p}$ such that $\mathbf{D(p) = D(u)}$, and in every fragment of the transition graph of the K–M

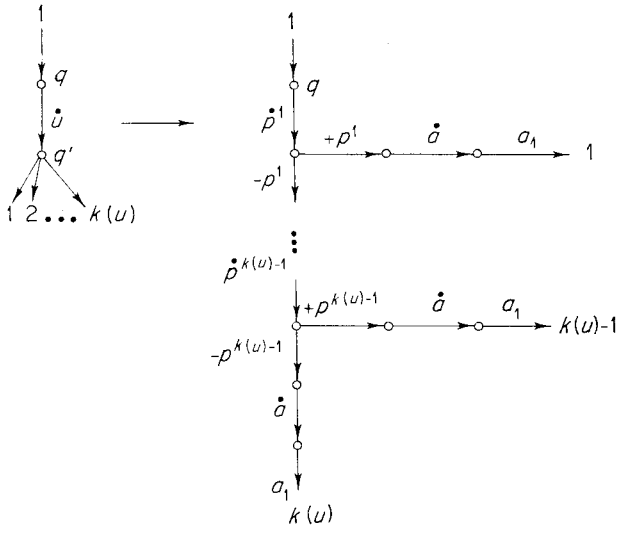

FIGURE 2

schema **S** we replace **u** by **p** and instead of one arc, assigned by the symbol $\mathbf{u}_1$, there appear two arcs, assigned by symbols $+\mathbf{p}$ and $-\mathbf{p}$, which lead to the same vertex.

4. In other respects the transition graphs of **S** and **S'** coincide.

The code **S'** of a K–M schema **S** has the following properties:

1. $\mathbf{Q}' \supset \mathbf{Q}$.

2. Let **L** be a one-to-one mapping which, for every element of **X**, brings in correspondence a word which belongs to **X'** in the following way:

2.1. If **u** satisfies the conditions of point 1 in the definition of the code, then we suppose

$$\mathbf{L(u)} = \dot{\mathbf{p}}^1, \qquad \mathbf{L(u_1)} = +\mathbf{p}^1\dot{\mathbf{a}}\mathbf{a}_1, \qquad \mathbf{L(u_2)} = -\mathbf{p}^1\dot{\mathbf{p}} + \mathbf{p}^2\dot{\mathbf{a}}\mathbf{a}_1, \ldots,$$

$$\mathbf{L(u_{k(u)-1})} = -\mathbf{p}^1\dot{\mathbf{p}}^2 - \mathbf{p}^2 \cdots \dot{\mathbf{p}}^{k(u)-1} + \mathbf{p}^{k(u)-1}\dot{\mathbf{a}}\mathbf{a}_1,$$

$$\mathbf{L(u_{k(u)})} = -\mathbf{p}^1\dot{\mathbf{p}}^2 - \mathbf{p}^2 \cdots \dot{\mathbf{p}}^{k(u)-1} - \mathbf{p}^{k(u)-1}\dot{\mathbf{a}}\mathbf{a}_1.$$

2.2. If **u** satisfies the conditions of point 2 in the definition of the code then all remains as in point 2.1 except the following: In the words $\mathbf{L(u_1)}, \ldots, \mathbf{L(u_{k(u)})}$ it is necessary to cut out the symbols of $\mathbf{X}_{\dot{\mathbf{a}}}$.

2.3. If **u** satisfies the conditions of point 3 in the definition of the code, then we suppose $\mathbf{L(u)} = \dot{\mathbf{p}}$, $\mathbf{L(u_1)} = +\mathbf{p}$ (**p** is mentioned in point 3 of the definition of the code.)

2.4. If $\mathbf{u}$ is a transformer or a resolver, then the mapping is an identity mapping on the symbols of $\mathbf{X}_a$.

Then the following statement is valid.

