JOURNAL OF COMPUTER AND SYSTEM SCIENCES: 3, 147–195 (1969)

Parallel Program Schemata

RICHARD M. KARP*

University of California, Berkeley, California 94720

AND

RAYMOND E. MILLER

IBM Watson Research Center, Yorktown Heights, New York 10598

Received April 15, 1968

ABSTRACT

This paper introduces a model called the parallel program schema for the representation and study of programs containing parallel sequencing. The model is related to Ianov's program schema, but extends it, both by modelling memory structure in more detail and by admitting parallel computation. The emphasis is on decision procedures, both for traditional properties, such as equivalence, and for new properties particular to parallel computation, such as determinacy and boundedness.

INTRODUCTION

A recurrent theme in the mathematical theory of computation is the problem of effectively determining from programs properties of the computations they perform and the functions they define. The present paper introduces a formal model for programs and, within the model, treats some aspects of this problem. The model is defined at a level of abstraction which leaves unspecified certain details pertinent to the operation of programs, and emphasis is placed on properties that hold true regardless of how these details are specified. This choice limits the types of questions that one can express within the model, but it permits the development of decision procedures for certain interesting properties whose analogues in more completely specified models would be undecidable.

Ianov's formulation of program schemata [*2*] (see also Rutledge [*10*]) is perhaps the earliest model at a level of detail similar to the present one. Ianov's approach stresses those aspects of the sequencing and control of a program which do not depend on the functions performed by instructions or the conditions which determine the outcomes of tests. To view this in another way, emphasis is placed on those properties which can be derived from a flowchart without considering the functions specified within the boxes.

---

* Formerly at IBM Watson Research Center, Yorktown Heights, New York 10598.

The present paper, which introduces the *parallel program schema*, is in the tradition of Ianov's work. However, in common with recent work of Luckham, Park, and Paterson [*7*], it treats another important element as well. Different data are given different names, and the names of the operands and results of each operation are specified. These names are represented within the model as locations in a memory, whereas, in Ianov's model, no such differentiation between memory locations is provided.[1]

The present work differs both from Ianov and Luckham, Park, and Paterson in that it is oriented toward parallel computation. In this respect, it is related to the authors' earlier work on computation graphs [*3*]. Also, in [*4*], a preliminary version of this work, some of the results are stated without proof. The emphasis on parallel computation is apparent in two ways. First, the control of sequencing is generalized to allow concurrent execution of several operations. Thus, it is possible to represent computations as they might be performed on a machine having several autonomous processing units using a common memory. Secondly, consideration of parallel computation suggests new questions and properties to be studied along with such traditional topics as the equivalence of programs. For example, the phenomenon of "races" between computation steps, leading to the possibility of indeterminacy, is treated extensively.

The practical motivation of this study stems from the need to understand the role of parallel computation in the design and effective use of computer systems. The approach taken in the present paper contributes to this goal in several ways. In particular, the results include methods applicable to determining the degree of parallelism of a program and to testing whether a program is determinate; that is, whether the results of a computation are independent of the relative speeds of concurrently executed operations. The formulation is also potentially applicable to the general representation of parallel sequencing, the design of compilers and operating systems for parallel computers, and the conversion of algorithms to more highly parallel equivalent forms.

Section I of the paper introduces the parallel program schema and defines the properties to be studied. Section II gives necessary and sufficient conditions for a certain class of schemata to be determinate. After some technical results concerning prefixes of computations are derived in Section III, Section IV treats a special type of schema called counter schema and gives decision procedures for properties such as determinacy, boundedness, termination, and repetition-freeness. The procedures are obtained by reducing these problems to solvable questions about a simple geometric structure, the vector addition system. A proof of the unsolvability of the equivalence problem for certain classes of schemata is also given. This unsolvability result, as

---

[1] It should be noted, however, that a finite amount of information about memory structure can be represented via the "shift relation" in Ianov's model.

well as an analogous unsolvability result by Luckham, Park, and Paterson [7] hinges on the subdivision of the memory into more than one location. In contrast, the equivalence problem is solvable for the Ianov model, where the memory is "monolithic." Section V specializes the counter schema further to obtain a model called the parallel flowchart. From another viewpoint, the parallel flowchart is a generalization of the flowchart commonly used by programmers which accommodates most of the methods which have been proposed for representing parallel sequencing. Section VI treats in detail the case of decision-free flowcharts. Here the analysis proves to be much easier and, in an important case, the equivalence problem is solvable.

I. PARALLEL PROGRAM SCHEMATA

A program can be regarded as a collection of primitive operations which depend on and affect memory locations, together with a specification of the rules governing the initiation and termination of operations. The model we shall consider is based on this viewpoint and is specifically designed as a representation for programs in which several operations or several performances of any operation can be concurrently in progress.

DEFINITION 1.1. A (parallel program) schema $\mathscr{S} = (M, A, \mathscr{T})$ is specified by:

(1) A set M of *memory locations*,

(2) A finite set $A = \{a, b, \ldots\}$ of *operations* and, for each operation a in A:

  (i) a positive integer $K(a)$ called the *number of outcomes* of a;

  (ii) a set $D(a) \subseteq M$ whose elements are the *domain locations* for a;

  (iii) a set $R(a) \subseteq M$ whose elements are the *range locations* for a.

(3) A quadruple $\mathscr{T} = (Q, q_0, \Sigma, \tau)$ called the *control*, where:

  (i) Q is a set of *states*;

  (ii) q_0 is a designated state called the *initial state*;

  (iii) Σ, the *alphabet*, is the union of

$$\Sigma_i = \bigcup_{a \in A} \{\bar{a}\}$$

the *initiation symbols* and

$$\Sigma_t = \bigcup_{a \in A} \{a_1, \ldots, a_{K(a)}\}$$

the *termination symbols*.

(iv) τ, the *transition function*, is a partial function from $Q \times \Sigma$ into Q which is total on $Q \times \Sigma_t$.

The sequencing of operations is enforced by the control $\mathscr{T}$. The elements of Σ denote the primitive steps which can affect the sequencing (elements of Σ_i denote initiations of operations and elements of Σ_t denote terminations of operations with given outcomes), and the set of states enables $\mathscr{T}$ to "remember" relevant information about the sequence of past events. The control $\mathscr{T}$ does not allow an operation a to be initiated until a state is reached for which $\tau(q, \bar{a})$ is defined. Once an operation is initiated, it can terminate anytime thereafter since τ is total on $Q \times \Sigma_t$. When an operation a is initiated, it obtains operands from the memory locations specified by $D(a)$, and when an operation a terminates, it places its results in the memory locations specified by $R(a)$ and selects an outcome a_j. In this way, a $K(a)$-way conditional transfer occurs when a terminates.

The control $\mathscr{T}$ may be viewed as a graph in which states are represented by nodes and the transitions between states are represented by directed edges. When the set of states is finite, such a graph is somewhat similar to a conventional flowchart. Of course, parallel program schemata also differ from conventional flowcharts in that parallel sequencing is possible, since initiations and terminations need not alternate.

Before giving a precise discussion of the way in which the computations associated with a schema evolve, we remark that a schema is uninterpreted; that is, it does not assign a particular "meaning" to its operations. The only information given about operations is their associated domain and range locations in the memory. This paper is concerned with properties of schemata which hold true for all interpretations placed on the operations. In order to study properties true for all interpretations, we introduce a precise concept of interpretation and state how a schema, together with an interpretation of its operations, defines computations.

DEFINITION 1.2. An *interpretation* $\mathscr{I}$ of a schema $\mathscr{S}$ is specified by:

(i) a function C associating with each element $i \in M$ a set $C(i)$;

(ii) an element $c_0 \in \times_{i \in M} C(i)$;

(iii) for each operation a, two functions:

$$F_a : \underset{i \in D(a)}{\times} C(i) \rightarrow \underset{i \in R(a)}{\times} C(i)$$

$$G_a : \underset{i \in D(a)}{\times} C(i) \rightarrow \{a_1, a_2, \ldots, a_{K(a)}\}.$$

The set $C(i)$ consists of the possible contents of memory location i, and c_0 denotes the initial contents of memory. The function F_a determines the result which operation a stores in memory upon its completion. Upon the completion of the operation a $K(a)$-way test takes place with the outcome determined by G_a.

DEFINITION 1.3. An *$\mathcal{S}$-instantaneous description* α is a triple (c, q, μ) in which:

(i) $c \in \times_{i \in M} C(i)$; by definition, $c(i)$ denotes the element selected from $C(i)$;

(ii) q is a state;

(iii) μ is a function associating with each $a \in A$, a finite sequence of elements from $\times_{i \in D(a)} C(i)$.

An $\mathcal{S}$-instantaneous description gives the status of a schema at any time during the performance of its operations: c gives the contents of memory and q gives the state of the control. Each list $\mu(a)$ can be thought of as a queue of tuples of data words. Each tuple corresponds to a performance of a which has been initiated but not yet terminated; the elements of the tuple give the values in the locations $D(a)$ at the time of initiation. As will be seen from Definition 1.5, the performances of a are assumed to terminate in the same order as they are initiated.

DEFINITION 1.4. The $\mathcal{S}$-instantaneous description $\alpha_0 = (c_0, q_0, \mu_0)$ where, for all a, $\mu_0(a)$ is the null sequence, is called the *initial $\mathcal{S}$-instantaneous description.*

In what follows, early lower case Roman letters will denote generic elements of A, and the letters σ and π will denote generic elements of Σ. Late Roman letters will denote finite or infinite words over the alphabet Σ. We let Σ^{ω} denote the set of infinite words and Σ^* the set of finite words over Σ. If $x \in \Sigma^*$, then $l(x)$ will denote the length of x. If $k \leqslant l(x)$, then x_k will denote the kth element of x, and ${}_kx$, the prefix of x of length k.

The next definition gets at the heart of the rules for sequencing of operations.

DEFINITION 1.5. The partial function $\alpha \cdot \sigma$, where $\alpha = (c, q, \mu)$ is an $\mathcal{S}$-instantaneous description and σ is an element of Σ, is defined as follows:

(1) If $\sigma = \bar{a}$, where $a \in A$, $\alpha \cdot \bar{a}$ is defined if and only if $\tau(q, \bar{a})$ is defined. If so, $\alpha \cdot \bar{a} = (c', q', \mu')$ where:

(i) $c' = c$

(ii) $q' = \tau(q, \bar{a})$

(iii) for $b \neq a$, $\mu'(b) = \mu(b)$; $\mu'(a)$ is the sequence obtained by adjoining to the end of $\mu(a)$ the tuple $\times_{i \in D(a)} c(i)$.

(2) If $\sigma = a_j$, where $a \in A$, $\alpha \cdot a_j$ is defined if and only if $\mu(a)$ is nonempty and $G_a(\xi) = a_j$, where ξ denotes the first element of $\mu(a)$. In this case, $\alpha \cdot a_j = (c', a', \mu')$ where:

(i) for $i \notin R(a)$, $c'(i) = c(i)$

(ii) for $i \in R(a)$, $c'(i)$ is the component of $F_a(\xi)$ corresponding to i

(iii) $q' = \tau(q, a_j)$

(iv) $\mu'(b) = \mu(b)$, $b \neq a$, and $\mu(a) = \xi\mu'(a)$.

It is a consequence of these rules that the performances of an operation a terminate in the order in which they are initiated.

Viewing $\mu(a)$ as a queue, then, we see that the queue is first-in first-out. This representation by $\mu(a)$ of instances of a which have been initiated, but not yet completed, is a formal one which is sufficient for our purposes but still allows for a wide variety of physical interpretations. For example, one may think of $\mu(a)$ as an actual queue in which sets of data for operation a are waiting until the appropriate computational facilities become available. Another quite different physical interpretation is that sufficient facilities are available for any number of concurrent performances of each operation, and that initiation symbols indicate actual initiation of operation performance. Here, $\mu(a)$ simply designates those performances which are currently in progress, and no actual queue of data is required. The actual timing of operation performances is then arbitrary, except for the order preserving constraint imposed by the first-started first-completed assumption made on the μ lists. As can be readily imagined, other physical interpretations for $\mu(a)$ can also be given.

It will be convenient to extend the function τ in the usual way by the identity $\tau(q, x\sigma) = \tau(\tau(q, x), \sigma)$, where the left-hand side is undefined if the right-hand side is. The partial function $\cdot$ is extended similarly.

Definition 1.6. A finite or infinite word x over the alphabet Σ is an $\mathscr{I}$-*computation* if:

(i) for every prefix y of x, $\alpha_0 \cdot y$ is defined;

(ii) if x is finite, then, for all $\sigma \in \Sigma$, $\alpha_0 \cdot x\sigma$ is undefined;

(iii) *finite delay property*: If y is a prefix of x and $\sigma \in \Sigma$ with the property that, for every z such that yz is a prefix of x, $\alpha_0 \cdot yz\sigma$ is defined, then, for some z', $yz'\sigma$ is a prefix of x.

The $\mathscr{I}$-computations represent the possible sequences of primitive steps which may occur in applying the algorithm represented by a schema $\mathscr{S}$ with interpretation $\mathscr{I}$. The fact that more than one $\mathscr{I}$-computation can exist is a consequence of the fact that, for a given α, there may exist several elements σ for which $\alpha \cdot \sigma$ is defined. Of course, it is usually intended that all the $\mathscr{I}$-computations should, in some sense, produce the same result.

Definition 1.7. If x is an $\mathscr{I}$-computation or a prefix of an $\mathscr{I}$-computation, then $\psi(x)$, called the *history* of x, is the following sequence of $\mathscr{I}$-instantaneous descriptions:

$$\psi(x) = \alpha_0 , \alpha_0 \cdot {}_1x, \alpha_0 \cdot {}_2x, \ldots, \alpha_0 \cdot {}_kx, \ldots$$

Also, let $\psi_i(x)$ denote the subsequence of $\psi(x)$ containing α_0 and containing $\alpha_0 \cdot {}_k x$ if and only if $x_k = a_j$ for some a such that $i \in R(a)$.

Thus, $\psi_i(x)$ is the sequence of $\mathscr{I}$-instantaneous descriptions associated with operations which store results in location i.

In order to extract components and sets of components from an $\mathscr{I}$-instantaneous description $\alpha = (c, q, \mu)$, we introduce several projection operators.[2]

$$\Pi_i(\alpha) = c(i), \qquad i \in M$$

$$\Pi_a(\alpha) = \mu(a), \qquad a \in A.$$

If $S = \{i_2, i_2, ..., i_r\}$ is a subset of M, where $i_1 < i_2 < \cdots < i_r$, then $\Pi_S(\alpha)$ is the ordered set $\langle c(i_1), c(i_2), ..., c(i_r) \rangle$. Projection operators may also be applied to sequences of $\mathscr{I}$-instantaneous descriptions. For example, $\Pi_i(\alpha^{(0)}, \alpha^{(1)}, ...) = \Pi_i(\alpha^{(0)}), \Pi_i(\alpha^{(1)}), ...$.

DEFINITION 1.8. Let x be an $\mathscr{I}$-computation or a prefix of an $\mathscr{I}$-computation. Then $\Omega_i(x) = \Pi_i(\psi_1(x))$ is called the *contents sequence of cell i* for x.

Thus, $\Omega_i(x)$ is the sequence consisting of the initial contents of cell i followed by the successive values which occur in cell i upon the terminations of operations which store in cell i.

DEFINITION 1.9. A schema $\mathscr{S}$ is *determinate* if, whenever x and y are $\mathscr{I}$-computations for the same interpretation $\mathscr{I}$,

$$\Omega_i(x) = \Omega_i(y) \qquad \text{for all} \qquad i \in M.$$

Thus determinacy establishes that the entire sequence of values stored in any single location is determined by the interpretation, even though the sequence of operations and the relative times at which different locations are changed may not be well determined.

DEFINITION 1.9. Two schemata $\mathscr{S} = (M, A, \mathscr{T})$ and $\mathscr{S}' = (M, A, \mathscr{T}')$ are *equivalent* if, for each $i \in M$ and each interpretation $\mathscr{I}$,

$$\{\Omega_i(x) \mid x \text{ is an } \mathscr{I}\text{-computation for } \mathscr{S}\}$$
$$= \{\Omega_i(y) \mid y \text{ is an } \mathscr{I}\text{-computation for } \mathscr{S}'\}.$$

DEFINITION 1.10. A schema $\mathscr{S}$ is called *bounded* if there is a constant K such that, for every $\mathscr{I}$, and for every $\mathscr{I}$-instantaneous description (c, q, μ) which appears in

---

[2] Here we make the natural assumption that $A \cap M = \varphi$; otherwise, a more complicated notation is needed to avoid ambiguity.

the history of an $\mathscr{I}$-computation, the sum of the lengths of all lists $\mu(a)$ is bounded by K. If K can be taken to equal 1, then $\mathscr{S}$ is called *serial.*

If a schema is bounded, then there is a limit on the amount of concurrent activity, or parallelism, in computations for this schema. If the schema is serial, then no concurrent activity is possible.

Most of this paper will be concerned with determining the extent to which such properties as the determinacy, equivalence, and boundedness of schemata can effectively be tested.

II. DETERMINACY

One of the most interesting phenomena connected with parallel computation is indeterminacy, arising from the fact that, when several steps are concurrently in progress or eligible to be initiated the results may depend on the order in which the corresponding terminations and initiations occur. In this section, we seek to establish necessary and sufficient conditions for a schema to be determinate. The theorems we derive are valid only for certain classes of schemata distinguished by properties of the control or of the operations. These properties are introduced in the following definitions.

DEFINITION 2.1. A schema is *persistent* if, whenever σ and π are distinct elements of Σ, and $\tau(q, \sigma)$ and $\tau(q, \pi)$ are defined, then $\tau(q, \sigma\pi)$ and $\tau(q, \pi\sigma)$ are also defined.

Thus, in a persistent schema, an operation once ready to be initiated remains ready until it is initiated.

DEFINITION 2.2. A schema is *commutative* if, whenever $\tau(q, \sigma\pi)$ and $\tau(q, \pi\sigma)$ are defined, $\tau(q, \sigma\pi) = \tau(q, \pi\sigma)$.

In a commutative schema, whenever two primitive events may occur in either order, the state of control is independent of the actual order of occurrence.

DEFINITION 2.3. A schema is *permutable* if, whenever σ and π are initiation symbols and $\tau(q, \sigma\pi)$ is defined, then $\tau(q, \pi)$ is also defined.