LEMMA (on coding). *A word* $\mathbf{x}_1 \cdots \mathbf{x}_n$ *over the alphabet* $\mathbf{X}$ *transits the state* $\mathbf{q}_0$ *of a one-valued automaton* $\mathbf{S}$ *into a state* $\mathbf{q} \in Q$ iff *the word* $\mathbf{L}(\mathbf{x}_1) \cdots \mathbf{L}(\mathbf{x}_n)$ *over the alphabet* $\mathbf{X}'$ *transits the initial state* $\mathbf{q}_0$ *of an automaton* $\mathbf{S}'$ *into the state* $\mathbf{q}$. (*It immediately follows from the definitions introduced above.*)

3. The Accessibility of a State in K–M Schemata

We assume the state $\mathbf{q} \in \mathbf{Q}$ of an arbitrary K–M schema $\mathbf{S}$ to be *accessible* if there exists a computation $\mathbf{x}_1 \cdots \mathbf{x}_i \cdots$ for such $\mathbf{S}$ at some $\mathbf{i} \geqslant 0$ we have

$$\tilde{\mathbf{t}}(\mathbf{x}_1 \cdots \mathbf{x}_i, \mathbf{q}_0) = \mathbf{q}.^1$$

Is there an algorithm recognizing the accessibility of an arbitrary state of an arbitrary K–M schema of the given set of these schemata? This is a problem of the state accessibility.

Consider the set $\mathscr{M}_1$, of finite one-valued K–M schemata $\mathbf{S}(\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}, \mathbf{q}_0, \mathbf{t})$, where

$$\mathbf{M} = \{\mathbf{r}_1, \mathbf{r}_2\}, \quad \mathbf{U} = \{\mathbf{u}, \mathbf{v}, \mathbf{w}\}, \quad \mathbf{D}(\mathbf{u}) = \mathbf{R}(\mathbf{u}) = \{\mathbf{r}_1\},$$

$$\mathbf{D}(\mathbf{v}) = \mathbf{R}(\mathbf{v}) = \{\mathbf{r}_2\}, \quad \mathbf{D}(\mathbf{w}) = \varnothing, \quad \mathbf{R}(\mathbf{w}) = \{\mathbf{r}_1, \mathbf{r}_2\},$$

$$\mathbf{X} = \{\dot{\mathbf{u}}, \mathbf{u}_1, \mathbf{u}_2, \dot{\mathbf{v}}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dot{\mathbf{w}}, \mathbf{w}_1\}.$$

THEOREM 2. *In* $\mathscr{M}_1$, *a set of finite one-valued* K–M *schemata, the accessibility problem of the state is unsolved.*

Proof. Let us prove that the question under discussion can be reduced to the "Post problem". Post proved that there is no algorithm which could recognize the following property for arbitrary words $\mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{y}_1', \ldots, \mathbf{y}_n'$ in the alphabet $\{\mathbf{v}_1, \mathbf{v}_2\}$ $(\mathbf{n} \geqslant 1)$:

(*) There is sequence $\mathbf{i}_1, \ldots, \mathbf{i}_p$ $(1 \leqslant \mathbf{i}_j \leqslant \mathbf{n}, 1 \leqslant \mathbf{j} \leqslant p, p \geqslant 1)$ such that

$$\mathbf{y}_{i_1} \cdots \mathbf{y}_{i_p} = \mathbf{y}'_{i_1} \cdots \mathbf{y}'_{i_p}.$$

On the ground of $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ let us determine a K–M schema

$$\mathbf{S}(\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}_y, \mathbf{q}_0^y, \mathbf{t}_y) = \mathbf{S}_y$$

[1] $\mathbf{t}(\mathbf{t}(\cdots \mathbf{t}(\mathbf{t}(\mathbf{x}_1, \mathbf{q}_0), \mathbf{x}_2) \cdots), \mathbf{x}_{i-1}), x_i) = \tilde{\mathbf{t}}(\mathbf{x}_1, \ldots, \mathbf{x}_i, \mathbf{q}_0)$.