A permutable schema has the property that if an operation is not ready to be initiated, it remains in that condition regardless of what other initiations occur.

DEFINITION 2.4. A schema $\mathscr{S}$ is *lossless* if, for every $a \in A$, $R(a) \neq \varphi$.

Thus, a lossless schema is one in which the termination of each operation places a result in the memory.

The next theorem shows that for a certain class of schemata, determinacy is equivalent to a kind of commutativity between primitive steps in a computation.

THEOREM 2.1. *Let $\mathscr{S}$ be a persistent, commutative, lossless schema. Then $\mathscr{S}$ is determinate if and only if the following condition holds for every interpretation $\mathscr{I}$:*

(A) *If α_0 is the initial $\mathscr{I}$-instantaneous description, u is an element of Σ^*, and σ and π are elements of Σ such that $\alpha_0 \cdot u\sigma\pi$ and $\alpha_0 \cdot u\pi\sigma$ are both defined, then $\alpha_0 \cdot u\sigma\pi = \alpha_0 \cdot u\pi\sigma$.*

As a first step towards the proof of Theorem 2.1 we discuss a special kind of interpretation which will also be useful elsewhere.

DEFINITION 2.5. An interpretation $\mathscr{I}$ is *one-one* if, for any operations a and b, any $s \in \times_{i \in D(a)} c(i)$ and $s' \in \times_{i \in D(b)} c(i)$, and any $i \in R(a) \cap R(b)$,

$$\Pi_i(F_a(s)) \neq \Pi_i(F_b(s'))$$

unless $a = b$ and $s = s'$.

LEMMA 2.1. *Let $\mathscr{S}$ be a schema and $\mathscr{I}$ an interpretation with initial instantaneous description α_0. Then there exists a one-one interpretation $\mathscr{I}'$ with initial instantaneous description α_0' having the same set of computations as $\mathscr{I}$, for which the following property holds. Let x and y be sequences such that $\alpha_0 \cdot x$ and $\alpha_0 \cdot y$ are defined; then*

$$\Pi_i(\alpha_0 \cdot x) \neq \Pi_i(\alpha_0 \cdot y) \Rightarrow \Pi_i(\alpha_0' \cdot x) \neq \Pi_i(\alpha_0' \cdot y).$$

Proof. The interpretation $\mathscr{I}'$ is constructed so that the value stored in a cell at a given step of an $\mathscr{I}'$-computation is an ordered pair. The first component of the pair gives the value that $\mathscr{I}$ would store at the corresponding step, and the second component is a "well-formed formula" which indicates the sequence of operations by which the value was determined, starting with the initial contents of memory. Supposing that $\mathscr{I}$ specified $C(i)$, c_0, $\{F_a \mid a \in A\}$, and $\{G_a \mid a \in A\}$, the interpretation $\mathscr{I}'$ is constructed as follows:

$$C'(i) = C(i) \times B^* \qquad \text{where} \quad B = A \cup \{i, (\,,\,)\}$$

$$\Pi_i(c_0') = (\Pi_i(c_0), i).$$

The formation of well-formed formulas is carried out in the manner described below. If $D(a)$ contains r elements, so that F_a and F_a' operate on r-tuples, then for $i \in R(a)$,

$$\Pi_i(F_a'((\xi_1, \eta_1), (\xi_2, \eta_2), \ldots, (\xi_r, \eta_r)))$$
$$= (\Pi_i(F_a(\xi_1, \xi_2, \ldots, \xi_r)), \operatorname{Cat}_a(\eta_1, \eta_2, \ldots, \eta_r)),$$

where, if $x_1, x_2, \dots, x_r$ are elements of B^*, then $\mathrm{Cat}_a(x_1, x_2, \dots, x_r) = a(x_1)(x_2) \cdots (x_r)$, which is also an element of B^*.

$$G'_a((\xi_1, \eta_1), (\xi_2, \eta_2), \dots, (\xi_r, \eta_r)) = G_a(\xi_1, \xi_2, \dots, \xi_r).$$

It is readily proved by induction on $l(x)$ that

(i) $(c_0, q_0, \mu_0) \cdot x$ is defined with respect to $\mathscr{I}$ if and only if $(c'_0, q_0, \mu_0) \cdot x$ is defined with respect to $\mathscr{I}'$;

(ii) if $(c_0, q_0, \mu_0) \cdot x$ and $(c'_0, q_0, \mu_0) \cdot x$ are defined, then they are related as follows:

If $\Pi_i(c'_0, q_0, \mu_0) \cdot x) = (\xi, \eta)$, then $\Pi_i((c_0, q_0, \mu_0) \cdot x) = \xi$. All that remains to be proved is that $\mathscr{I}'$ is one–one. Clearly, if

$$\begin{aligned} &\Pi_i(F'_a((\xi_1, \eta_1), (\xi_2, \eta_2), \dots, (\xi_r, \eta_r))) \\ &\quad = \Pi_i(F'_b((\lambda_1, \phi_1), (\lambda_2, \phi_2), \dots, (\lambda_t, \phi_t))) \end{aligned}$$

then from the properties of Cat_a and Cat_b it follows that $a = b$, $r = t$, and $\eta_j = \phi_j$, $j = 1, 2, \dots, r$. Also, it is easily seen that if (ξ, η) and (λ, η) can both be stored in cell i during $\mathscr{I}'$-computations, then $\xi = \lambda$, since the well-formed formula η specifies exactly how the value in cell i was computed as a composition of the single-valued functions $F_a, F_b, \dots$. Thus, $\mathscr{I}'$ is one–one.

We proceed toward the proof of Theorem 2.1 via a series of simple lemmas.

LEMMA 2.2. *Condition* (A) *of Theorem* 2.1 *holds for every interpretation if and only if it holds for every one–one interpretation.*

Proof. If Condition (A) holds for every interpretation, it obviously holds for every one–one interpretation. Conversely, let $\mathscr{I}$ be an interpretation, and let $\mathscr{I}'$ be a one–one interpretation with the same computations as $\mathscr{I}$, and meeting the condition of Lemma 2.1. Then $\mathscr{I}$ satisfies Condition (A) if $\mathscr{I}'$ does.

LEMMA 2.3. *Let $\mathscr{S}$ be a persistent, commutative, lossless schema, $\mathscr{I}$, a one–one interpretation, and α an $\mathscr{I}$-instantaneous description. Then for each pair of operations a and b,*

(a) *if $\alpha \cdot \bar{a}\bar{b}$ and $\alpha \cdot \bar{b}\bar{a}$ are defined, then $\alpha \cdot \bar{a}\bar{b} = \alpha \cdot \bar{b}\bar{a}$;*

(b) *if $\alpha \cdot \bar{a}b_l$ and $\alpha \cdot b_l\bar{a}$ are defined, then $\alpha \cdot \bar{a}b_l = \alpha \cdot b_l\bar{a}$ if and only if*

(i) *$R(b) \cap D(a) = \varphi$ or*

(ii) *b_l is a repetition*; i.e., $\Pi_{R(b)}(\alpha) = \Pi_{R(b)}(\alpha \cdot b_l)$;

(c) *if* $\alpha \cdot a_j b_l$ *and* $\alpha \cdot b_l a_j$ *are defined, then* $\alpha \cdot a_j b_l = \alpha \cdot b_l a_j$ *if and only if* $R(a) \cap R(b) = \varphi$.

Proof. Let $\alpha = (c, q, \mu)$.

(a) There is nothing to be proven unless $a \neq b$. Assuming this, $(c, q, \mu) \cdot \bar{a}\bar{b} = (c, \tau(q, \bar{a}\bar{b}), \mu')$, where $\mu'(d) = \mu(d)$, $d \notin \{a, b\}$, $\mu'(a) = \mu(a)\Pi_{D(a)}(c)$, and $\mu'(b) = \mu(b)\,\Pi_{D(b)}(c)$; $(c, q, \mu) \cdot \bar{b}\bar{a}$ is defined similarly. Thus, using the commutativity of τ, $(c, q, \mu) \cdot \bar{a}\bar{b} = (c, q, \mu) \cdot \bar{b}\bar{a}$.

(b) Consider first the case $a \neq b$. $(c, q, \mu) \cdot \bar{a}b_l = (c', \tau(q, \bar{a}b_l), \mu')$ and $(c, q, \mu) \cdot b_l\bar{a} = (c', \tau(q, b_l\bar{a}), \mu'')$ where $\mu'(d) = \mu''(d) = \mu(d)$, $d \neq a, b$, $\mu'(b) = \mu''(b)$, $\mu'(a) = \mu(a)\,\Pi_{D(a)}(c)$ and $\mu''(a) = \mu(a)\,\Pi_{D(a)}(c')$. Hence, because τ is commutative, $(c, q, \mu) \cdot \bar{a}b_l = (c, q, \mu) \cdot b_l\bar{a}$ if and only if $\Pi_{D(a)}(c') = \Pi_{D(a)}(c)$; similar reasoning shows that this is also true when $a = b$. Now (since $\mathscr{I}$ is one–one) $\Pi_{D(a)}(c') = \Pi_{D(a)}(c)$ if and only if $R(b) \cap D(a) = \varphi$ or b_l is a repetition.

(c) We may assume $a \neq b$; for if $a = b$, then $(c, q, \mu) \cdot a_j b_l$ and $(c, q, \mu) \cdot b_l a_j$ are both defined only when $j = l$. Also $(c, q, \mu) \cdot a_j b_l = (c', \tau(q, a_j b_l), \mu')$, and $(c, q, \mu) \cdot b_l a_j = (c'', \tau(q, b_l a_j), \mu')$, where c' differs from c'' precisely in those locations contained in $R(b) \cap R(a)$. Hence, $\alpha \cdot a_j b_l = \alpha \cdot b_l c_j$ precisely when no such locations exist.

LEMMA 2.4. *Let* $\mathscr{S}$ *be a persistent, commutative, lossless schema,* $\mathscr{I}$ *a one–one interpretation, and* α_0 *the initial* $\mathscr{I}$*-instantaneous description. Let* v *be an element of* Σ^*, *and* σ *and* π *elements of* Σ *such that* $\alpha_0 \,.\, v\sigma\pi = \alpha_0 \cdot v\pi\sigma$. *Then, for any* w *in* $\Sigma^* \cup \Sigma^\omega$,

(a) $v\sigma\pi w$ *is an* $\mathscr{I}$*-computation if and only if* $v\pi\sigma w$ *is an* $\mathscr{I}$*-computation*;

(b) *for any* $i \in M$, $\Omega_i(v\sigma\pi w) = \Omega_i(v\pi\sigma w)$.

Proof. Conclusion (a) follows by checking that $v\sigma\pi w$ satisfies the conditions of Definition 1.6 if and only if $v\pi\sigma w$ does. By inspecting the cases enumerated in the proof of Lemma 2.3, we find that, if $\alpha_0 \cdot v\sigma\pi = \alpha_0 \cdot v\pi\sigma$, then $\Omega_i(v\sigma\pi) = \Omega_i(v\pi\sigma)$; conclusion (b) then follows readily.

LEMMA 2.5. *Let* $\mathscr{S}$ *be a persistent schema,* $\mathscr{I}$ *an interpretation, and* α_0 *the initial* $\mathscr{I}$*-instantaneous description. Let* u *and* v *be elements of* Σ^*, w *an element of* $\Sigma^* \cup \Sigma^\omega$, *and* σ *an element of* Σ.

(a) *If* $\alpha_0 \cdot u\sigma$ *is defined,* $\sigma \notin v$ *and* $\alpha_0 \cdot uv$ *is defined, then* $\alpha_0 \cdot uv\sigma$ *is defined*;

(b) *if* $\alpha_0 \cdot u\sigma$ *is defined and* uw *is an* $\mathscr{I}$*-computation, then* $\sigma \in w$.

Proof. Conclusion (a) follows from Definition 6, with persistence required only where σ is an initiation symbol. Conclusion (b) follows from (a), together with the finite delay property.

Proof of Theorem 2.1. Let $\mathscr{I}$ be a one–one interpretation for which Condition (A) is satisfied. Suppose x and y are $\mathscr{I}$-computations such that, for some i, $\Omega_i(x) \neq \Omega_i(y)$. We shall show that, for any $n \leqslant l(x)$, (for all n if x is an infinite sequence), there is an $\mathscr{I}$-computation $z(n)$ such that

(i) $z(n)$ has the same cell sequences as y and

(ii) ${}_nz(n) = {}_nx$.

It will then follow that $\Omega_i(x) = \Omega_i(y)$ for all i, giving a contradiction. The proof is by induction. Setting $z(0) = y$, we have the result for $n = 0$. Let us assume that the result holds for $n = k$, and that $l(x) = k + 1$. Then $(\alpha_0 \cdot ({}_kx)) \cdot x_{k+1}$ is defined, or, equivalently, $(\alpha_0 \cdot ({}_kz(k))) \cdot x_{k+1}$ is defined. For brevity, let t denote ${}_kz(k)$. By conclusion (b) of Lemma 2.5, $z(k)$ can be written as $z(k) = tvx_{k+1}u$, for some v, where $x_{k+1} \in v$. If v is null, we may take $z(k + 1) = z(k)$. Otherwise, suppose $v = w\pi$, $\pi \in \Sigma$. Since $(\alpha_0 \cdot t) \cdot x_{k+1}$ is defined, it follows from conclusion (a) of Lemma 2.5 that $(\alpha_0 \cdot tw) \cdot x_{k+1}$ is defined, and $(\alpha_0 \cdot tw\pi) \cdot x_{k+1}$ is defined. Also, since $(\alpha_0 \cdot tw) \cdot \pi$ is defined, $(\alpha_0 \cdot twx_{k+1}) \cdot \pi$ is defined. But, by hypothesis, $(\alpha_0 \cdot tw) \cdot x_{k+1}\pi = (\alpha_0 \cdot tw) \cdot \pi x_{k+1}$. Hence, by Lemma 2.4, $twx_{k+1}\pi u$ is an $\mathscr{I}$-computation having the same cell sequences as $z(k)$. Similar interchanges can be used to "slide" x_{k+1} past each element of v, yielding eventually the required $\mathscr{I}$-computation $z(k + 1) = ({}_{k+1}x)\,vu$.

We have assumed that $\Omega_i(x) \neq \Omega_i(y)$ for some i. But this is easily seen to contradict the result just proven. For, if $\Omega_i(x) \neq \Omega_i(y)$, then, for some positive integer k, one of the following holds:

(i) $[\Omega_i(x)]_k$ and $[\Omega_i(y)]_k$ are both defined, but are unequal;

(ii) $[\Omega_i(y)]_k$ is defined, but $[\Omega_i(x)]_k$ is not;

(iii) $[\Omega_i(x)]_k$ is defined, but $[\Omega_i(y)]_k$ is not.

In cases (i) and (ii), there is a suitably large p such that $[\Omega_i(z(p))]_k \neq [\Omega_i(x)]_k$, contradicting the property that $\Omega_i(z(p)) = \Omega_i(y)$; case (iii) is disposed of similarly. These contradictions establish that $\Omega_i(x) = \Omega_i(y)$. Thus, Condition (A) implies determinacy.

To complete the proof, suppose that $\alpha_0 \cdot u\sigma\pi \neq \alpha_0 \cdot u\pi\sigma$. Then, by Lemma 2.3, either $\pi = b_l$ and $\sigma = \bar{a}$, where $R(b) \cap D(a) \neq \varphi$ and b_l is not a repetition or $\pi = b_l$ and $\sigma = a_j$ where $R(b) \cap R(a) \neq \varphi$. In either case, there is an i such that $\Omega_i(u\sigma\pi)$ and $\Omega_i(u\pi\sigma)$ are of equal length, and differ in their last element. If these sequences are prefixes of computations x' and x'', respectively, then $\Omega_i(x') \neq \Omega_i(x'')$. Hence, Condition (A) is necessary and sufficient for determinacy.

COROLLARY 2.1. *Let $\mathscr{S}$ be a persistent, commutative, lossless schema. Then $\mathscr{S}$ is determinate if and only if, for each interpretation $\mathscr{I}$, with initial $\mathscr{I}$-instantaneous description* α_0;

(i) *if* $\alpha_0 \cdot u\bar{a}b_l$ *and* $\alpha_0 \cdot ub_l\bar{a}$ *are defined, then* $R(b) \cap D(a) = \varphi$ *or* $\Pi_{R(b)}(\alpha_0 \cdot ub_l) = \Pi_{R(b)}(\alpha_0 \cdot u)$ (i.e., b_l *is a repetition*) *and,*

(ii) *if* $\alpha_0 \cdot ua_jb_l$ *and* $\alpha_0 \cdot ub_la_j$ *are defined, then* $R(b) \cap R(a) = \varphi$.

This corollary follows directly from Theorem 2.1 and Lemma 2.3.

Our criterion for determinacy can be simplified provided that we restrict our attention to schemata with the property that between any two initiations of an operation some result is stored in one of the domain locations of that operation.

DEFINITION 2.6. A schema $\mathscr{S}$ is called *repetition-free* if the following implication holds:

> $v\bar{a}w\bar{a}x$ is an $\mathscr{I}$-computation for some $\mathscr{I} \Rightarrow w$ contains a termination symbol c_j such that $R(c) \cap D(a) \neq \varphi$.

The proviso that b_l is not a repetition can, of course, be omitted if $\mathscr{S}$ is known to be repetition-free. The condition can further be simplified if $\mathscr{S}$ is permutable. To express the simplification, we introduce the relation $\rho \subseteq A \times A$, defined as follows: $a\rho b \Leftrightarrow R(a) \neq \varphi$, $R(b) \neq \varphi$ and $[D(a) \cap R(b)] \cup [R(a) \cap D(b)] \cup [R(a) \cap R(b)] \neq \varphi$.

A further definition is needed. Let a be an element of A, x an element of $\Sigma^* \cup \Sigma^\omega$, and m a positive integer. If x contains an mth occurrence of a, and also contains an mth occurrence of a symbol from the set $\{a_1, a_2, \ldots, a_{K(a)}\}$, then these two occurrences are called *mates*.