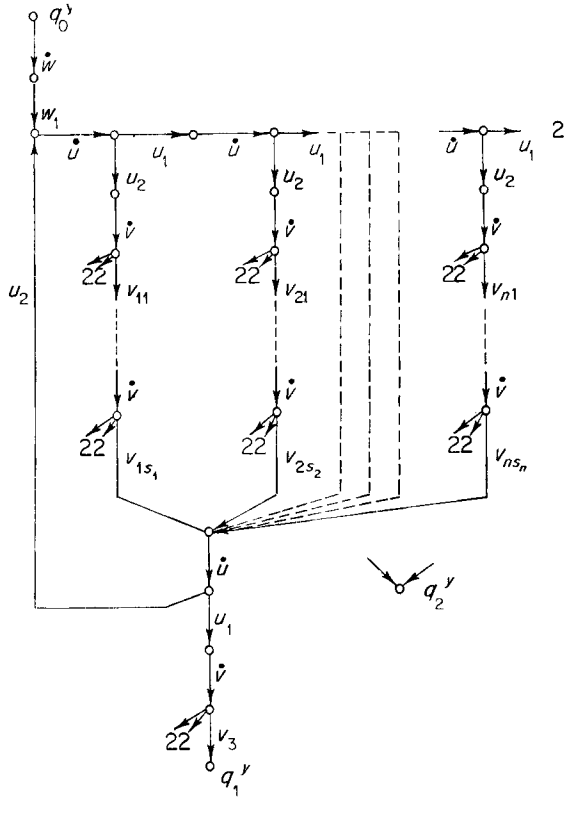

FIGURE 3

(it is defined in [1]). Its transition graph is shown in Fig. 3, the circles there correspond to different states of $\mathbf{Q_y}$; arcs, denoted by the index 2, are directed to the vertex $\mathbf{q_2^y}$. If an arc or a state denotation is of no importance for the proof, we omit it. Then we have the following:

$$\mathbf{y_1} = \mathbf{v_{11}} \cdots \mathbf{v_{1s_1}},$$
$$\cdots$$
$$\cdots$$
$$\mathbf{y_n} = \mathbf{v_{n1}} \cdots \mathbf{v_{ns_n}},$$

where $\mathbf{v_{i_j}} \in \{\mathbf{v_1}, \mathbf{v_2}\}$.

Analogously we determinate a K–M schema

$$\mathbf{S_{y'}} = (\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q_{y'}}, \mathbf{q_0^{y'}}, \mathbf{t_{y'}}) \qquad \text{for } \mathbf{y'} = (\mathbf{y_1'}, \ldots, \mathbf{y_n'}).$$

Then we construct a K–M schema

$$\mathbf{S}_{yy'} = (\mathbf{M}, \mathbf{U}, \mathbf{X}, \mathbf{Q}_y \cup \mathbf{Q}_{y'}, \mathbf{q}_0^y, \mathbf{t}_{yy'})$$

in the following way: "Join" the transition graph $\mathbf{S}_y$, assuming $q_1^y = q_0^{y'}$.

We show that the state $\mathbf{q}_1^{y'}$ is accessible in $\mathbf{S}_{yy'}$, iff the condition (*) is true.

Necessity. Let $\mathbf{q}_1^{y'}$ be accessible in $\mathbf{S}_{yy'}$ in a computation $\mathbf{N}$, confirmed by the interpretation $\mathbf{J}$. Then the words

$$\begin{aligned} \mathbf{N}_y &= \mathbf{w}_1\mathbf{u}_1^{i_1-1}\mathbf{u}_2\mathbf{y}_{i_1}\mathbf{u}_2 \cdots \mathbf{u}_2\mathbf{u}_1^{i_p-1}\mathbf{u}_2\mathbf{y}_{i_p}\mathbf{u}_1\mathbf{v}_3 = \mathbf{l}\mathbf{v}_3\,, \\ \mathbf{N}_{y'} &= \mathbf{w}_1\mathbf{u}_1^{i_1'-1}\mathbf{u}_2\mathbf{y}'_{i_1'}\mathbf{u}_2 \cdots \mathbf{u}_2\mathbf{u}_1^{i_{p'}'-1}\mathbf{u}_2\mathbf{y}'_{i_{p'}'}\mathbf{u}_1\mathbf{v}_3 = \mathbf{l}'\mathbf{v}_3 \end{aligned} \qquad (**)$$

are computations for $\mathbf{S}_y$ and $\mathbf{S}_{y'}$ at the interpretation $\mathbf{J}$ and $\mathbf{N} = \mathbf{l}\mathbf{v}_3\mathbf{l}'\mathbf{v}_3$ (here all initiation symbols are eliminated from computations to make them readable.) The fact that (*) is true is due to the following:

1. The subsequences of the "termination" symbols of the operators $\mathbf{u}$ and $\mathbf{v}$ do not depend on each other but only on an interpretation.