COROLLARY 2.2. *Let* $\mathscr{S}$ *be a schema which is repetition-free, lossless, persistent, commutative, and permutable. Then the following statements are equivalent:*

(1) $\mathscr{S}$ is not determinate;

(2) for some interpretation $\mathscr{I}$, with initial $\mathscr{I}$-instantaneous description α_0, there exist $u \in \Sigma^*$, $a \in A$ and $b \in A$ such that either

(i) $R(b) \cap D(a) \neq \varphi$ and, for some l, $\alpha_0 \cdot u\bar{a}b_l$ and $\alpha_0 \cdot ub_l\bar{a}$ are both defined or

(ii) $R(a) \cap R(b) \neq \varphi$ and, for some j and l, $\alpha_0 \cdot ua_jb_l$ and $\alpha_0 \cdot ub_la_j$ are both defined

(3) for some interpretation $\mathscr{I}$, with initial $\mathscr{I}$-instantaneous description α_0, there exist $w \in \Sigma^*$, $a \in A$ and $b \in A$ such that $a\rho b$ and $\alpha_0 \cdot w\bar{a}$ and $\alpha_0 \cdot w\bar{b}$ are both defined;

(4) for some interpretation $\mathscr{I}$ there exist $\mathscr{I}$-computations x and y and operations a and b such that $a\rho b$ and the subsequence $\mathscr{E}_{\bar{a}\bar{b}}(x)$ obtained by extracting all occurrences of $\bar{a}$ and $\bar{b}$ from x differs from the subsequence $\mathscr{E}_{\bar{a}\bar{b}}(y)$ similarly obtained from y.

Proof. The proof will be carried out by establishing the chain of implications (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4) $\Rightarrow$ (1). (1) $\Leftrightarrow$ (2) follows directly from Corollary 2.1 and the assumption that $\mathscr{S}$ is repetition-free.

(2) $\Rightarrow$ (3). Assuming (2), there exists a shortest u such that, for some a and b, either (i) or (ii) holds. Suppose it is (ii) that holds (the alternative case is handled similarly). Then u contains occurrences of $\bar{a}$ and $\bar{b}$ which are the mates of a_j and b_l, respectively. Assume without loss of generality that the occurrence of $\bar{a}$ is the earlier. Then u can be written as $u = u_1\bar{a}u_2\bar{b}u_3$. If u_2 is null, (3) directly follows from permutability. If u_2 is not null, it can be written $u_2 = cu_2'$, where $c \in \Sigma$. If c is an initiation symbol, then by permutability $\alpha_0 \cdot u_1c\bar{a}$ is defined; also, by commutativity and Definition 1.5, $\alpha_0 \cdot u_1c\bar{a} = \alpha_0 \cdot u_1\bar{a}c$; hence, by conclusion (a) of Lemma 2.4, $u_1c\bar{a}u_2'\bar{b}u_3$ is a prefix of an $\mathscr{I}$-computation. If c is a termination symbol, then it is the mate of some initiation symbol in u_1; hence, by persistence and the definition of an $\mathscr{I}$-computation, $\alpha_0 \cdot u_1c\bar{a}$ is defined. Also, since u is the shortest sequence giving rise to (2), $\alpha_0 \cdot u_1c\bar{a} = \alpha_0 \cdot u_1\bar{a}c$. Hence, $u_1c\bar{a}u_2'\bar{b}u_3$ is a prefix of an $\mathscr{I}$-computation. The symbol $\bar{a}$ can similarly be "slid" past each element of u_2, and we conclude that $u_1u_2\bar{a}\bar{b}$ is a prefix of an $\mathscr{I}$-computation; but, by permutability, $u_1u_2\bar{b}\bar{a}$ is also a prefix of an $\mathscr{I}$-computation; and, since $R(a) \cap R(b) \neq \varphi$, $a\rho b$. Thus, (3) follows.

(3) $\Rightarrow$ (4). By persistence, $w\bar{a}\bar{b}$ and $w\bar{b}\bar{a}$ are prefixes of some $\mathscr{I}$-computations x and y; but, clearly, $\mathscr{E}_{\bar{a}\bar{b}}(x) \neq \mathscr{E}_{\bar{a}\bar{b}}(y)$.

(4) $\Rightarrow$ (1). We may assume (by Lemma 2.1) that $\mathscr{I}$ is one–one. Suppose $\mathscr{E}_{\bar{a}\bar{b}}(x) \neq \mathscr{E}_{\bar{a}\bar{b}}(y)$; and assume, for contradiction, that $\mathscr{S}$ is determinate. Let $z = u_1\bar{c}u_2c_mu_3$ be an $\mathscr{I}$-computation in which the indicated instances of $\bar{c}$ and c_m are mates. Since determinacy is assumed, the "sliding" argument of Theorem 2.1 applies, and shows that $u_1\bar{c}c_mu_2u_3$ is also an $\mathscr{I}$-computation. By an induction based on the validity of this interchange process, there follows the existence of an $\mathscr{I}$-computation $\tilde{x}$ related to x as follows:

(a) $\tilde{x}$ has the same length as x (it is infinite if x is);

(b) for $k = 1, 2, \ldots$

$(\tilde{x})_{2k-1}$ is the kth initiation symbol in x and $(\tilde{x})_{2k}$ is the mate (in x) of the kth initiation symbol in x.

For example, if x were $\bar{a}\bar{b}\bar{c}b_2c_3\bar{a}a_1\bar{d}a_2\ldots$, then $\tilde{x}$ would be $\bar{a}a_1\bar{b}b_2\bar{c}c_3\bar{a}a_2\bar{d}\ldots$. An $\mathscr{I}$-computation $\tilde{y}$ can similarly be derived from y, and, clearly, $\mathscr{E}_{\bar{a}\bar{b}}(\tilde{x}) = \mathscr{E}_{\bar{a}\bar{b}}(x) \neq \mathscr{E}_{\bar{a}\bar{b}}(y) = \mathscr{E}_{\bar{a}\bar{b}}(\tilde{y})$. Now since $a\rho b$, either $R(a) \cap R(b) \neq \varphi$ or $R(a) \cap D(b) \neq \varphi$ or $R(b) \cap D(a) \neq \varphi$. These three cases can all be treated by similar methods; we discuss only the first case in detail. Choose $i \in R(a) \cap R(b)$, and let $S_i = \{c \mid i \in R(c)\}$. Let $\mathscr{E}_{S_i}(\tilde{x})$ denote the subsequence of $\tilde{x}$ consisting of initiation symbols for operations in S_i, and let $\mathscr{E}_{S_i}(\tilde{y})$ be defined similarly. Also, since $\mathscr{E}_{\bar{a}\bar{b}}(\tilde{x}) \neq \mathscr{E}_{\bar{a}\bar{b}}(\tilde{y})$, and since $\{a, b\} \subseteq S_i$, it follows that, for some p, $[\mathscr{E}_{S_i}(x)]_p \neq [\mathscr{E}_{S_i}(\tilde{y})]_p$. But since initiations and terminations are paired in $\tilde{x}$ and in $\tilde{y}$, it follows that the pth elements of $\Omega_i(\tilde{x})$ and $\Omega_i(\tilde{y})$ are produced by different operations. Hence, since $\mathscr{I}$ is one–one, $\Omega_i(\tilde{x}) \neq \Omega_i(\tilde{y})$, and $\mathscr{S}$ is indeterminate. Similar conclusions are reached in the other two cases ($R(a) \cap D(b) \neq \varphi$ and $R(b) \cap D(a) \neq \varphi$). Hence, (4) $\Rightarrow$ (1).

In Section V, condition (3) of Corollary 2.2 will play a central part in the development of an effective test for the determinacy of a certain class of schemata.

III. A CHARACTERIZATION OF $p(\mathscr{S})$

As we continue to discuss properties of schemata which do not depend upon the particular interpretation to be considered, it is convenient to discuss computations of schemata without giving an interpretation $\mathscr{I}$. Thus, a finite or infinite word x over the alphabet Σ is called a *computation* if it is an $\mathscr{I}$-computation for some interpretation $\mathscr{I}$. Let $p(\mathscr{S})$ denote the set of all finite words which are prefixes of computations for $\mathscr{S}$; then, as the following lemma shows, $p(\mathscr{S})$ determines the set of all computations for $\mathscr{S}$.

LEMMA 3.1. *Let $\mathscr{S}$ be a schema and x an element of $\Sigma^* \cup \Sigma^\omega$. Then x is a computation for $\mathscr{S}$ if and only if x satisfies the following three conditions:*

1) *For all k, ${}_kx \in p(\mathscr{S})$.*

2) *If x is finite, then $x\sigma \notin p(\mathscr{S})$ for any $\sigma \in \Sigma$.*

3) *If $x \in \Sigma^\omega$, then, for every $\sigma \in \Sigma$, and for every n, the following implication holds: If $({}_kx)\,\sigma \in p(\mathscr{S})$ for all $k \geqslant n$, then, for some $u \in \Sigma^*$, $({}_nx)\,u\sigma$ is a prefix of x.*

The proof of this lemma is obtained quite simply by relating its three conditions to the three conditions of Definition 1.6.

In this section we give a convenient characterization of $p(\mathscr{S})$ in a form which does not make reference to interpretations. Most of the decision procedures of Section IV are obtained by expressing the property to be tested as a property of $p(\mathscr{S})$ and then using this characterization.

Let $x = x_1x_2 \cdots x_t$ be an element of Σ^*. For $i \in M$ and $1 \leqslant k \leqslant t$, let

$$n(i, k) = \max(\{0\} \cup \{r \mid r \leqslant k \quad \text{and} \quad x_r = c_j\,, \quad \text{where} \quad c \in A \quad \text{and} \quad i \in R(c)\}).$$

Then, $n(i, k)$ locates the last occurrence (if any exist) in $x_1x_2 \cdots x_k$ of a symbol for the termination of an operation whose range locations include location i.

We now define an equivalence relation $\equiv$ between occurrences of initiation symbols in x as follows:

$x_k \equiv x_l$ if for some a $x_k = x_l = \bar{a}$ and

(i) $k = l$ or

(ii) for each $i \in D(a)$ either $n(i, k) = n(i, l) = 0$, or both are nonzero and the mate of $x_{n(i,k)}$ and the mate of $x_{n(i,l)}$ are equivalent in the relation $\equiv$.

Thus, two occurrences of $\bar{a}$ are related in $\equiv$ if the initiations which they represent necessarily operate on the same data.

The characterization of $p(\mathscr{S})$ is given in the following theorem.

THEOREM 3.1. *Let x be an element of Σ^*. Then $x \in p(\mathscr{S})$ if and only if the following three properties hold.*

Property 1. *For each $a \in A$, each prefix of x contains at least as many occurrences of $\bar{a}$ as it contains occurrences of termination symbols for a.*

Property 2. *$\tau(q_0, x)$ is defined.*

Property 3. *If $x_k \equiv x_l$ and both x_k and x_l have mates in x, then these mates are equal.*

The proof of this theorem is straightforward. The necessity of these properties is clear. Property 1 is necessary since by (2) of the definition of $\cdot$, $\alpha \cdot a_j$ is defined only when $\mu(a)$ is nonempty. Property 2 is necessary by condition (i) of the definition of an $\mathscr{I}$-computation. Property 3 states in a formal way that, if an operation is performed twice on the same data, then the outcomes of the two performances must be the same. This property is required because the functions G_a are single-valued. The sufficiency follows from the explicit construction of a one-one interpretation $\mathscr{I}$ such that x is a prefix of an $\mathscr{I}$-computation.

The characterization of $p(\mathscr{S})$ simplifies when $\mathscr{S}$ is repetition-free. It is immediate that $\mathscr{S}$ is repetition-free if and only if the relation $\equiv$ is trivial; i.e., that $x_k \equiv x_l$ if and only if $k = l$.

COROLLARY 3.1. *If $\mathscr{S}$ is repetition-free, then Properties* 1 *and* 2 *of Theorem 3.1 are sufficient, as well as necessary, for x to be an element of $p(\mathscr{S})$.*

Proof. If $\mathscr{S}$ is repetition-free, then no two distinct initiation symbols are in the relation $\equiv$ and Property 3 holds vacuously.

A schema is called *finite-state* if its control has a finite number of states.

COROLLARY 3.2. *Let $\mathscr{S}$ be a repetition-free finite-state schema. Then $p(\mathscr{S})$ is a regular event if and only if $\mathscr{S}$ is bounded.*

Proof. By Theorem 2 of [9], $p(\mathscr{S})$ is a regular event if and only if the following equivalence relation over Σ^* is of finite index:

$$xEy \Leftrightarrow \text{for all } w, \qquad xw \in p(\mathscr{S}) \Leftrightarrow yw \in p(\mathscr{S}).$$

Let $\Delta_a(x)$ equal the number of occurrences of $\bar{a}$ in x minus the number of occurrences of terminations $a_1, \dots, a_{K(a)}$ in x. Then Δ_a assumes infinitely many values for some a if and only if $\mathscr{S}$ is not bounded. Now observe

(1) If $\Delta_a(x) > \Delta_a(y)$, then $x \not{E} y$ since, for some string w consisting of $\Delta_a(x)$ terminations of a, $xw \in p(\mathscr{S})$ but $yw \notin p(\mathscr{S})$.

(2) If $\Delta_a(x) = \Delta_a(y)$ for all a, and $\tau(q_0, x) = \tau(q_0, y)$ then xEy, since Properties 1 and 2 of Theorem 1 clearly hold for xw if and only if they hold for yw.

From (1), E is of infinite index whenever $\mathscr{S}$ is not bounded; and, from (2), E is of finite index whenever $\mathscr{S}$ is bounded.

Although Corollary 3.2 is not used subsequently, we include it because it relates the degree of parallelism of a schema $\mathscr{S}$ to the structure of $p(\mathscr{S})$, and hence, to the complexity of the mechanisms required to control the sequencing of $\mathscr{S}$.

The following example shows that the hypothesis that $\mathscr{S}$ be repetition-free in Corollary 3.2 cannot be omitted.

Example 3.1. Figure 3.1 illustrates the control $\mathscr{T}$ for a finite-state schema $\mathscr{S}$.

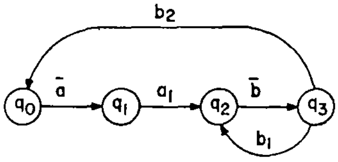


FIG. 3.1

The transitions not shown are either undefined or do not occur in computations. To complete the definition of $\mathscr{S}$,

Let $A = \{a, b\}$, $K(a) = 1$, $K(b) = 2$,

$$M = \{1, 2\}, \quad D(a) = \{1\}, \quad R(a) = D(b) = R(b) = \{2\}.$$

This schema is serial and, therefore, bounded.

The outcome of b is determined by the contents of memory location 2, which in turn is determined by the number of performances of b since the last performance of a. Hence, there is no computation which, for any integer k, contains both

$$\bar{a}a_1(\bar{b}b_1)^k\bar{b}b_2$$

and

$$\bar{a}a_1(\bar{b}b_1)^k\bar{b}b_1$$

as substrings of consecutive symbols. It can readily be seen that, because of this condition, $p(\mathscr{S})$ is not regular (in fact, it is not context-free).

IV. DECISION PROBLEMS

Up to this point, we have discussed relationships among properties of schemata without being concerned about effective tests of these properties. We now turn to questions concerning the existence of such tests. For this purpose, we restrict attention to the counter schemata, a special class of schemata having a finite presentation. The class of counter schemata includes the finite-state schemata, as well as the parallel flowcharts introduced in Section V.

The results given in this section indicate the extent to which effective procedures can be developed for the analysis of counter schemata. Most of the results are positive, and consist of decision procedures applicable to counter schemata with certain additional properties. On the negative side, it is shown that the equivalence problem for serial finite-state schemata is recursively unsolvable. Also, two decision problems are left as open questions.

The fundamental step in the development of our decision procedures is the introduction of a simple geometric structure called a vector addition system, and the proof that certain questions about vector addition systems are decidable. Once this has been accomplished, decision procedures for counter schemata are obtained by applying the following strategy:

(1) using Theorem 3.1, a property of $\mathscr{S}$ is restated as a property of $p(\mathscr{S})$;

(2) the property of $p(\mathscr{S})$ is then expressed as a decidable property of a vector addition system.

DEFINITION 4.1. A *counter schema* is a schema $\mathscr{S} = (M, A, \mathscr{T})$ in which the control $\mathscr{T}$ is specified in terms of the following entities:

a nonnegative integer k;
a finite set Σ;
a finite set S with a distinguished element s_0;
a vector $\pi \in N^k$, where N denotes the nonnegative integers;
a function v from Σ into N^k such that if $\sigma \in \Sigma_t$, then $v(\sigma) \geqslant 0$;
a partial function $\theta : S \times \Sigma \to S$ which is total on $S \times \Sigma_t$.

$\mathscr{T} = (Q, q_0, \Sigma, \tau)$ where
$Q = S \times N^k$
$q_0 = (s_0, \pi)$
$\tau((s, x), \sigma)$ is defined if $\theta(s, \sigma)$ is defined and $x + v(\sigma) \geqslant 0$; in that case, $\tau((s, x), \sigma) = (\theta(s, \sigma), x + v(\sigma))$.

Thus, a state of the control of a counter schema consists of a finite part together with the values of k "counters," each of which holds a nonnegative integer. Each initiation or termination causes these values to be incremented or decremented.

A. *Vector addition systems*

Because vector addition systems underly all of our decision procedures, we begin by discussing these systems as mathematical structures in their own right. These structures are of independent interest, and, in addition to their relevance to program schemata, arise from such subjects as the word problem for commutative semigroups (this connection has been exploited by M. Rabin), the theory of bounded context-free languages [*1*] and the theory of uniform recurrence equations [*5*].

An r-dimensional *vector addition system* is a pair $\mathscr{W} = (d, W)$ in which d is an r-dimensional vector of nonnegative integers, and W is a finite set of r-dimensional integer vectors. The *reachability set* $R(\mathscr{W})$ is the set of all vectors of the form $d + w_1 + w_2 + \cdots + w_s$ such that

$$w_i \in W \qquad i = 1,2,\ldots, s$$

and

$$d + w_1 + w_2 + \cdots + w_i \geqslant 0 \qquad i = 1, 2,\ldots, s.$$

Thus a point is in $R(\mathscr{W})$ if it can be reached from d by a sequence of displacements in the set W in such a way that each intermediate point is in the first orthant of r-space.

We shall prove that certain questions about reachability sets are recursively solvable, and for this purpose we introduce the following terminology:

(1) The relation $\leqslant$ between r-dimensional vectors is defined as follows: $y \leqslant z$ if and only if $y_i \leqslant z_i$, $i = 1, 2,\ldots, r$;

(2) 0 sometimes denotes the zero vector;

(3) ω is a symbol such that, if n is an integer, then $n < \omega$ and $n + \omega = \omega$;

(4) A *rooted tree* is a directed graph such that one vertex (the *root* δ) has no edges directed into it, each other vertex has exactly one edge directed into it, and each vertex is reachable from the root.

If ξ and η are distinct vertices of a rooted tree, and there is a path from ξ to η, then we say $\xi \prec \eta$; if there is an edge from ξ to η, then η is a *successor* of ξ. A vertex without successors is called an *end*.

We shall associate with any vector addition system $\mathscr{W}$ a rooted tree $\mathscr{T}(\mathscr{W})$, and shall give a rule for labelling each vertex ξ with an r-dimensional vector $l(\xi)$ whose coordinates are elements of $N \cup \{\omega\}$. $\mathscr{T}(\mathscr{W})$ and the function $l(\xi)$ are defined recursively according to the following rules.

(1) The root is labelled d;

(2) Let η be a vertex;

(a) if, for some vertex $\xi \prec \eta$, $l(\xi) = l(\eta)$, then η is an end;

(b) otherwise, the successors of η are in one-to-one correspondence with the elements $w \in W$ such that $0 \leqslant l(\eta) + w$. Let the successor of η corresponding to w

be denoted by η_w. For each i the ith coordinate of the label $l(\eta_w)$ is determined as follows:

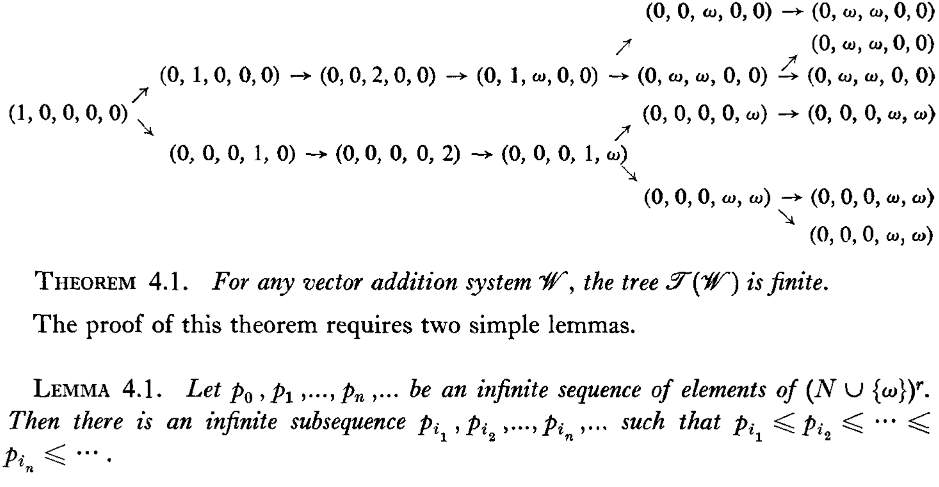
(i) if there exists $\xi < \eta_w$ such that $l(\xi) \leqslant l(\eta) + w$ and $l(\xi)_i < (l(\eta) + w)_i$ then $l(\eta_w)_i = \omega$;

(ii) if no such ξ exists, then $l(\eta_w)_i = (l(\eta) + w)_i$.

Example 4.1. To illustrate the construction of $\mathscr{T}(\mathscr{W})$, consider the vector addition $\mathscr{W} = (d, W)$ where

$$\begin{aligned} d &= (1, 0, 0, 0, 0) \\ W &= \{(-1, 1, 0, 0, 0), (-1, 0, 0, 1, 0), (0, -1, 2, 0, 0), \\ &\qquad (0, 1, -1, 0, 0), (0, 0, 0, -1, 2), (0, 0, 0, 1, -1)\} \end{aligned}$$

$\mathscr{T}(\mathscr{W})$:


(0, 0, ω, 0, 0) → (0, ω, ω, 0, 0)
(0, ω, ω, 0, 0)
(0, 1, 0, 0, 0) → (0, 0, 2, 0, 0) → (0, 1, ω, 0, 0) → (0, ω, ω, 0, 0) → (0, ω, ω, 0, 0)
(1, 0, 0, 0, 0)
(0, 0, 0, 0, ω) → (0, 0, 0, ω, ω)
(0, 0, 0, 1, 0) → (0, 0, 0, 0, 2) → (0, 0, 0, 1, ω)
(0, 0, 0, ω, ω) → (0, 0, 0, ω, ω)
(0, 0, 0, ω, ω)

THEOREM 4.1. *For any vector addition system $\mathscr{W}$, the tree $\mathscr{T}(\mathscr{W})$ is finite.*

The proof of this theorem requires two simple lemmas.

LEMMA 4.1. *Let $p_0, p_1, \ldots, p_n, \ldots$ be an infinite sequence of elements of $(N \cup \{\omega\})^r$. Then there is an infinite subsequence $p_{i_1}, p_{i_2}, \ldots, p_{i_n}, \ldots$ such that $p_{i_1} \leqslant p_{i_2} \leqslant \cdots \leqslant p_{i_n} \leqslant \cdots$.*

Proof. Extract an infinite subsequence nondecreasing in the first coordinate, extract from this an infinite subsequence nondecreasing in the second coordinate, and so forth.

LEMMA 4.2. (König Infinity Lemma [6]) *Let T be a rooted tree in which each vertex has only a finite number of successors and there is no infinite path directed away from the root. Then T is finite.*

Proof of Theorem 4.1. If $\delta, \eta_1, \eta_2, \ldots$ is the sequence of successive vertices in an infinite path directed away from the root, then, by Lemma 4.1, there is an infinite subsequence $\eta_{i_1}, \eta_{i_2}, \ldots, \eta_{i_n}, \ldots$ such that $l(\eta_{i_1}) \leqslant l(\eta_{i_2}) \leqslant \cdots \leqslant l(\eta_{i_n}) \leqslant \cdots$. Since none of these vertices is an end, we never have $l(\eta_{i_n}) = l(\eta_{i_{n+1}})$ (otherwise, by clause 2(a) in the definition of $\mathscr{T}(\mathscr{W})$, the path would be finite). Thus, by 2(b), $\eta_{i_{n+1}}$ has at least one more coordinate equal to ω than η_{i_n} does. Since the number of coordinates is finite, we have reached a contradiction, and therefore, no infinite path exists. Thus, by Lemma 4.2, $\mathscr{T}(\mathscr{W})$ is finite.

We remark that, since $\mathscr{T}(\mathscr{W})$ is finite, its construction, using the recursive definition, is clearly effective.

The following theorem is the key to several decision problems concerning vector addition systems.

THEOREM 4.2. *Let x be an r-dimensional vector of nonnegative integers. Then the following statements are equivalent:*

(1) *there is a $y \in R(\mathscr{W})$ such that $x \leqslant y$;*

(2) *there is a vertex $\eta \in \mathscr{T}(\mathscr{W})$ such that $x \leqslant l(\eta)$.*

Proof. We begin by proving (2) $\Rightarrow$ (1). The idea of the proof is that, if ξ is a vertex in $\mathscr{T}(\mathscr{W})$, then there are vectors in $R(\mathscr{W})$ which agree with $l(\xi)$ in its finite coordinates, and can be made arbitrarily large in the coordinates equal to ω by repetition of the sequence of vectors which led to the occurrence of ω. The details of the construction involve some calculation. Suppose $x \leqslant l(\eta)$. Let the path from δ to η have the successive vertices $\delta = \xi_0, \xi_1, \dots, \xi_k = \eta$. For $l = 1, \dots, k$, let v_l be the vector associated with the edge directed into ξ_l; i.e., $\xi_l = (\xi_{l-1})_{v_l}$. Assume without loss of generality that the first h components of $l(\xi)$ are equal to ω, and that the other components are less than ω. Assume further that, in the path from δ to ξ, ω's are introduced in the order $1, 2, \dots, h$. Then, for each i, $1 \leqslant i \leqslant h$ there exists a consecutive subsequence $t_i = v_{c(i)}, v_{c(i)+1}, v_{c(i)+2}, \dots, v_{n(i)}$ such that the vector $u_i = v_{c(i)} + v_{c(i)+1} + \cdots + v_{d(i)}$ is positive in the ith coordinate and nonnegative in the $(i+1)$st through nth coordinates. (t_i is the subsequence which "accounts for" the ith ω.) Let $-\Delta$ be a lower bound on all the (negative) coordinates of $u_1, \dots, u_h$. Let $\{n_1, n_2, \dots, n_h\}$ be any set of nonnegative integers satisfying:

$$n_i \geqslant (x-d)_i + \Delta(h + 2 - i + n_{i+1} + \cdots + n_h), \qquad 1 \leqslant i \leqslant k. \tag{1}$$

Such a set certainly exists because of the triangular form of the system of inequalities.

Choose $s_1, s_2, \dots, s_{h+1}$ so that, for $1 \leqslant i \leqslant h$, $s_1 s_2 \cdots s_i$ is the prefix of $v_1 v_2 \cdots v_k$ up to the first occurrence of ω in coordinate i, and $s_1 s_2 \cdots s_{h+1} = v_1 v_2 \cdots v_k$. Then the sequence

$$\sigma = s_1 t_1^{n_1} s_2 t_2^{n_2} \cdots s_h t_h^{n_h} s_{h+1} = u_1 u_2 \cdots u_f$$

has the following properties:

(a) $d + u_1 + \cdots + u_f \geqslant x$

(b) each partial sum $d + u_1 + \cdots + u_i$ is nonnegative.

The detailed derivation of (a) and (b) from the system of inequalities (1) is omitted.

To complete the proof, suppose that

$$d + u_1 + \cdots + u_f \in R(\mathscr{W}), \qquad x \leqslant d + u_1 + \cdots + u_f,$$

and

$$d + u_1 + \cdots + u_m \geqslant 0, \qquad m = 1, 2, \ldots, f,$$

where the u_m are elements of W. Apply the following operation to the sequence $d, d + u_1, d + u_1 + u_2, \ldots, d + u_1 + u_2 + \cdots + u_f$ as many times as possible:

(i) find the first element of the sequence (call it u') such that, for some earlier element u'', $u'' \leqslant u'$.

(a) if $u'' = u'$, delete all elements following u';

(b) otherwise, for each i such that $(u'')_i < (u')_i$, replace the ith coordinate of u' and of each vector beyond u' in the sequence by ω.

Then the sequence obtained at the conclusion of this process is the sequence of labels in some path directed from the root of $\mathscr{T}(\mathscr{W})$, and the final label in this sequence is a vector greater than or equal to $d + u_1 + \cdots + u_f$. Hence, (2) is satisfied.

Example 4.2. This example illustrates the construction of the sequence σ given in the first half of the proof of Theorem 4.2. Suppose $d = (1, 1, 1, 4)$ and $W = \{(0, 0, 0, -1), (2, -1, 0, 0), (-1, 1, 0, 0), (-1, -3, 4, 0)\}$. Consider the following path in $\mathscr{T}(\mathscr{W})$:

$$(1, 1, 1, 4) \xrightarrow{0,0,0,-1} (1, 1, 1, 3) \xrightarrow{2,-1,0,0} (3, 0, 1, 3) \xrightarrow{-1,1,0,0} (\omega, 1, 1, 3)$$

$$\xrightarrow{-1,1,0,0} (\omega, \omega, 1, 3) \xrightarrow{-1,-3,4,0} (\omega, \omega, \omega, 3).$$

$$\begin{aligned} s_1 &= (0, 0, 0, -1), (2, -1, 0, 0), (-1, 1, 0, 0) & t_1 &= (2, -1, 0, 0), (-1, 1, 0, 0) \\ s_2 &= (-1, 1, 0, 0) & t_2 &= (-1, 1, 0, 0) \\ s_3 &= (-1, -3, 4, 0) & t_3 &= (-1, -3, 4, 0) \end{aligned}$$

Take $x = (22, 16, 9, 3) \leqslant (\omega, \omega, \omega, 3)$. Let $\Delta = 3$. The system of inequalities (1) for this case is:

$$\begin{aligned} n_1 &\geqslant 21 + 3(4 + n_2 + n_3) \\ n_2 &\geqslant 15 + 3(3 + n_3) \\ n_3 &\geqslant 8 + 3 \cdot 2 \end{aligned}$$

A solution is: $n_3 = 14$, $n_2 = 66$, $n_1 = 273$, giving the sequence $\sigma = (0, 0, 0, -1), (2, -1, 0, 0), (-1, 1, 0, 0), ((2, -1, 0, 0), (-1, 1, 0, 0))^{273}(-1, 1, 0, 0)(-1, 1, 0, 0)^{66}$ $(-1, -3, 4, 0)(-1, -3, 4, 0)^{14}$, which establishes that the point $(193, 23, 61, 3) \geqslant x$ is in $R(\mathscr{W})$.

The following corollaries will prove useful in the development of decision procedures for parallel program schemata.

COROLLARY 4.1. *It is decidable of a vector addition system* $\mathscr{W}$ *and a point* x *whether* $R(\mathscr{W})$ *contains a point* $y \geqslant x$.

COROLLARY 4.2. *It is decidable of an r-dimensional vector addition system* $\mathscr{W}$ *and a set* $\Theta \subseteq \{1, 2,\ldots, r\}$ *whether the coordinates in* Θ *are simultaneously unbounded*; i.e., *whether, for every* $x \in N^r$ *there exists a* $y \in R(\mathscr{W})$ *such that* $y_i \geqslant x_i$ *for all* $i \in \Theta$.

Proof. This property holds if and only if there is an end $\beta \in \mathscr{T}(\mathscr{W})$ such that, for all $i \in \Theta$, $(l(\beta))_i = \omega$.

COROLLARY 4.3. *It is decidable of a vector addition system* $\mathscr{W}$ *whether* $R(\mathscr{W})$ *is infinite.*

Lest the reader be left with the impression that all reasonable questions about $R(\mathscr{W})$ can be answered by inspecting $\mathscr{T}(\mathscr{W})$, we mention the following unpublished theorem of Michael Rabin.

THEOREM 4.3. *There is no algorithm to decide whether two vector addition systems* $\mathscr{W}$ *and* $\mathscr{W}'$ *have the same reachability set* (i.e., *whether* $R(\mathscr{W}) = R(\mathscr{W}')$).

Rabin obtains this result by showing that determining whether an exponential Diophantine equation has a solution (an undecidable question) can be reduced to deciding whether two reachability sets are equal.

B. *Decidable Problems*

In this section, we apply the results about vector addition systems to counter schemata and establish the decidability of some properties. To accomplish this, we introduce an "encoding" which associates a vector addition system with each counter schema.

Suppose that $\mathscr{S} = (M, A, \mathscr{T})$ is a counter schema specified as in Definition 4.1. The vector addition system $\mathscr{W}_{\mathscr{S}}$ will have $|S| + k + |A| + 2^{|A|}$ dimensions. In specifying $\mathscr{W}_{\mathscr{S}}$ it will be convenient to regard each vector as a function which associates an integer with each element of a set, which indexes the coordinates. In our case, we use the index set $S \cup \{1,\ldots, k\} \cup A \cup 2^A$. This sets up a coordinate for each element of S and each counter, as well as a coordinate for each operation and for each subset of operations. We define $\mathscr{W}_{\mathscr{S}} = (d, W)$ where:

$$\begin{aligned}
d(s_0) &= 1 && \\
d(s) &= 0 && s \in S, \quad s \neq s_0 \\
d(i) &= \pi_i && i = 1, 2,\ldots, k \\
d(\varnothing) &= 1 && \\
d(T) &= 0 && T \subseteq A, \quad T \neq \varnothing.
\end{aligned}$$

The elements of W are in one–one correspondence with the triples (s, σ, T) such that $\theta(s, \sigma)$ is defined and $T = A$. The vector w corresponding to (s, σ, T) is defined by:

$$\begin{aligned} w(t) &= \delta(t, \theta(s, \sigma)) - \delta(t, s),\text{[3]} \qquad && t \in S \\ w(i) &= [v(\sigma)]_i \qquad && i = 1, 2, \ldots, k \end{aligned}$$

If $\sigma = \bar{b} \in \Sigma_i$, then

$$\begin{aligned} w(a) &= \delta(a, b) \qquad && a \in A \\ w(U) &= \delta(U, T \cup \{b\}) - \delta(U, T) \qquad && U \subseteq A \end{aligned}$$

If $\sigma = b_j \in \Sigma_t$, then

$$\begin{aligned} w(a) &= -\delta(a, b) \qquad && a \in A \\ w(U) &= \delta(U, T \cap \{a \mid R(b) \cap D(a) = \varphi\}) - \delta(U, T) \qquad && U \subseteq A \end{aligned}$$

In explaining the rationale behind this construction, it is useful to introduce the set $p'(\mathscr{S}) \subseteq \Sigma^*$, consisting of all strings satisfying Properties 1 and 2 of Theorem 3.1. If $\mathscr{S}$ is repetition-free, then $p(\mathscr{S}) = p'(\mathscr{S})$; in general, $p(\mathscr{S}) \subseteq p'(\mathscr{S})$. For any string $x \in p'(\mathscr{S})$, let $\tau((s_0, \tau), x) = (s, \rho)$. Define a set $T_x \subseteq A$ by: $a \in T_x$ if x can be expressed as $x_1 \bar{a} x_2$, where neither $\bar{a}$ nor any termination symbol b_j such that $R(b) \cap D(a) \neq \varphi$ occurs in x_2. Thus, a repetition would result from initiating a after the sequence x if and only if $a \in T_x$. Define a vector w_x, with index set $S \cup \{1, 2, \ldots, k\} \cup A \cup 2^A$, as follows:

$$\begin{aligned} w_x(t) &= \delta(s, t) \qquad && t \in S \\ w_x(i) &= \rho_i \qquad && i = 1, 2, \ldots, k \\ w_x(a) &= \Delta_a(x) \qquad && a \in A \\ w_x(U) &= \delta(T_x, U) \qquad && U \subseteq A. \end{aligned}$$

Lemma 4.3. $R(\mathscr{W}_{\mathscr{S}}) = \{w_x \mid x \in p'(\mathscr{S})\}$.

We omit the simple inductive proof of this lemma. The lemma shows that each vector in the reachability set $R(\mathscr{W}_{\mathscr{S}})$ is an "encoding" of information about the instantaneous description which results from applying some hypothetical computation sequence $x \in p'(\mathscr{S})$.