2. The "lossless" operator $\mathbf{w}$ is switched on two times in $\mathbf{N}$.

3. As the subsequence $\mathbf{u}_2\mathbf{u}_3$ in the words $\mathbf{l}$ and $\mathbf{l}'$ can appear only after a similar previous subsequence of the "termination" symbols of the operator $\mathbf{u}$, then these subsequences are equal, and it means that $\mathbf{p} = \mathbf{p}'$ and $\mathbf{i}_1 = \mathbf{i}_1'$,..., $\mathbf{i}_p = \mathbf{i}_p'$.

4. As the symbol $\mathbf{v}_3$ in the words $\mathbf{N}_y$ and $\mathbf{N}_{y'}$ can appear only after similar subsequences of the "termination" symbols of the operator $\mathbf{v}$, then

$$\mathbf{y}_{i_1} \cdots \mathbf{y}_{i_p} = \mathbf{y}'_{i_1'} \cdots \mathbf{y}'_{i_{p'}'}\,.$$

Sufficiency. Let the condition (*) be true. Let $\mathbf{J}$ be such an interpretation of K–M schemata $\mathbf{S}_y$, $\mathbf{S}_{y'}$, $\mathbf{S}_{yy'}$ that $\mathbf{N}_y$ is (**). We can get the sequence

$$\mathbf{w}_1\mathbf{u}_1^{i_1'-1}\mathbf{u}_2\mathbf{y}'_{i_1'}\mathbf{u}_2 \cdots \mathbf{u}_2\mathbf{u}_1^{i_{p'}'-1}\mathbf{u}_2\mathbf{y}'_{i_{p'}'}\mathbf{u}_1\mathbf{v}_3$$

due to (*) by a commutative of the symbols of $\mathbf{N}_y$ which does not destroy the subsequence of the "termination" symbols of either the operators $\mathbf{u}$ or the operator $\mathbf{v}$.

It easy to see that

1. Such a sequence of "termination" symbols is confirmed by the interpretation $\mathbf{J}$.

2. This sequence transforms $\mathbf{S}_{y'}$ into the state $\mathbf{q}_1^{y'}$.

Hence this sequence is a computation for $\mathbf{S}_{y'}$ confirmed by the interpretation $\mathbf{J}$. Thus it is already easy to derive that $\mathbf{q}_1^{y'}$ is accessible in $\mathbf{S}_{yy'}$ at $\mathbf{J}$.

COROLLARY. *In a set of finite one-valued flowcharts the problem of the state accessibility is unsolved. (As a result of Theorem* 2 *and Lemma on coding.)*

Remark. The construction of S_y and $S_{y'}$ follows the construction of the schemata $\mathscr{S}(X)$ in [1, pp. 175, 176] with inessential changes.

Note 1. The problem of recognition of the existence of a finite computation for an arbitrary finite one-valued K–M schema **S** is unsolved. (In fact, if $\mathbf{S}_{yy'}$ is transformed into $\mathbf{S}'_{yy'}$ so that in the state $\mathbf{q}_1^{y'}$ the computation is stopped but after the states $\mathbf{q}_2^{y'}$ and $\mathbf{q}_2^{y}$ it continues infinitely long, then the discussed question is again reduced to the "Post problem".)

Note 2. Unsolvability of the problem of the state accessibility for the set K–M schemata of $\mathbf{S}_{yy'}$ is due to the three reasons:

1. "Repeated copies" of the operators **u**, **v**, **w** in the transition graph $\mathbf{S}_{yy'}$.
2. Commutation of the operators **u** and **v**.
3. "Repeated sendings" of the operator **w**.

As to the repeated sendings, the problem of the state accessibility is unsolved without them in passing to the set of infinite one-valued K–M schemata, i.e., to such schemata for which Q is an infinite set.