Lemma 4.4. *If $\mathscr{S}$ is repetition-free, then*

$$R(\mathscr{W}_{\mathscr{S}}) = \{w_x \mid x \in p(\mathscr{S})\}.$$

Proof. If $\mathscr{S}$ is repetition-free, then $p(\mathscr{S}) = p'(\mathscr{S})$.

Theorem 4.4. *It is decidable whether a given counter schema is repetition-free.*

---

[3] For any two elements X and Y, $\delta(X, Y) = 1$ if $X = Y$, and 0 if $X \neq Y$.

Proof. From the definition of $p'(\mathcal{S})$ and T_x, it follows that $\mathcal{S}$ is repetition-free if and only if:

$$\text{for all } x \in p'(\mathcal{S}), \qquad a \in T_x \Rightarrow x\bar{a} \notin p'(\mathcal{S}).$$

By Lemma 4.3, this is equivalent to the following property of $\mathcal{W}_{\mathcal{S}}$:

For some $a \in A$, $T \subseteq A$ containing a, and $s \in S$ such that $\theta(s, a)$ is defined, there exists a vector $u \in R(\mathcal{W}_{\mathcal{S}})$, such that:

$$\begin{aligned} &u(T) = 1 \\ &u(s) = 1 \\ &u(i) \geqslant [v(\bar{a})]_i \qquad i = 1, 2, \ldots, k. \end{aligned}$$

It is a consequence of Lemma 4.3, that $u(s) \in \{0, 1\}$ for $s \in S$ and $u(T) \in \{0, 1\}$ for $T \subseteq A$. Hence, the above system is equivalent to

$$\begin{aligned} &u(T) \geqslant 1 \\ &u(s) \geqslant 1 \\ &u(i) \geqslant [v(\bar{a})]_i \qquad i = 1, 2, \ldots, k. \end{aligned}$$

But, by Corollary 4.1, the existence of such a u can be checked for each of the finitely many triples (a, T, s) in question.

In giving decision procedures for counter schemata known to be repetition-free, we use a vector addition system $\mathcal{V}_{\mathcal{S}} = (e, V)$, obtained from $\mathcal{W}_{\mathcal{S}} = (d, W)$ by restricting d and the elements of W to the index set $S \cup \{1, 2, \ldots, k\} \cup A$. We remark that $R(\mathcal{V}_{\mathcal{S}})$ is the restriction of $R(\mathcal{W}_{\mathcal{S}})$ to the index set $S \cup \{1, 2, \ldots, k\} \cup A$.

THEOREM 4.5. *It is decidable whether a given repetition-free counter schema $\mathcal{S}$ is commutative.*

Proof. Clearly, $\mathcal{S}$ is not commutative if and only if, for some triple $(s, \sigma, \sigma') \in S \times \Sigma \times \Sigma$ such that $\theta(s, \sigma, \sigma')$ and $\theta(s, \sigma'\sigma)$ are defined and unequal, there exists an x such that $\tau(q_0, x) = s$, $x\sigma\sigma' \in p(\mathcal{S})$, and $x\sigma'\sigma \in p(\mathcal{S})$. Using Lemma 4.4 and the relation between $\mathcal{W}_{\mathcal{S}}$ and $\mathcal{V}_{\mathcal{S}}$, this condition becomes:

for some triple $(s, \sigma, \sigma') \in S \times \Sigma \times \Sigma$ such that $\theta(s, \sigma\sigma')$ and $\theta(s, \sigma'\sigma)$ are defined and unequal, there exists a $u \in R(\mathcal{V}_{\mathcal{S}})$ such that $u(s) = 1$ and

$$u(i) \geqslant \max[v(\sigma)_i, v(\sigma')_i, (v(\sigma) + v(\sigma'))_i], \qquad i = 1, 2, \ldots, k.$$

Applying Corollary 4.1, we find that this condition can be checked for each of the finitely many triples (s, σ, σ') requiring consideration.

The technique of applying Corollary 4.1 as in Theorems 4.4 and 4.5 does not appear to carry over to testing whether a repetition-free counter schema is persistent or permutable. Short of decidability, however, we mention convenient sufficient conditions for the persistence and permutability of counter schemata. A counter schema $\mathscr{S}$ specified as in Definition 4.1 is permutable if $v(\bar{a}) \leqslant 0$ for each $a \in A$ and, whenever $\theta(s, \bar{a}, \bar{b})$ is defined, $\theta(s, \bar{b})$ is defined. $\mathscr{S}$ is persistent if, for distinct operations $a \in A$ and $b \in A$, $v(\bar{a}) < 0 \Rightarrow v(\bar{b})_i = 0$, and, whenever σ and π are distinct elements of Σ such that $\theta(s, \sigma)$ and $\theta(s, \pi)$ are defined, $\theta(s, \sigma\pi)$ is defined.

Further positive results can be gleaned using $\mathscr{V}_{\mathscr{S}}$.

THEOREM 4.6. *It is decidable whether a given repetition-free counter schema* $\mathscr{S}$ *is bounded.*[4]

Proof. It follows from Lemma 4.4 that $\mathscr{S}$ is bounded if and only if, for all $a \in A$, $\{u(a) \mid u \in R(\mathscr{V}_{\mathscr{S}})\}$ is bounded. By Corollary 4.2, this is decidable.

A more detailed result which follows similarly is the decidability of whether $\{\Delta_a(x) \mid x \in p(\mathscr{S})\}$ is finite for a given a. When this set is finite, one can effectively determine $\max_{x \in p(\mathscr{S})} \Delta_a(x)$.

COROLLARY 4.4. *It is decidable whether a given repetition-free counter schema is serial.*

DEFINITION 4.2. Let $\mathscr{S}$ be a schema. An operation $a \in A$ is *terminating* if a occurs only a finite number of times in each computation of $\mathscr{S}$.

THEOREM 4.7. *It is decidable of a repetition-free counter schema* $\mathscr{S}$ *whether a given operation* $a \in A$ *is terminating. Also, if* a *is terminating, one can effectively find the maximum number of initiations of* a *in a computation.*

Proof. One can construct a vector addition system $\mathscr{V}'^{a}_{\mathscr{S}}$ by adjoining to $\mathscr{V}_{\mathscr{S}}$ a coordinate which counts the initiations of a. Then a is terminating if and only if this coordinate is bounded. By Corollary 4.2, this is decidable and, if the coordinate is bounded, its maximum value can be read off the tree $\mathscr{T}(\mathscr{V}'_{\mathscr{S}})$.

From Definition 4.2 it might appear that an operation a in a repetition-free counter schema could be terminating without the existence of a uniform upper bound on the number of occurrences of $\bar{a}$ in a computation. The connection with vector addition systems establishes that this is impossible.

THEOREM 4.8. *It is decidable whether a given repetition-free counter schema* $\mathscr{S}$ *has only a finite number of accessible states* (i.e., *whether* $\mathscr{S}$ *is a finite-state schema*).

Proof. $\mathscr{S}$ is finite-state if and only if every counter is bounded. By Corollary 4.2, this is decidable.

The following result is one of the main theorems in the present paper.

---

[4] Following a suggestion of D. Slutz, we remark that every repetition-free, commutative, permutable, and persistent counter schema is bounded.

THEOREM 4.9. *It is decidable whether a repetition-free lossless, persistent, commutative counter schema is determinate.*

Proof. Applying Corollary 2.1 to the repetition-free case, we see that $\mathscr{S}$ is determinate if and only if there is no $x \in \Sigma^*$ such that $x\sigma\sigma' \in p(\mathscr{S})$ and $x\sigma'\sigma \in p(\mathscr{S})$, where (σ, σ') is one of a finite list of pairs of elements of Σ. This can be checked using $\mathscr{V}_{\mathscr{S}}$ in the manner described in the proof of Theorem 4.5.

The decision procedures given in Theorems 4.5–4.9 all require the hypothesis of repetition-freeness. There is an alternative property called strong boundedness, which can replace the repetition-freeness hypothesis in these theorems. The counter schema $\mathscr{S}$ is called *strongly bounded* if, in the vector addition system $\mathscr{W}_{\mathscr{S}}$, the coordinates indexed $1, 2,..., k$, corresponding to the k counters, are bounded. An interpretation of this property can be given. Let $\tilde{\mathscr{S}}$ be a schema obtained from $\mathscr{S}$ be modifying the domains and ranges of operations so that they are all equal and nonempty. Then $\tilde{\mathscr{S}}$ is repetition-free, $p(\tilde{\mathscr{S}}) = p'(\mathscr{S})$ and $\mathscr{W}_{\mathscr{S}} = \mathscr{W}_{\tilde{\mathscr{S}}}$. Applying Lemma 4.4 to $\tilde{\mathscr{S}}$, it follows that $\tilde{\mathscr{S}}$ is bounded if and only if the coordinates indexed $\{1, 2,..., k\}$ in $\mathscr{W}_{\tilde{\mathscr{S}}}$ are bounded; i.e., if and only if $\mathscr{S}$ is strongly bounded. Clearly, it is decidable whether $\mathscr{S}$ is strongly bounded, and the bounds on the coordinates can effectively be determined.

In general, decision problems about nonrepetition-free counter schemata are difficult (and probably undecidable in most cases) because it is necessary to keep track of repetitions, which constrain outcomes. Given strong boundedness, however, only a finite number of occurrences of any operation must be kept track of, and the necessary information can be encoded into the finitely many coordinates of a vector addition system. Without giving further details, we state that strong boundedness can replace repetition-freeness in Corollary 4.4. and Theorems 4.5, 4.7, 4.8, and 4.9.

This completes our catalog of decision problems which can be solved using vector addition systems. To conclude our enumeration of problems about counter schemata which can be solved constructively, we mention the following result.

THEOREM 4.10. *There is an algorithm to construct from a given determinate counter schema $\mathscr{S}$ an equivalent serial counter schema $\mathscr{S}'$. If $\mathscr{S}$ is repetition-free, then $\mathscr{S}'$ is.*

Proof. Let S be specified as in Definition 4.1. Assume $A = (a^{(1)},..., a^{(h)})$. To obtain $\mathscr{S}'$ we replace S, s_0, and θ by S', s_0', and θ', defined as follows:

$$S' = S \times \{1, 2,..., h\} \times \{0, 1\}; \qquad s_0' = (s_0, 1, 0); \qquad \theta'(\langle s, \alpha, \beta\rangle, \bar{a}^{(i)})$$

is defined if and only if:

(i) $\theta(s, \bar{a}^{(i)})$ is defined;

(ii) $\beta = 0$;

(iii) for all $k \in \{\alpha, \alpha + 1 (\mathrm{mod}\, h), \alpha + 2 (\mathrm{mod}\, h), ..., i - 1 (\mathrm{mod}\, h)\}$, $\theta(q, \bar{a}^{(k)})$ is undefined.

If $\theta'(\langle s, \alpha, \beta\rangle, \bar{a}^{(i)})$ is defined, it is equal to $\theta(s, \bar{a}^{(i)}, \alpha + 1 (\mathrm{mod}\, h), 1)$. Finally, $\theta'(s, \alpha, \beta, a_j^{(i)}) = \langle\theta(s, a_j^{(i)}), \alpha, 0\rangle$.

The coordinate β ensures that initiations and terminations alternate, so that $\mathscr{S}'$ is serial. The coordinate α counts cyclically through the operation indices, ensuring that

(1) at most one operation is eligible for initiation;

(2) if $\tau(s_0, x\bar{a}^{(i)})$ is defined and $\theta(s_0, x)$ is also defined, then the control of $\mathscr{S}'$ will allow the initiation of $\bar{a}^{(i)}$ following x after at most $h - 1$ other initiations have intervened.

It follows that, for each interpretation $\mathscr{I}$, $\mathscr{S}'$ has exactly one $\mathscr{I}$-computation; moreover, that $\mathscr{I}$-computation is also an $\mathscr{I}$-computation for $\mathscr{S}$ (in particular, the cyclic queue discipline enforced by the coordinate α guarantees the finite delay property).

C. *An undecidable problem*

In this subsection we prove that the equivalence problem for finite-state schemata is recursively unsolvable. In fact, the equivalence problem is unsolvable both for the class of persistent finite-state schemata and for the class of serial finite-state schemata. Since the counter schemata include the finite-state schemata, the equivalence problem for counter schemata is unsolvable. Only one case is known in which the equivalence problem is solvable. This is the case of decision-free flowcharts, a class of counter schemata discussed in Section VI.

The proofs of the undecidability results involve the construction of suitable indeterminate schemata. Thus the decidability of the equivalence problem for determinate finite-state schemata remains an open problem. The question of the existence of an effective test of determinacy for the class of finite-state schemata is also open, although Theorem 4.9, applied to finite-state schemata, settles this question for an important subclass.

Our proofs will use the undecidability of the Post correspondence problem [8] in a manner reminiscent of its application to two-tape automata by Rabin and Scott [9]. The correspondence problem is the following: Given two n-tuples $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$ of words over an alphabet Γ, to decide whether there exists a sequence of indices $i_1, i_2, ..., i_p$ such that $x_{i_1} x_{i_2} \cdots x_{i_p} = y_{i_1} y_{i_2} \cdots y_{i_p}$. This problem is undecidable for any alphabet Γ with more than one letter; that is, no single algorithm can settle all instances of the correspondence problem over Γ.

We shall give a method of converting instances of the correspondence problem over the alphabet $\{b_1, b_2\}$ into instances of an equivalence problem between schemata. This will suffice to prove the unsolvability of the equivalence problem.

Let us give some preliminary remarks to motivate the conversion process. Let $\mathscr{S}$ be any schema such that: $M = \{1, 2\}$, $A = \{a, b\}$, $D(a) = R(a) = \{1\}$, $K(a) = 3$, $D(b) = R(b) = \{2\}$, $K(b) = 3$. Note that neither operation affects the domain locations of the other; thus the sequence of outcomes of operation a depends on the interpretation $\mathscr{I}$, but does not depend on the manner in which initiations and terminations of b are interspersed among the initiations and terminations of a; the kth element of this sequence of outcomes is $G_a(F_a^{[k-1]}(\Pi_1(c_0)))$. Similarly, the sequence of outcomes of b is determined entirely by the interpretation.

Now let $X = x_1, x_2, \ldots, x_n$ be an n-tuple of words over the alphabet $\{b_1, b_2\}$. We shall call an interpretation $\mathscr{I}$ *consistent* with the pair $(X; i_1, i_2, \ldots, i_p)$, where $i_1, i_2, \ldots, i_p$ is a sequence of indices, provided that

(i) if a is executed repeatedly, beginning with the initial $\mathscr{I}$-instantaneous description, the sequence of outcomes will have as a prefix:

$$a_1^{i_1-1}a_2a_1^{i_2-1}a_2 \cdots a_1^{i_p-1}a_2a_3$$

and

(ii) if b is executed repeatedly, beginning with the initial $\mathscr{I}$-instantaneous description, the sequence of outcomes will have as a prefix:

$$x_{i_1}x_{i_2} \cdots x_{i_p}b_3 .$$

Now let $Y = y_1, y_2, \ldots, y_n$ be a second n-tuple of words over the alphabet $\{b_1, b_2\}$. Then, clearly, the instance of the correspondence problem determined by X and Y has a solution if and only if, for some sequence $i_1, i_2, \ldots, i_n$, there is an interpretation $\mathscr{I}$ consistent with both $(X; i_1, i_2, \ldots, i_p)$ and $(Y; i_1, i_2, \ldots, i_p)$.

We shall indicate a construction which converts this restatement of the correspondence problem into a question of equivalence of finite-state schemata. We introduce determinate schemata $\mathscr{S}(X)$, $\bar{\mathscr{S}}(X)$, $\mathscr{S}(Y)$ and $\bar{\mathscr{S}}(Y)$, each having $M = \{1, 2\}$, $A = \{a, b\}$, $D(a) = R(a) = \{1\}$, $K(a) = 3$, $D(b) = R(b) = \{2\}$, and $K(b) = 3$. If $\mathscr{I}$ is consistent with $(X; i_1, i_2, \ldots, i_p)$, then $\mathscr{S}(X)$ has a unique, finite $\mathscr{I}$-computation in which

(i) the sequence of outcomes of a is

$$a_1^{i_1-1}a_2a_1^{i_2-1}a_2 \cdots a_1^{i_p-1}a_2a_3 \quad \text{and}$$

(ii) the sequence of outcomes of b is

$$x_{i_1}x_{i_2} \cdots x_{i_p}b_3 .$$

If $\mathscr{I}$ is not consistent with any pair of the form $(X; i_1, i_2, \ldots, i_p)$, then each $\mathscr{I}$-computation has infinitely many occurrences of $\bar{a}$ and of $\bar{b}$.

The schema $\mathscr{S}(X)$ proceeds by executing a repeatedly and checking for the sequence of outcomes $a_1^{i-1}a_2$, then executing b repeatedly and checking for the sequence of outcomes x_i. Any deviation from such outcomes before the first occurrence of a_3 forces the schema into a nonterminating cycle. Further, if the first outcome of b, after a_3, is not b_3, then the schema also cycles.

The following example illustrates the construction of $\mathscr{S}(X)$.

Example 4.3. Let $n = 3$, $x_1 = b_1$, $x_2 = b_2$, and $x_3 = b_1b_1$. Then the control of $\mathscr{S}(X)$ is given by the following state-transition diagram of Figure 4.1.

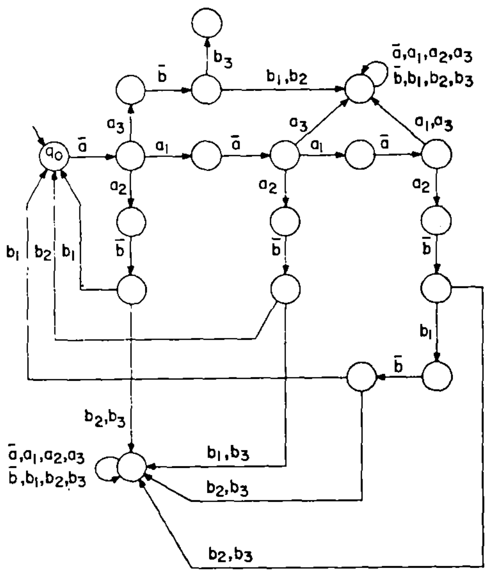


FIG. 4.1

The schema $\mathscr{S}(X)$ has the following properties:

(a) if $\mathscr{I}$ is consistent with $(X; i_1, i_2, ..., i_p)$ then every $\mathscr{I}$-computation of $\mathscr{S}(X)$ has infinitely many occurrences of both $\bar{a}$ and $\bar{b}$; the (unique, finite) $\mathscr{I}$-computation for $\mathscr{S}(X)$ is a prefix of each of these computations;

(b) if there is no sequence of indices $i_1, i_2, ..., i_p$ such that $\mathscr{I}$ is consistent with $(X; i_1, ..., i_p)$ then, in any $\mathscr{I}$-computation, either

(i) the sequence of outcomes of a is infinite and contains no occurrence of a_3 or

(ii) the sequence ends with the first occurrence of a_3.