The K–M schema $\tilde{\mathbf{S}}_{yy'}$ constructed by us is such that the state $(\mathbf{q}_1^{y}, \Lambda)$ of this schema is accessible iff the state $\mathbf{q}^{y}{}_1$ is accessible in $\mathbf{S}_{yy'}$. Thus the problem of accessibility of the states $(\mathbf{q}_1^{y}, \Lambda)$ in the class of the schemata $\{\tilde{\mathbf{S}}_{yy'}\}$ is reduced to the problem of accessibility of the states $\mathbf{q}_1^{y}$ in the class of the schemata $\{\mathbf{S}_{yy'}\}$ and is therefore unsolvable. In this case the schemata $\tilde{\mathbf{S}}_{yy'}$ have no reset operators of the kind **w** from $\mathbf{S}_{yy'}$. To make it more clear it is possible to change the definition of the schema $\mathbf{S}_{yy'}$ replacing the fragment from "... of infinite one-valued K–M schemata..." up to the end of the definition of the schema as follows:

... of infinite one-valued K–M schemata

$$\tilde{\mathbf{S}}_{yy'} = S(\tilde{M}, \tilde{V}, \tilde{X}, \tilde{Q}, \tilde{q}_0, \tilde{t}),$$

where

3.1. $\tilde{\mathbf{M}} = \{\mathbf{r}\}$, $\mathbf{V} = \{\mathbf{e}\}$, $\mathbf{D(e)} = \mathbf{R(e)} = \{\mathbf{r}\}$, $\tilde{\mathbf{X}} = \{\dot{\mathbf{e}}, \mathbf{e}_1\}$.

3.2. $\tilde{Q} = \{\langle \mathbf{q}, \mathbf{l}\rangle : \mathbf{q} \in \{\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2\}, \mathbf{l}$ is the word on the alphabet **X** or an empty word $\Lambda\}$, where **X** is taken from the definition of the schema $\mathbf{S}_{yy'}$.

3.3. $\tilde{\mathbf{q}}_0 = (\mathbf{q}_0, \Lambda)$.

3.4. Let us assume that for some word **l** in the alphabet **X** the following conditions are fulfilled:

(i) the value of $\mathbf{t}_y(\mathbf{q}_0^{y}, \mathbf{l})$ is defined in $\mathbf{S}_y$ and equales $\mathbf{q}_1^{y}$,

(ii) there exists a word **l′** obtained as result of the permutation of the symbols of the

word $\mathbf{l}$ not changing the order of termination symbols of the operator $\mathbf{u}$ as well as of the operator $\mathbf{v}$ and such that the value of $\mathbf{t}_{y'}(\mathbf{q}_0^y, \mathbf{l}')$ is defined in the schema $\mathbf{S}_{y'}$ and equals $\mathbf{q}_1^{y'}$.

Then for any $\mathbf{x} \in \tilde{\mathbf{X}}$ we let

$$\tilde{\mathbf{t}}((\mathbf{q}_0, \mathbf{l}), \mathbf{x}) = (\mathbf{q}_1, \Lambda),$$

and in other cases

$$\tilde{\mathbf{t}}((\mathbf{q}, \mathbf{l}), \mathbf{x}) = (\mathbf{q}_2, \Lambda).$$

4. On the Program Schemata Equivalence

Let us formulate some conditions sufficient for the undecidability of an equivalence problem. First we introduce some definitions.

An equivalence relation in a class of K–M schemata is called *interpretational* if the equivalence of two schemata of this class follows from the equality of computation sets of these schemata.

An operator u is *accessible* in a K–M schema $\mathbf{S}$ if the symbol $\dot{\mathbf{u}}$ is contained in some computation of $\mathbf{S}$ and $\mathbf{u}$ is the latest in this computation.

An equivalence relation in a class of K–M schemata is called *nondegenerate* if the nonequivalence of two schemata of this class follows from the existence of an operator $u(R(u) \neq \varnothing)$ which is accessible in one of these schemata and inaccessible in the other.

Theorem 3. *If the equivalence relation in the set of finite one-valued* K–M *schemata (flowcharts) is interpretational and non-degenerate then the equivalence problem is undecidable.*

Proof. Let $\mathbf{S}$ and $\mathbf{S}'$ be finite one-valued K–M flowcharts and satisfy the following conditions:

1. If the word $\mathbf{N}$ is a computation for $\mathbf{S}$ and, when working with it, $\mathbf{S}$ does not pass the fixed state $\mathbf{q}$, then $\mathbf{N}$ is a computation for $\mathbf{S}'$.

2. If the word $\mathbf{N}$ is the computation for $\mathbf{S}$ and, when working with it, $\mathbf{S}$ passes the state $\mathbf{q}$, then

2.1. the word $\mathbf{N}$ has a finite length,

2.2. the word $\mathbf{N}$ has the symbols of $\mathbf{X}_u$, where $\mathbf{u}$ is an operator $(R(u) \neq \varnothing)$ and $\mathbf{u}$ is not accessible in $\mathbf{S}'$.

Let us assume an interpretational, nondegenerate equivalence relation. Then it is easy to see that $\mathbf{S}$ and $\mathbf{S}'$ are nonequivalent iff the state $\mathbf{q}$ is accessible in $\mathbf{S}$. Hence, due to

Theorem 2 and its corollary, it is easy to get that the equivalence problem is undecidable for finite one-valued K–M flowcharts.

"It remains to show that there exist the schemata $\mathbf{S}$ and $\mathbf{S}'$ satisfying the above conditions. As $\mathbf{S}$ we can take the schemata $\mathbf{S}^*_{yy'}$ which differs from $\mathbf{S}_{yy'}$ only in one new operator $\mathbf{u}^*$ ($\mathbf{X}_{u^*} = \{\dot{u}^*, \mathbf{u}_1{}^*\}$, $\mathbf{R}(\mathbf{u}^*) = \{\mathbf{r}\}$), in two new states $\mathbf{q}^*$, $\mathbf{q}^{**}$, and $\mathbf{t}^*$ can be obtained from $\mathbf{t}_{yy'}$ only by letting $\mathbf{t}^*(\mathbf{q}_1^{y'}, \mathbf{u}^*) = \mathbf{q}^*$, $\mathbf{t}^*(\mathbf{q}^*, \mathbf{u}_1{}^*) = \mathbf{q}^{**}$. Here the operator $\mathbf{u}^*$ plays the role of the lossless operator, and $\mathbf{q}_1^{y'}$, that state from the proof. The schema $\mathbf{S}_{yy'}$ will be taken as the schema $\mathbf{S}'$."

Let us give some examples of interpretational nondegenerate equivalence relations (in the set of finite one-valued K–M schemata).

1. The above mentioned 0-equivalence relation.

2. K–M schemata $\mathbf{S}_1$ and $\mathbf{S}_2$ are *functionally equivalent* if each interpretation confirms either simultaneously infinite computations for these K–M schemata, or such simultaneously finite computations which define one and the same finite content of the memory (at one and the same initial data of the memory).

The undecidability of the functional equivalence problem is proved in [5] for extending the set of K–M schemata by means of special Algol operators $\mathbf{r}_i := \mathbf{r}_j$ (see the introduction of this paper).

3. Let $\mathbf{r} \in \mathbf{M}$ and $\mathbf{r}(\mathbf{N}_i)$ ($\mathbf{i} = 1, 2$) be the sequences of integers which are constructed on the ground of $\mathbf{N}_i = \mathbf{x}_1 \cdots \mathbf{x}_n \cdots$, as follows. Let us get the sequence $(\mathbf{N}_i)_r$ replacing each symbol $\mathbf{x}_n (\mathbf{n} \geqslant 1, \mathbf{x}_n = \mathbf{u}_j, \mathbf{j} \geqslant 1, \mathbf{r} \in \mathbf{R}(\mathbf{u}))$ for the number which is appropriate to variable $\mathbf{r}$ as a result of the switching of $\mathbf{x}_n$. The numerical subsequence of $(\mathbf{N}_i)_r$ is $\mathbf{r}(\mathbf{N}_i)$. The K–M schemata $\mathbf{S}_1$ and $\mathbf{S}_2$ are *equivalent in this history of cells* if each interpretation $\mathbf{J}$ confirms such computations $\mathbf{N}_1$ and $\mathbf{N}_2$ of $\mathbf{S}_1$ and $\mathbf{S}_2$ that for each variable $\mathbf{r}$: $\mathbf{r}(\mathbf{N}_1) = \mathbf{r}(\mathbf{N}_2)$.