Similarly, the sequence of outcomes of b is infinite and contains no occurrence of b_3, or else this sequence ends with the first occurrence of b_3.

Now let $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_n$, and let c and d be two operations such that $(R(c) \cap R(d)) \cup (R(c) \cap D(d)) \cup (D(c) \cap R(d)) = \varphi$. Consider the two persistent schemata shown in Figures 4.2 and 4.3.

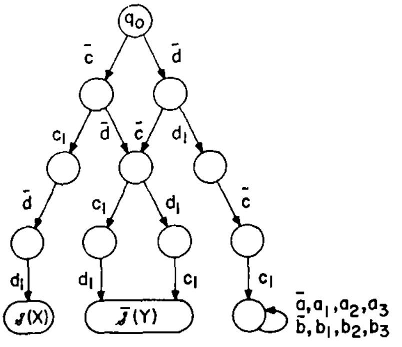


FIG. 4.2

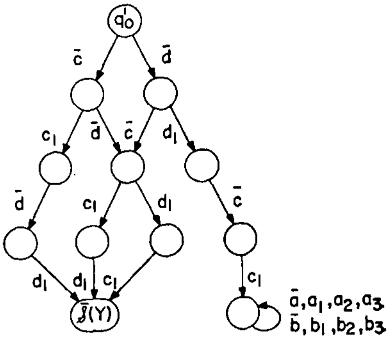


FIG. 4.3

Clearly $\mathscr{V}$ and $\mathscr{V}'$ fail to be equivalent if and only if, for some interpretation $\mathscr{I}$, $\mathscr{S}(X)$ has a finite $\mathscr{I}$-computation and $\mathscr{S}(Y)$ has none; and this is possible if and only if, for some sequence $i_1, \ldots, i_p$,

$$x_{i_1} x_{i_2} \cdots x_{i_p} = y_{i_1} y_{i_2} \cdots y_{i_p} .$$

Thus, every instance of the Post correspondence problem over the alphabet $\{b_1, b_2\}$ can be reduced to the question of equivalence between two persistent finite-state schemata. Hence, there cannot be an algorithm for this class of equivalence questions, since the existence of such an algorithm would imply the solvability of the correspondence problem.

Figure 4.4 shows a slight modification of the above constructions which, in a similar way, establishes the unsolvability of equivalence between serial (but not persistent) finite-state schemata. In this construction, any two of the sets $D(c)$, $D(d)$, $D(e)$, $R(c)$, $R(d)$, $R(e)$ are disjoint.

Our results may be summarized in the following theorems.

THEOREM 4.11. *It is undecidable whether two persistent finite-state schemata are equivalent.*

THEOREM 4.12. *It is undecidable whether two serial finite-state schemata are equivalent.*

Since the above constructions produce nondeterminate schemata, this method of proof does not establish the unsolvability of equivalence between determinate

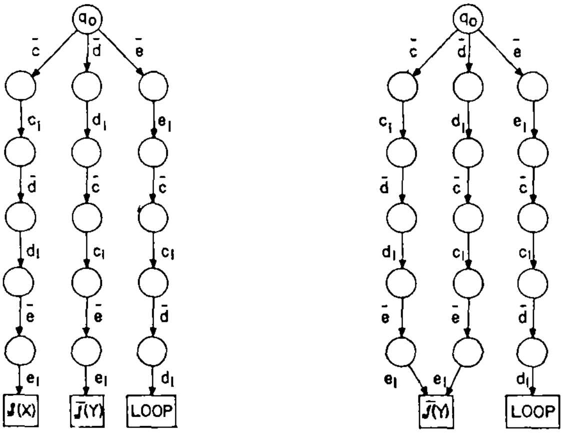


FIG. 4.4

schemata, and this question remains open. Luckham, Park, and Paterson [7] have proved the equivalence problem unsolvable for a class of schemata related to our serial determinate finite-state schemata. In their model a distinction is made between instructions and statements (the latter corresponding to our operations), and several statements with different domain or range locations may be associated with the same instruction. In terms of our model, this amounts to restricting the class of interpretations to those in which, for some pairs a, b of operations, the functions F_a and G_a are identical with, respectively, F_b and G_b. We have found no way of adapting their proof to the present model.

V. PARALLEL FLOWCHARTS

In this section we introduce a special class of counter schemata called parallel flowcharts. We suggest a convenient graphical representation of parallel flowcharts and then informally discuss their adequacy for the representation of parallel algorithms.

DEFINITION 5.1. A *(parallel) flowchart* is a counter schema in which:

(1) $S = \{s_0\}$;

(2) $\theta(s_0, \sigma)$ is defined for all $\sigma \in \Sigma$;

(3) If σ is a termination symbol, then each component of $v(\sigma)$ is either 0 or 1.

(4) If σ is an initiation symbol, then each component of $v(\sigma)$ is either 0 or -1.

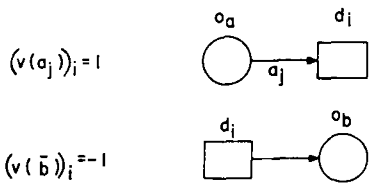
(5) For any two distinct initiation symbols σ and σ', if $v(\sigma)_i = -1$, then $v(\sigma')_i = 0$.

THEOREM 5.1. *Every flowchart is persistent, commutative, and permutable.*

The proof is immediate from the commutativity of vector addition and the sufficient conditions for persistence and permutability of counter schemata given in Section IV.

A flowchart $\mathscr{S}$ can be represented as a directed network $G(\mathscr{S})$ with *operation nodes*, $o_a, o_b, o_c, \ldots$ corresponding to the operations, and *counters* $d_1, d_2, \ldots, d_k$. Each counter d_i is labelled with π_i, its initial value. Branches run either from operation nodes to counters, or from counters to operation nodes. Each branch directed out of an operation node o_a is labelled with an outcome a_j. The branches correspond to the nonzero components of the vectors $v(\sigma)$ as follows:

If $(v(a_j))_i = 1$, the termination of a with outcome a_j increments counter i by 1; if $(v(\bar{a}))_i = -1$, the initiation of b decrements counter i by 1. Condition 5 of Definition 5.1 implies that not more than one branch is directed out of any counter node.



Example 5.1. The interpreted flowchart presented informally in Figure 5.1 represents an algorithm to find the first positive number in the sequence $x_1, x_2, \ldots, x_m$; the result is placed in location z. The subsequences $x_1, x_3, x_5, \ldots$ and $x_2, x_4, x_6, \ldots$ are examined concurrently; the investigation of a subsequence is terminated either when its first positive element is found or when all elements of lower index than the first positive element in the other subsequence have been examined.

For the reader who wishes to construct a formal specification of this parallel flowchart, the following remarks may be helpful:

(1) Operation nodes left blank correspond to operations with null domain and range;

(2) The operation labelled "$x_i > 0$" has the set of domain locations $\{\mathbf{x}, i\}$, where $\mathbf{x}$ contains the entire sequence $x_1, x_2, \ldots, x_m$. This artifice is necessary because our model treats the set of domain locations of an operation as fixed, rather than changeable through index modification.

Analogies can be drawn between certain proposed instructions for representing parallel sequencing in computer languages and certain capabilities of flowcharts. For example, the *fork* instruction causes two or more operations to be initiated upon a given outcome. In flowcharts this is achieved by having more than one $+1$ in a

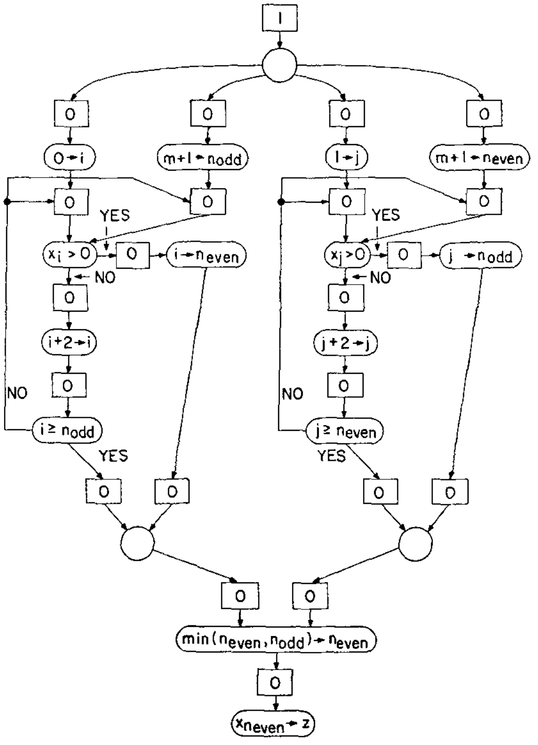
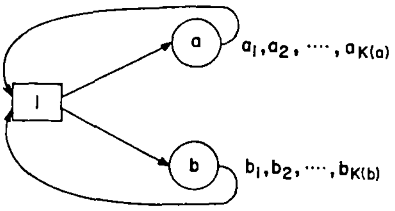


FIG. 5.1

vector $v(a_j)$ (i.e., having an outcome increment more than one counter). The *join* instruction causes the initiation of an operation when a set of preceding operations have all terminated with given outcomes. This corresponds to having a vector $v(a)$ with more than one -1 (an operation node fed by more than one counter). The *quit* instruction causes a sequence of instructions to terminate, without necessarily causing the termination of the entire algorithm. This may be represented by having an outcome which increments no counters.

Certain desirable types of parallel sequencing cannot be represented using parallel flowcharts. For example, suppose we wish to enforce the constraint that operations a and b, which can simultaneously be eligible for initiation, shall never be in progress (i.e., initiated but not terminated) concurrently. Such a constraint cannot be satisfied within any persistent schema; however, if we relax clause 5 in the definition of a parallel flowchart (thereby forfeiting persistence) and allow a counter to feed two operations, then the following arrangement suffices:



Although it would seem desirable to allow a counter to be decremented by more than one operation, we do not so generalize flowcharts for the results in this paper.

As a second example, consider an algorithm such that, each time operations *a* and *b* are performed in parallel, the outcomes select one of the four operations *c*, *d*, *e*, and *f*. The structure shown in Figure 5.2 seems to perform such a selection.

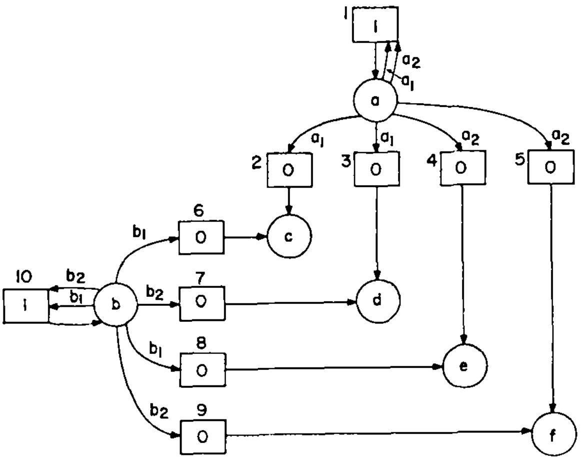


FIG. 5.2

For instance, the pair (a_1, b_1) selects operation *c*. However, if the first two pairs of outcomes are (a_1, b_1) and (a_2, b_2), then operations *d* and *f*, as well as the intended operations *c* and *e*, will be initiated. This occurs because "spurious" 1's are placed in counters 3 and 8 upon the outcomes a_1 and b_1, and remain in these counters until *d* and *e*, respectively, are initiated. Moreover, the difficulty is not particular to the parallel flowchart chosen, but is inherent in the definition of a parallel flowchart; for it is easy to prove that, in any parallel flowchart, $\tau(q, a_1b_2\bar{d})$ defined and $\bar{d} \notin x \Rightarrow \tau(q, a_1xb_2\bar{d})$ defined. This difficulty, like the previous ones, can be circumvented by relaxing clause 5 in Definition 5.1, and allowing a counter to feed two operations. With this added freedom, the structure shown in Figure 5.3 suffices.

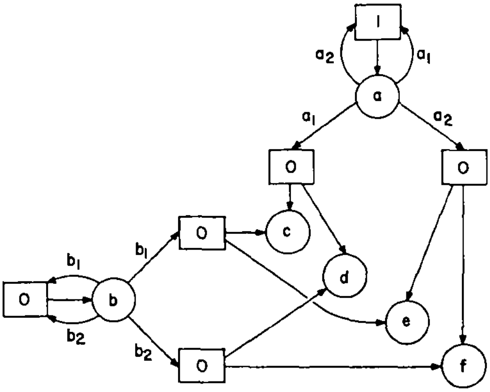


FIG. 5.3

VI. DECISION-FREE SCHEMATA

A schema $\mathscr{S}$ is called *decision-free* if every operation of $\mathscr{S}$ has exactly one outcome; i.e., for each $a \in A$, $K(a) = 1$. Quite apparently, this class of schemata is very restricted. It does suffice, however, for the representation of certain iterative parallel computations similar to those that can be represented by computation graphs [*3*]. The restriction to decision-free flowcharts leads to more detailed results than have been possible in previous sections, and to straightforward tests of properties which, for counter schemata in general, can be checked only by very complex decision procedures based on vector addition systems. As we shall show in this section, the number of performances of an operation in a decision-free flowchart does not depend upon the interpretation or computation chosen; moreover, a simple algorithm is given for calculating this number for each operation. Also, we give a simple test for decision-free flowcharts to be repetition-free and determinate and another simple test for the equivalence of decision-free, repetition-free, determinate flowcharts.

Any decision-free schema $\mathscr{S}$ has the property that every sequence $x \in \Sigma^*$ which satisfies Properties 1 and 2 of Theorem 3.1 is an element of $p(\mathscr{S})$. Since $K(a) = 1$, Property 3 is automatically satisfied for such schemata. The ability to dispense with Property 3 of Theorem 3.1 was basic in our analysis of repetition-free schemata. Since this can also be done for decision-free schemata, many of the techniques that applied to repetition-free schemata also apply here.

A. *Computation of* $\#_{\mathscr{S}}(a)$

The next lemma and theorem establish that in a decision-free flowchart, the number of performances of any operation is independent of the interpretation and computation being considered.

LEMMA 6.1. *Let $\mathcal{S}$ be a decision-free schema, and let $\mathcal{I}$ and $\mathcal{I}'$ be two interpretations. Then x is an $\mathcal{I}$-computation if and only if x is an $\mathcal{I}'$-computation.*

Proof. From Definition 1.5 it is clear that a word y is a prefix of an $\mathcal{I}$-computation if and only if it is a prefix of an $\mathcal{I}'$-computation. Thus, the conditions for a sequence x to be an $\mathcal{I}$-computation, as given in Definition 1.6, are also independent of the interpretation.

THEOREM 6.1. *Let $\mathcal{S}$ be a decision-free persistent commutative schema. Then, for any $a \in A$, any two computations of $\mathcal{S}$ contain the same number of occurrences of $\bar{a}$.*

Proof. Let x and y be two arbitrary computations of $\mathcal{S}$. The main part of the proof consists of showing, by induction, that for any n there is a computation $z(n)$ such that:

(i) for each $a \in A$, $z(n)$ has the same number of occurrences of $\bar{a}$ as y. (Also, it has the same number of occurrences of a_1.)

(ii) ${}_n z(n) = {}_n x$.

The inductive proof is essentially the same "sliding" argument as that used in the proof of Theorem 2.1, so it will not be repeated here.

Now, by (i), $z(n)$ has the same number of occurrences of $\bar{a}$ as y, and, by (ii), for any $n \leqslant l(x)$, we have made ${}_n z(n) = {}_n x$. Thus, if a kth occurrence of $\bar{a}$ appears in x, it also occurs in y; thus, y must contain at least as many occurrences of $\bar{a}$ as x. Since x and y are arbitrary computations, the proof is complete.

For each operation a of $\mathcal{S}$, let $\#_{\mathcal{S}}(a)$ denote the number of initiations of a in any computation. In Theorem 4.7, it has been established that $\#_{\mathcal{S}}(a)$ can, in principle, be computed if $\mathcal{S}$ is a repetition-free counter schema. In this subsection we give a more efficient method for computing $\#_{\mathcal{S}}(a)$ when $\mathcal{S}$ is a decision-free flowchart.

The computation of $\#_{\mathcal{S}}(a)$ proceeds in three steps. First, an iterative technique is used to determine those operations a for which $\#_{\mathcal{S}}(a) = 0$. Next, those operations a for which $0 < \#_{\mathcal{S}}(a) < \infty$ are determined by using properties of certain subgraphs of $G(\mathcal{S})$, the graph associated with the flowchart $\mathcal{S}$. Finally, a system of equations is constructed for computing the numbers $\#_{\mathcal{S}}(a)$ for those operations for which $0 < \#_{\mathcal{S}}(a) < \infty$.

To determine $\{a \mid \#_{\mathcal{S}}(a) = 0\}$, consider the following iteration scheme:

$$\begin{aligned}
T_0 &= \phi \\
S_0 &= \{i \mid (q_0)_i > 0\} \\
T_{j+1} &= T_j \cup \{a \mid (d_i\,, o_a) \in G(\mathcal{S}) \Rightarrow i \in S_j\} \\
S_{j+1} &= S_j \cup \{i \mid \text{for some } a \in T_{j+1}\,, (o_a\,, d_i) \in G(\mathcal{S})\}.
\end{aligned}$$

In this iteration scheme S_0 indicates the counters that are initially positive; T_1 is the set of operations which are fed only by counters indicated by S_0 (and can thus be initiated); S_j indicates additional counters which can later become positive; and T_{j+1} indicates other operations which can be initiated due to the additional counters becoming positive. The iteration terminates when, for some k, $T_k = T_{k+1}$. It is clear that such a k exists, since A is a finite set, and that $T_k = \{a \mid \#_{\mathscr{S}}(a) > 0\}$, since the elements of T_k are precisely those operations fed only by counters which can attain positive values. Thus, $A \cap \bar{T}_k = \{a \mid \#_{\mathscr{S}}(a) = 0\}$.

The following theorem is useful for specifying an algorithm to obtain the set of operations $\{a \mid \#_{\mathscr{S}}(a) < \infty\}$.

THEOREM 6.2. *Let $\mathscr{S}$ be a decision-free flowchart. Then operation a is terminating if and only if o_a is contained in a subgraph G' of $G(\mathscr{S})$ having the following properties:*

(i) *if o_b is a vertex of G', then there is an edge of G' directed into o_b if and only if $\#_{\mathscr{S}}(b) \neq 0$.*

(ii) *if d_i is a vertex of G', then every edge of $G(\mathscr{S})$ directed into d_i is an edge of G'.*

(iii) *G' is acyclic.*

Proof. First we define a subgraph H of $G(\mathscr{S})$ which contains every vertex o_a for which a is terminating, and show that H satisfies (i)–(iii). Let x be a computation for $\mathscr{S}$ and let k be chosen so that ${}_kx$ contains every termination symbol associated with any terminating operation. The subgraph H is defined as follows:

(a) o_a is a vertex of H if and only if a is a terminating operation.

(b) d_i is a vertex of H if and only if:

(1) each operation that feeds counter i is terminating;

(2) there is an operation b fed by d_i, b is terminating and $\#_{\mathscr{S}}(a) > 0$; and

(3) the ith coordinate of $\tau(q_0, {}_kx)$ is zero (i.e., the value of the counter corresponding to vertex d_i is zero after the sequence ${}_kx$, and thus constrains an operation from further initiation).

(c) Every edge of $G(\mathscr{S})$ between vertices of H is an edge of H.

Now, we must show that H satisfies (i)–(iii). By condition (b)(2) for H, if o_a has $\#_{\mathscr{S}}(a) = 0$, then there is no $d_i \in H$ such that d_i feeds o_a in $G(\mathscr{S})$. Thus those operations for which $\#_{\mathscr{S}}(a) = 0$ satisfy condition (i). Now consider any terminating operation b such that $o_b \in H$ and $\#_{\mathscr{S}}(b) > 0$. If in H no counter feeds o_b, then every counter feeding o_b in $G(\mathscr{S})$ either has a positive value after the sequence ${}_kx$ or else is incremented by a nonterminating operation. This would allow operation b to be initiated and terminated again, however, which violates the assumption on how k was chosen in ${}_kx$. Thus H satisfies condition (i). Now by (b)(1), a vertex d_i is in H only if every operation feeding counter i is a terminating operation. Then, since o_b is in H for each terminating operation b, and each edge of $G(\mathscr{S})$ between vertices of H is also an edge of H, it follows that H satisfies (ii). Finally, to show that H satisfies (iii), assume to the contrary, that H contains a cycle C. In a cycle each vertex has an edge entering it and since any $o_a \in H$ for which $\#_{\mathscr{S}}(a) = 0$ has no entering edge, each operation vertex b of C must be such that $\#_{\mathscr{S}}(b) > 0$. Of the operation vertices in C consider that operation b which terminates last in ${}_k x$. In C we have an edge (o_b, d_i), and since $\mathscr{S}$ is decision-free, we have that the ith coordinate of $\tau(q_{0,k}x)$ is greater than zero. By condition (b)(3) for H, however, this is impossible, so H is acyclic.

To complete the proof, let G' be a subgraph of $G(\mathscr{S})$ satisfying (i), (ii), and (iii). We must show that $\#_{\mathscr{S}}(a) < \infty$ for each $o_a \in G'$. Since G' is acyclic there must be some vertex v of G' which has no entering edge. If v is an operation vertex o_a, then by (i) $\#_{\mathscr{S}}(a) = 0$. If v is a counter vertex d_i, then by (ii) no edge of $G(\mathscr{S})$ enters d_i so counter d_i can only be decremented and if operation a is fed by d_i, then $\#_{\mathscr{S}}(a) < \infty$. Now if $k(c)$ is the length of the longest path in G' to a vertex o_c of G', it follows by induction on $k(c)$ that $\#_{\mathscr{S}}(c) < \infty$ for each $o_c \in G'$. This follows from the obvious proposition that if $(d_i, o_c) \in G'$, and, for all b such that $(o_b, d_i) \in G'$, $\#_{\mathscr{S}}(b) < \infty$, then $\#_{\mathscr{S}}(c) < \infty$.

The construction of H in the proof of Theorem 6.2 describes how the terminating vertices are exactly those vertices which trace back to counters having no inputs and to operation vertices o_a for which $\#_{\mathscr{S}}(a) = 0$. The set of terminating vertices may be generated conveniently by the following iteration:

$$U_0 := \{a \mid \#_{\mathscr{S}}(a) = 0\}$$

$$U_{j+1} := U_j \cup \{a \mid \exists\, d_i \text{ such that } (d_i, o_a) \in G(\mathscr{S}) \text{ and } (o_b, d_i) \in G(\mathscr{S}) \Rightarrow b \in U_j\}.$$

The iteration ends when, for some k, $U_{k+1} = U_k$; then $U = U_k = \{a \mid \#_{\mathscr{S}}(a) < \infty\}$.

Once the terminating vertices are known, then $\{\#_{\mathscr{S}}(a) \mid 0 < \#_{\mathscr{S}}(a) < \infty\}$ can be determined in several ways. A prefix y of a computation can be generated iteratively up to a point when the conditions required, in the proof of Theorem 6.2, for a subgraph H are satisfied and then $\#_{\mathscr{S}}(a)$ can be counted in y for each terminating operation. Note that H cannot, in general, be constructed directly from U since the counters which "constrain" the operations from further initiations are not known in advance. Generally, a more convenient computation of the quantities $\#_{\mathscr{S}}(a)$ is through the solution of the following system of equations.

Let $S = \{i \mid (o_b, d_i) \in G(\mathscr{S}) \Rightarrow b \in U\}$, then for $a \in U$ and $\#_{\mathscr{S}}(a) \neq 0$ we have:

$$\#_{\mathscr{S}}(a) = \operatorname*{Min}_{\{i \mid (d_i, o_a) \in G(\mathscr{S}) \text{ and } i \in S\}} \Big[(q_0)_i + \sum_{(o_b, d_i) \in G(\mathscr{S})} \#_{\mathscr{S}}(b) \Big].$$

This system is readily solved, using the proof of the following theorem.

THEOREM 6.3. *Consider the system of equations*

$$x_i = \operatorname*{Min}_{1\leqslant k\leqslant N_i} (a_i{}^k + \Sigma_j b_{ij}^k x_j) \qquad i = 1, 2, \dots, n, \tag{2}$$

for which:

(i) *there exists a solution*;

(ii) $a_i{}^k \geqslant 0$ *for all i and k*;

(iii) $b_{ij}^k = 0$ *or* 1;

(iv) *there exists no sequence* $(i_0, k_0), (i_1, k_1), \dots, (i_r, k_r)$ *such that*

$$b_{i_l j}^{k_l} = \begin{cases} 1 & \text{if} \quad j = i_{l+1} \pmod r \\ 0 & \text{otherwise} \end{cases}$$

and $a_{i_l}^{k_l} = 0$.

For such a system, we conclude:

(a) *The solution is unique*;

(b) *There is at least one pair (i, k) such that*

$$b_{ij}^k = 0 \qquad \textit{for all} \qquad j. \tag{3}$$

(c) *If (i^*, k^*) satisfies (3) and $a_{i^*}^{k^*} \leqslant a_i{}^k$ whenever (i, k) satisfies (3), then $x_{i^*} = a_{i^*}^{k^*}$.*

Proof. Let y_i, $i = 1, 2, \dots, n$ be a solution to (2). Then for each i there is a k_i such that

$$y_i = a_i^{k_i} + \Sigma_j b_{ij}^{k_i} y_j .$$

Let $S = \{i \mid y_i = \min_{j=1,2,\dots,n} (y_j)\}$. Then, for $i \in S$, we have either $b_{ij}^{k_i} = 0$ for all j so that the equation for y_i is of the form

$$y_i = a_i^{k_i}$$

or $a_i^{k_i} = 0$ and $b^{k_i} = 1$ for exactly one j, where $j \in S$. This follows from the minimality of the y_i, together with (ii) and (iii). But, if the latter alternative held for each $i \in S$, condition (iv) would be violated. This proves (b). Now, we have $y_{i'} = a_{i'}^{k_{i'}}$ for some $i' \in S$. Let i^* be as in (c). Then $y_{i*} \geqslant y_{i'} = a_{i'}^{k_{i'}} \geqslant a_{i*}^{k}$. From the equations, however, $y_{i*} \leqslant a_{i*}^{k*}$, hence, $y_{i*} = a_{i*}^{k*}$ proving (c). Now by replacing $x_{i'}$ by $a_{i'}^{k_{i'}}$ in (2), we obtain a reduced system of equations satisfying (i)–(iv). Thus, another component of the solution may be obtained by inspection of the reduced system using Property (c). This process may be repeated until a complete solution is obtained, and this solution is clearly unique, proving (a), and completing the proof.

The following example illustrates the technique for computing $\#_{\mathscr{S}}(a)$ for each operation a of $\mathscr{S}$.

Example 6.1. Consider the flowchart having a $G(\mathscr{S})$ as shown in Figure 6.1.

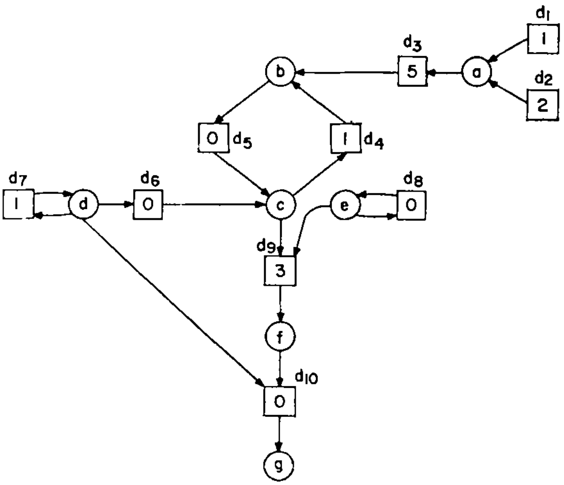


FIG. 6.1

The iteration to compute $\{a \mid \#_{\mathscr{S}}(a) = 0\}$ proceeds as follows:

$$T_0 = \phi;$$

$$S_0 = \{i \mid (q_0)_i > 0\} = \{1, 2, 3, 4, 7, 9\}$$

$$T_1 = T_0 \cup \{a \mid (d_i, o_a) \in G(\mathscr{S}) \Rightarrow i \in S_j\} = \{a, b, d, f\}$$

$$S_1 = S_0 \cup \{i \mid \text{for some } a \in T_1, (o_a, d_i) \in G(\mathscr{S})\} = \{1, 2, 3, 4, 5, 6, 7, 9, 10\}$$

$$T_2 = \{a, b, c, d, f, g\}$$

$$S_2 = S_1, \qquad T_3 = T_2$$

$$\therefore \quad \{a \mid \#_{\mathscr{S}}(a) = 0\} = \{e\}.$$

Next, the set of terminating vertices is generated

$$U_0 = \{e\}$$

$$U_1 = U_0 \cup \{a \mid (d_i, o_a) \in G(\mathscr{S}) \text{ and } (o_b, d_i) \in G(\mathscr{S}) \Rightarrow b \in U_0\} = \{e, a\}$$

$$U_2 = \{a, e, b\}$$

$$U_3 = \{e, a, b, c\}$$

$$U_4 = \{a, b, c, e, f\} = U_5 .$$

Therefore, the set of terminating vertices is $U = \{a, b, c, e, f\}$. The set $S = \{1, 2, 3, 4, 5, 8, 9\}$. The system of equations to compute $\#_{\mathscr{S}}(a)$ for each of these vertices is:

$$\begin{aligned} \#_{\mathscr{S}}(a) &= \mathrm{Min}[1, 2] \\ \#_{\mathscr{S}}(b) &= \mathrm{Min}[5 + \#_{\mathscr{S}}(a), 1 + \#_{\mathscr{S}}(c)] \\ \#_{\mathscr{S}}(c) &= \mathrm{Min}[\#_{\mathscr{S}}(b), \#_{\mathscr{S}}(a)] \\ \#_{\mathscr{S}}(f) &= \mathrm{Min}[3 + \#_{\mathscr{S}}(c)]. \end{aligned}$$

The solution to this system of equations is:

$$\#_{\mathscr{S}}(a) = 1,\ \#_{\mathscr{S}}(b) = 6,\ \#_{\mathscr{S}}(\mathrm{c}) = 6,\ \#_{\mathscr{S}}(f) = 9, \text{ and of course, } \#_{\mathscr{S}}(e) = 0,$$

completing the example.

B. *Decidable properties of decision-free flowcharts*

It has been noted earlier that, in the case of decision-free schemata, Property 3 of Theorem 3.1 holds trivially; in general, however, repetition-freeness is required in order to dispense with Property 3 and apply the techniques associated with t-counter transition systems. Moreover, the hypothesis of losslessness can be omitted from certain theorems about decision-free flowcharts since conditional transfers which leave no trace in the memory cannot occur. It follows easily that some of the results of Section IV, when they are applied to decision-free schemata, can be sharpened. The modified results are given in Theorem 6.4.

THEOREM 6.4. *The following questions concerning a decision-free counter schema $\mathscr{S}$ are decidable*:

(i) *Is $\{\Delta_a(x) \mid x \in p(\mathscr{S})\}$ finite? If so, what is the maximum element of the set?*

(ii) *Is $\mathscr{S}$ serial?*

(iii) *Is there an upper bound on the values stored in a given counter?*

(iv) *In the case where $\mathscr{S}$ is repetition-free, persistent, and commutative, is $\mathscr{S}$ determinate?*

This subsection will be devoted to a further analysis of decision problems concerning decision-free flowcharts. The main results of this analysis are a simple test to determine whether a decision-free flowchart is repetition-free and determinate, and a simple test of the equivalence of decision-free repetition-free determinate flowcharts. These tests are based on the construction of the "characterizing sequence" of a decision-free flowchart.

LEMMA 6.2. *Let $\mathscr{S}$ be a decision-free repetition-free flowchart. Then $\mathscr{S}$ is determinate* *if and only if, for every pair of operations* (a, b) *such that* apb, $R(a) \neq \varphi$ *and* $R(b) \neq \varphi$, *and any two computations* x *and* y,

$$\mathscr{E}_{\bar{a}\bar{b}}(x) = \mathscr{E}_{\bar{a}\bar{b}}(y).$$

This lemma is a slight variant of Corollary 2.2 and the proof is similar.

Lemma 6.2 points to the invariance (independence of computation) of the subsequence $\mathscr{E}_{\bar{a}\bar{b}}(x)$, whenever apb, as an important structural property of decision-free flowcharts. Lemmas 6.3 and 6.4, together with Theorem 6.5 and its corollaries, give a more detailed analysis of this invariance. The main results are that if $\mathscr{E}_{\bar{a}\bar{b}}(x)$ is invariant, then it must have a special "alternating" form (Theorem 6.5), and that $\mathscr{E}_{\bar{a}\bar{b}}(x)$ is an invariant if and only if its first four elements are well determined independent of the computation (Corollary 6.2).

LEMMA 6.3. *Let* $\mathscr{S}$ *be a decision-free flowchart including operations* a *and* b. *Let* z *be an element of* $p(\mathscr{S})$ *such that* $\bar{b} \in z$, $zv\bar{a} \in p(\mathscr{S})$ *for some* v, *and the following implication holds*: $zu\bar{a} \in p(\mathscr{S}) \Rightarrow \bar{b} \in u$ (i.e., *a cannot be initiated before b*). *Then* $G(\mathscr{S})$ *has a subgraph* G' *with the following properties*:

(i) G' *is acyclic, and each of its vertices lies in a path directed into* o_a;

(ii) *if* $o_c \in G'$ *where* $c = b$ *or* $\#_{\mathscr{S}}(c) = 0$, *then no edge of* G' *is directed into* o_c; *otherwise, exactly one edge is directed into* o_c;

(iii) *if* $d_i \in G'$, *then every edge of* $G(\mathscr{S})$ *directed into* d_i *is an edge of* G';

(iv) *if* $d_i \in G'$, *then* $[\tau(q_0, z)]_i = 0$, *and if* $o_c \in G'$, *then* $\Delta_c(z) = 0$.

The intuitive content of this lemma is that, under the given hypotheses, every path along which an initiation signal might propagate to o_a includes o_b. The proof, which we omit, is similar to the proof of Theorem 6.2, in which a related graphical construction is given. We note, however, that the hypothesis that $\bar{b} \in z$ cannot be dispensed with; for, in the flowchart of Figure 6.2 with z taken as the null sequence, the implication $zu\bar{a} \in p(\mathscr{S}) \Rightarrow \bar{b} \in u$ holds, but no graph G' with the required properties exists (in particular, G' cannot be taken to be acyclic).

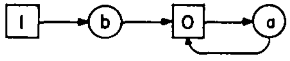


FIG. 6.2

For any $w \in p(\mathscr{S})$, let $\#_w(a)$ denote the number of occurrences of $\bar{a}$ in w. Then the following consequence of Lemma 6.3 may be observed.

LEMMA 6.4. *Under the hypotheses of Lemma* 6.3 *there exist nonnegative integers* α *and* β *such that*

$$\#_z(a) = \alpha \#_z(b) + \beta$$

and, for every $w \in p(\mathscr{S})$,

$$\#_w(a) \leqslant \alpha \#_w(b) + \beta.$$

Proof. For each edge $(d_i, o_c) \in G'$, and for each $w \in p(\mathscr{S})$,

$$\#_w(c) \leqslant (q_0)_i + \sum_{\{d \mid (o_c, d_i) \in G(\mathscr{S})\}} \#_w(d),$$

with equality holding for $z = w$. Since G' is acyclic the inequalities corresponding to interior vertices of G' can be eliminated; setting $\#_w(c) = 0$ for every c such that $\#_{\mathscr{S}}(c) = 0$, the desired linear inequality (linear equation in the case $w = z$) is at hand.