The equivalence relation is introduced in [1].

4. Let $\Gamma(\mathbf{N}_1)$ ($\mathbf{i} = 1, 2$) be the graph plotted on $\mathbf{N}_i = \mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots$ as follows. All the pairs $(\mathbf{x}, \mathbf{n})$, where $\mathbf{x} = \mathbf{x}_n$ ($\mathbf{n} \geqslant 1$), are the vertices of the graph. From vertex $(\mathbf{x}, \mathbf{n})$ to vertex $(\mathbf{x}', \mathbf{n}')$, the arc, marked $\mathbf{r}$, is directed iff, when $\mathbf{n} < \mathbf{n}'$, $\mathbf{x}$ is "termination" symbol of some operator $\mathbf{u}$, $\mathbf{x}'$ is the "initiation" symbol of some operator $\mathbf{u}'$, $\mathbf{r} \in \mathbf{R}(\mathbf{u})$ $\mathbf{r} \in \mathbf{D}(\mathbf{u}')$ and for each transformer $\mathbf{u}''$, such that $\mathbf{x}_j$ is a some "termination" symbol of the operator $\mathbf{u}''$ and $\mathbf{n} < \mathbf{j} < \mathbf{n}'$, and it is true that $\mathbf{r} \notin \mathbf{R}(\mathbf{u}'')$. The K–M schemata $\mathbf{S}_1$ and $\mathbf{S}_2$ are *equivalent in the information graph* if every interpretation $\mathbf{J}$ confirms such computations $\mathbf{N}_1$ and $\mathbf{N}_2$ that information graphs $\Gamma(\mathbf{N}_1)$ and $\Gamma(\mathbf{N}_2)$ are equal.

Ianov's [16] equivalence (with decidable equivalence problem) and semigroup equivalence are examples of noninterpretational, nondegenerate relations.

Let us define semigroup equivalence and for it give the following result (see [11]), in which some sufficient conditions for decidability of the group equivalence problem are formulated.

Let all the transforms of $\mathbf{U}$ as well as a unique element $\mathbf{e} \in \mathbf{U}$ be generatrices of a semi-group $\mathbf{G}$. A finite set of defining relations, which define the relation of equality $\stackrel{\circ}{=}$ of the words of the semigroup, is given. Let μ be an arbitrary one-valued mapping $\mathbf{G}$ in $\mathbf{B}$, and each element of $\mathbf{B}$ associate some element of $\{+\mathbf{p}, -\mathbf{p}\}$ to each resolver $\mathbf{p} \in \mathbf{U}$.

Finite operational schemata $\mathbf{S_i(M, U, X, Q_i, q_0^i, t,)}$ $(\mathbf{i} = 1, 2)$ are *equivalent relatively to the semigroup* $\mathbf{G}$ iff for each function μ the words $\mathbf{c_1(S_1, \mu)}$ and $\mathbf{c_2(S_2, \mu)}$ are simultaneously finite and equal in the semigroup $\mathbf{G}$, where $\mathbf{c_i}$ $(\mathbf{i} = 1, 2)$ is a finite content of the memory $\mathbf{M}$ as a result of the computation $\mathbf{N_i}$, if we assume that

1. $\mathbf{M}$ contains one variable $\mathbf{r}$,
2. elements of $\mathbf{G}$ are the values of the variable $\mathbf{r}$,
3. initial content $\mathbf{c_0}$ of a memory $\mathbf{M}$ is equal to $\mathbf{e} \in \mathbf{G}$,
4. for each transformer $\mathbf{u}$ is true: $\mathbf{f_u(c) = cu}$,
5. for each resolver $\mathbf{u}$ is true: $\mathbf{g_u(c) = (\mu(c))(u)}$.

THEOREM. *In the set of finite one-valued flowcharts the equivalence problem, relative to the semigroup* $\mathbf{G}$, *is decidable if we have the following conditions:*

1. *The problem of the equality of the words in a semigroup* $\mathbf{G}$ *is decidable.*
2. $\mathbf{G}$ *is a semigroup with left cancellation, i.e., if* $\mathbf{e_1e_2 \stackrel{\circ}{=} e_1e_3}$ *then* $\mathbf{g_2 \stackrel{\circ}{=} g_3}$, *where* $\mathbf{g_i}$ $(\mathbf{i} = 1, 2, 3)$ *are the words in* $\mathbf{G}$.
3. $\mathbf{G}$ *is a semigroup with indecomposable unit* $\mathbf{e} \in \mathbf{G}$.

The theorem is proved in [11] (see [12]).

REFERENCES

1. RICHARD M. KARP AND RAYMOND E. MILLER, Parallel program schemata, *J. Comput. System Sci.* 3 (1969), 147–195.
2. A. P. YERSHOV AND A. A. LAPUNOV, On formalization of conception of program, *Kibernetika Kiev* 5 (1967), 40–56, in Russian.
3. H. G. RICE, *Trans. Amer. Math. Soc., Ser. 2* 43 (1953), 358.
4. S. IGARASHI, An axiomatic approach to the equivalence problems of algorithms with applications, Reprinted from Report of the Computer Center, Vol. I, pp. 1–101, University of Tokyo, 1, 1–101, Tokyo, 1968.
5. A. A. LETICHEVSKY, Functional equivalence of discrete transformers 1, *Kibernetika Kiev* 2 (1969), 5–16, in Russian.
6. D. LUCKHAM AND D. PARK, "The Undecidability of the Equivalence Problem for Program Schemata," Report No. 1141, Bolt, Beranek, and Newman, 1964.
7. I. D. ZASLAVSKY, Graph-schemata with a memory, *Trudy Mat. Inst. Steklov.* 72 (1964), 99–195, in Russian.

8. V. E. ITKIN, Ianov's program schemata with identical operations, Trudy Vsesoyusnoi konferentsii programirovaniu A, Kiev, 1968, in Russian.
9. V. E. ITKIN, "Parallel Program Schemata," Archive of Computer Center of Siberian Division of the USSR Academy of Sciences, 1969, in Russian.
10. DONALD M. KAPLAN, Regular expressions and the equivalence of programs, *J. Comput. System Sci.* 3 (1969), 361–386.
11. A. A. LETICHEVSKY, Functional equivalence of discrete transformes II, *Kibernetika Kiev.*
12. V. A. NYEPOMYASHCHIY, On a method of recognition of equivalence of program schemata and discrete transformes, Trudy II Vsesoyusnoi konferentsii po programirovaniu K, Novosibirsk, 1970, in Russian.
13. J. D. RUTLEDGE, On Ianov's program schemata, *J. ACM* (1964), 1–9.
14. M. A. TAYCLIN, Automata equivalence in regard to a commutative semigroup, *Algebra i Logic* 8 (1969), 553–600, in Russian.
15. V. A. TUZOV, Decision of problems for formal flowcharts with commutative operators (see Ref. [12]).
16. I. I. IANOV, The logical schemes of algorithms, *Problems of Cybernetics* 1 (1958), 75–125.
17. E. POST, A variant of a recursively unsolvable problem, *Bull. Amer. Math. Soc.* 52 (1946), 264–268.
18. V. E. KOTOV AND A. S. NARINYANI, On transformation of sequential programs into asynchronous parallel programs, *in* "Information Processing," North-Holland, Amsterdam, 1969.
19. J. D. RUTLEDGE, Program schemata as Automata: Part 1, Conference Record of the 1970 Eleventh Annual Symposium on Switching and Automata Theory, October 1970, pp. 7–24.
20. D. C. LUCKHAM, D. M. R. PARK, AND M. S. PATERSON, On formalised computer programs, *J. Comput. System Sci.* (1970), 220–249.

Printed in Belgium