Note that for the flowchart of Figure 6.2 no relation of the form $\#_w(a) \leqslant \alpha \#_w(b) + \beta$ holds, even though a cannot be initiated before b is. This is no contradiction, however, since there is no z in $p(\mathscr{S})$ for which all the hypotheses of Lemma 6.4 hold.

THEOREM 6.5. *Let* $\mathscr{S}$ *be a decision-free flowchart. Let* a *and* b *be operations such that* $\#_{\mathscr{S}}(a) > 0$, $\#_{\mathscr{S}}(b) > 0$ *and, for any two computations* x *and* y, $\mathscr{E}_{\bar{a}\bar{b}}(x) = \mathscr{E}_{\bar{a}\bar{b}}(y)$. *Then* $\mathscr{E}_{\bar{a}\bar{b}}(x)$ *has the following properties*:

(i) *if* $\#_{\mathscr{S}}(a) > 1$ *and* $\#_{\mathscr{S}}(b) > 1$, *then* $|\#_{\mathscr{S}}(a) - \#_{\mathscr{S}}(b)| \leqslant 1$, *and occurrences of* $\bar{a}$ *and* $\bar{b}$ *alternate*;

(ii) *if* $\#_{\mathscr{S}}(a) = 1$ *then* $\bar{a}$ *occurs as either the first or second element in* $\mathscr{E}_{\bar{a}\bar{b}}$; *similarly for* $\bar{b}$.

Proof. Let $r = \min(2, \#_{\mathscr{S}}(a)) + \min(2, \#_{\mathscr{S}}(b))$, and let $\mathscr{E}_{\bar{a}\bar{b}}(x)$ begin with the sequence $s_1 s_2 \cdots s_r$.

With no loss of generality, consider the case where $s_1 = \bar{a}$. Then, by repeated application of Lemma 6.4, we find that all of the possibilities for the sequence $s_1 s_2 \cdots s_r$ are accounted for as follows:

$$\begin{aligned}
s_1 s_2 &= \bar{a}\bar{a} \Rightarrow \#_{\mathscr{S}}(b) = 0\\
s_1 s_2 s_3 &= \bar{a}\bar{b}\bar{b} \Rightarrow \#_{\mathscr{S}}(a) = 1\\
s_1 s_2 s_3 s_4 &= \bar{a}\bar{b}\bar{a}\bar{a} \Rightarrow \#_{\mathscr{S}}(b) = 1\\
s_1 s_2 s_3 s_4 &= \bar{a}\bar{b}\bar{a}\bar{b} \Rightarrow \text{for all } w\ \#_w(b) \leqslant \#_w(a)\\
&\qquad\qquad \text{and either } \#_w(a) \leqslant \#_w(b) + 1\\
&\qquad\qquad \text{or } \#_{\mathscr{S}}(a) = 2.
\end{aligned}$$

In each case, Properties (i) and (ii) follow at once from these implications. To illustrate how the implications are derived, consider the case $s_1 s_2 s_3 s_4 = \bar{a}\bar{b}\bar{a}\bar{b}$. Then there is a z

satisfying the hypotheses of Lemma 6.4 such that $\#_z(a) = 2$ and $\#_z(b) = 1$. Thus, there are nonnegative integers α and β such that $2 = \alpha \cdot 1 + \beta$ and for all $w \in p(\mathscr{S})$; $\#_w(a) \leqslant \alpha \#_w(b) + \beta$. The possible choices are $\alpha = 0$, $\beta = 2$; $\alpha = 1$, $\beta = 1$; and $\alpha = 2$, $\beta = 0$; and the last case is not possible since $s_1 = \bar{a}$.

It will be convenient to notice that the proof of Theorem 6.5 actually can be obtained from weaker hypotheses than those of the theorem as we show in Corollary 6.1.

DEFINITION 6.1. For all $w \in p(\mathscr{S})$ and all $a \in A$, let $S(w, a) = \{b \mid b \in \Sigma$ and, for some u, $wu\bar{b} \in p(\mathscr{S})$ and $\bar{a} \notin u\}$. That is, $S(w, a)$ is the set of operations which after w, can be initiated before the next initiation of a.

COROLLARY 6.1. *Let v be an element of $p(\mathscr{S})$ containing* $\min(2, \#_{\mathscr{S}}(a))$ *occurrences of $\bar{a}$, and* $\min(2, \#_{\mathscr{S}}(b))$ *occurrences of $\bar{b}$. Suppose that z satisfies the following: If $z = z_1 \sigma z_2$, where $\sigma \in \{\bar{a}, \bar{b}\}$, then either $a \notin S(z_1\sigma, b)$ or $b \notin S(z_1\sigma, a)$. Then, in any two computations x and y, $\mathscr{E}_{\bar{a}\bar{b}}(x) = \mathscr{E}_{\bar{a}\bar{b}}(y)$.*

The proof of this corollary is almost identical to the proof of Theorem 6.5.

A more simply stated result implied by this corollary is the following:

COROLLARY 6.2. *The following statements are equivalent:*

(i) *In any two computations x and y, $\mathscr{E}_{\bar{a}\bar{b}}(x) = \mathscr{E}_{\bar{a}\bar{b}}(y)$;*

(ii) *in any two computations x and y, ${}_r(\mathscr{E}_{\bar{a}\bar{b}}(x)) = {}_r(\mathscr{E}_{\bar{a}\bar{b}}(y))$* where

$$r = \min(2, \#_{\mathscr{S}}(a)) + \min(2, \#_{\mathscr{S}}(b)).$$

Theorem 6.5 imposes a strong restriction on the form which $\mathscr{E}_{\bar{a}\bar{b}}(x)$ must take if it is to be independent of the computation x considered. Using this result, we can derive simple conditions which are necessary and sufficient for a decision-free flowchart $\mathscr{S}$ to be repetition-free and determinate.

THEOREM 6.6. *Let $\mathscr{S}$ be a decision-free flowchart. Then $\mathscr{S}$ is repetition-free and determinate if and only if the following conditions hold:*

(i) *$a\rho b$, $R(a) \neq \varphi$ and $R(b) \neq \varphi \Rightarrow$ for any two computations x and y, $\mathscr{E}_{\bar{a}\bar{b}}(x) = \mathscr{E}_{\bar{a}\bar{b}}(y)$;*

(ii) *$D(a) \cap R(a) \neq \varphi \Rightarrow$ in any computation, occurrences of $\bar{a}$ and a_1 alternate;*

(iii) *$\#_{\mathscr{S}}(a) \geqslant 2$ and $D(a) \cap R(a) = \varphi \Rightarrow$ for some b such that $R(b) \cap D(a) \neq \varphi$, occurrences of $\bar{a}$ and $\bar{b}$ alternate in any computation.*

Proof. Assume (i), (ii), and (iii). Suppose $\mathscr{S}$ were not repetition-free. Then, for some a there would exist a sequence $v\bar{a}w\bar{a} \in p(\mathscr{S})$ such that $c_1 \in w \Rightarrow R(c) \cap D(a) = \varphi$. If $R(a) \cap D(a) \neq \varphi$, then, setting $c = a$, (ii) is contradicted; otherwise, by (iii), w can be written $w_1\bar{b}w_2$, where $R(b) \cap D(a) \neq \varphi$, but $b_1 \notin w_2$. Hence $p(\mathcal{S})$ contains the sequence $v\bar{a}w_1\bar{b}w_2\bar{a}b_1$. But, by persistence and permutability, $v\bar{a}w_1w_2\bar{a}\bar{b} \in p(\mathcal{S})$, contradicting (iii). Hence $\mathcal{S}$ is repetition-free. Now, by Lemma 6.2, $\mathcal{S}$ is determinate.

Now assume that $\mathcal{S}$ is repetition-free and determinate. By Lemma 6.2, (i) holds; clearly, if (ii) did not hold, then determinacy would be violated. If (iii) did not hold, then, from Theorem 6.5, the following conclusions would be reached:

> for every b such that $R(b) \cap D(a) \neq \varphi$, $\#_{\mathcal{S}}(b) = 1$; either $\#_{\mathcal{S}}(a) = 2$ and $\mathcal{E}_{\bar{a}\bar{b}} = \bar{b}\bar{a}\bar{a}$ for every such b or $\#_{\mathcal{S}}(a) > 2$ and for every such b, $\mathcal{E}_{\bar{a}\bar{b}}$ has one of the following two initial sequences:
>
> $$\bar{b}\bar{a}\bar{a}\bar{a},\ \bar{a}\bar{b}\bar{a}\bar{a}.$$

In the former case, the first and second occurrences of $\bar{a}$ operate on the same data; in the latter case, the second and third occurrences of $\bar{a}$ operate on the same data. Either way, a repetition must occur.

Using Theorem 6.6 and Corollary 6.2, we shall derive a very simple test to determine whether a decision-free flowchart is both repetition-free and determinate. The test will depend on the concept of the characterizing sequence $v_{\mathcal{S}}$ of a decision-free flowchart $\mathcal{S}$.

LEMMA 6.5. *Given a decision-free flowchart $\mathcal{S}$ one can effectively construct a sequence $v_{\mathcal{S}} \in p(\mathcal{S})$ containing, for each a, exactly* $\min(2, \#_{\mathcal{S}}(a))$ *occurrences of $\bar{a}$, and* $\min(2, \#_{\mathcal{S}}(a))$ *occurrences of a_1.*

Proof. Let $\mathcal{S}'$ equal $\mathcal{S}$ except that $\mathcal{S}'$ has an additional counter feeding each operation. Each new counter has initial value 2 and is never incremented. Then, for any operation a, $\#_{\mathcal{S}'}(a) = \min(2, \#_{\mathcal{S}}(a))$. This can be shown formally, but it is intuitively evident from the property that in a decision-free flowchart, the third termination of one operation is not required for the second initiation of another operation (cf., Theorem 6.5). Any computation of $\mathcal{S}'$ is a prefix of a computation of $\mathcal{S}$ and hence, can serve as $v_{\mathcal{S}}$.

The sequence $v_{\mathcal{S}}$, whose length is not more than four times the number of operations in $\mathcal{S}$, is called the *characterizing sequence* of $\mathcal{S}$. The sequence $v_{\mathcal{S}}$ plays a significant role in developing simple tests for computational properties of $\mathcal{S}$. First, as the next theorem indicates, $v_{\mathcal{S}}$ can be used in determining whether $\mathcal{S}$ is repetition-free and determinate. Second, the characterizing sequence and the quantities $\#_{\mathcal{S}}(a)$ form the basis for a test of equivalence between repetition-free determinate decision-free flowcharts.

THEOREM 6.7. *Let $\mathcal{S}$ be a decision-free flowchart and let $v_{\mathcal{S}}$ have the properties* *stated in Lemma* 6.5. *Then* $\mathscr{S}$ *is both repetition-free and determinate if and only if the following implications hold*:

(a) *if* $w\bar{a}$ *is a prefix of* $v_{\mathscr{S}}$ *and* apb, *then* $\bar{b} \notin S(w, \bar{a})$;

(b) *if* $D(a) \cap R(a) \neq \varphi$ *and* $w = u\bar{a}$ *is a prefix of* $v_{\mathscr{S}}$, *then* $\bar{a} \notin S(w, a_1)$;

(c) *if* $\#_{\mathscr{S}}(a) \geqslant 2$ *and* $D(a) \cap R(a) = \varphi$, *then, for some* b *such that* $R(b) \cap D(a) \neq \varphi$, *occurrences of* $\bar{a}$ *and* $\bar{b}$ *alternate in* $v_{\mathscr{S}}$.

Proof. If $\mathscr{S}$ is repetition-free and determinate, it follows directly from Theorem 6.6 that (a), (b), and (c) hold, since $v_{\mathscr{S}} \in p(\mathscr{S})$. Conversely, assume that (a), (b), and (c) hold. We shall verify conditions (i), (ii), and (iii) of Theorem 6.6. By Corollary 6.1, (a) $\Rightarrow$ (i). Let b be the operation singled out in (c); then $\mathscr{E}_{\bar{a}\bar{b}}$ is constant over all computations and, by Theorem 6.5, the alternation of $\bar{a}$ and $\bar{b}$ in $v_{\mathscr{S}}$ implies their alternation in $\mathscr{E}_{\bar{a}\bar{b}}(x)$, for any computation x. A similar argument, not given, shows that (b) $\Rightarrow$ (ii).

The test for $\mathscr{S}$ to be repetition-free and determinate is simply to construct the sequence $v_{\mathscr{S}}$ and to test the conditions of the theorem. This test uses the algorithm given in the following lemma.

LEMMA 6.6. *The following iteration determines* $S(w, a)$:

$$\begin{aligned} T_0 &= \varphi \\ S_0 &= \{i \mid (\tau(q_0, w))_i > 0\} \\ T_{j+1} &= T_j \cup \{b \mid b \neq a \quad \textit{and} \quad (d_i, o_b) \in G(\mathscr{S}) \Rightarrow i \in S\} \\ S_{j+1} &= S_j \cup \{i \mid \textit{for some } b \in T_{j+1}, (o_b, d_i) \in G(\mathscr{S})\}. \end{aligned}$$

The iteration terminates when, for some k, $T_{k+1} = T_k$; *and* $S(w, a) = T_k$.

The proof of this lemma is omitted; it simply follows from the fact that an operation can be initiated if and only if all of the counters feeding the operation have positive values.

The next theorem uses $v_{\mathscr{S}}$ for a simple test of the equivalence of repetition-free determinate decision-free flowcharts.

THEOREM 6.8. *Let* $\mathscr{S}$ *and* $\mathscr{S}'$ *be repetition-free determinate decision-free flowcharts with the same set of operations. Then* $\mathscr{S}$ *and* $\mathscr{S}'$ *are equivalent if and only if*:

(i) *for all* a, $\#_{\mathscr{S}}(a) = \#_{\mathscr{S}'}(a)$;

(ii) *if* apb *then* $\mathscr{E}_{\bar{a}\bar{b}}(v_{\mathscr{S}}) = \mathscr{E}_{\bar{a}\bar{b}}(v_{\mathscr{S}'})$.

Proof. By Theorem 6.5 and Corollary 6.1, conditions (i) and (ii) imply that, if x is a computation for $\mathscr{S}$, and x', a computation for $\mathscr{S}'$, then $\mathscr{E}_{\bar{a}\bar{b}}(x) = \mathscr{E}_{\bar{a}\bar{b}}(x')$. Because of persistence (and determinacy), we may take x and x' to be "serial" computations, in which each termination symbol immediately follows its mate. For $i \in M$, let $S_i = \{\bar{a} \mid i \in R(a)\}$. Choose any interpretation $\mathscr{I}$. By a simple induction we can show that, for any i and any k, if $x = x_1 \bar{c} c_1 x_2$ and $x' = x'_1 \bar{d} d_1 x'_2$, where $\bar{c}$ and d are the kth occurrences of an element of S_i in x and x', respectively, then $c = d$, $\Pi_{D(c)}(q_0 \cdot x_1) = \Pi_{D(c)}(q'_0 \cdot x'_1)$, and thus $\Pi_{R(c)}(q_0 \cdot x_1 \bar{c} c_1) = \Pi_{R(c)}(q'_0 \cdot x'_1 \bar{c} c_1)$, where q_0 and q'_0 are the initial states of $\mathscr{S}$ and $\mathscr{S}'$, respectively. Hence, x and x', and therefore (by determinacy), any two $\mathscr{I}$-computations for $\mathscr{S}$ and $\mathscr{S}'$, produce identical cell sequences.

VII. Possible Extensions

The model introduced in this paper omits many considerations about the nature of algorithms and the dynamics of their execution. It would be desirable to extend the model to permit investigation of some of these considerations. One possible direction is to restrict the class of interpretations considered, or, putting it another way, to permit the specification of the schema to include some information about the nature of the operations. For example, a formal means could be provided to specify that two operations compute the same function or that an operation computes the identity function (i.e., it merely transfers data). Also, certain operations might be specified as performing indexing; that is, modifying the domain or range locations of other operations in some regular fashion. It would be interesting to consider the extension of our results to such partially interpreted models.

Another direction is to restrict the class of computations in some meaningful way. For example, some information about the time required to perform operations or about the details of queue discipline might be given. Thus, the effects of priority queueing, interrupts, and the like could be modelled. It remains to be seen whether such elaborations of the present model would lead to interesting results.

The continued search for interesting special classes of schemata and their properties also seems to hold promise. For example, it might be desirable to study classes of counter schemata more general than parallel flowcharts in order to permit types of sequencing not now representable; in particular, the results on decision-free flowcharts can probably be generalized.

There is also room for considerable further investigation of the present model in its most general form. Section IV mentions some unsolved decision problems, and better algorithms for the known solvable problems would be valuable. Other formulations of determinacy and equivalence suggest themselves; for example, a schema might be considered determinate if the final contents of a selected set of memory locations were well defined. Finally, techniques for transforming schemata to equivalent schemata with desirable properties should be sought. Of particular interest along these lines is the study of maximally parallel forms of schemata.

ACKNOWLEDGMENT

The authors wish to thank M. O. Rabin, A. L. Rosenberg, J. D. Rutledge, and D. L. Slutz for their interest and helpful suggestions.

REFERENCES

1. S. GINSBURG. "The Mathematical Theory of Context-Free Languages." McGraw-Hill, New York, 1966.
2. I. I. IANOV. The logical schemes of algorithms. Problems of Cybernetics, **I**, 75–127 (1958).
3. R. M. KARP AND R. E. MILLER. Properties of a model for parallel computations: determinacy termination, queueing. *SIAM J. Appl. Math.* **14**, 1390–1411 (1966).
4. R. M. KARP AND R. E. MILLER. Parallel program schemata: A mathematical model for parallel computation. IEEE Conference Record of the Eighth Annual Symposium on Switching and Automata Theory, October 1967, pp. 55–61.
5. R. M. KARP, R. E. MILLER, AND S. WINOGRAD. The organization of computations for uniform recurrence equations. *J. Assoc. Comp. Mach.* **14**, 563–590 (1967).
6. D. KÖNIG. "Theorie der Endlichen und Unendlichen Graphen." Akademische Verlagsgesellschaft, Leipzig, 1936.
7. D. C. LUCKHAM, D. M. R. PARK, AND M. S. PATERSON. "On formalised computer programs" (Preliminary Draft), Programming Research Group, Oxford University (1967).
8. E. POST. A variant of a recursively unsolvable problem. *Bull. Am. Math. Soc.* **52**, 264–268 (1946).
9. M. O. RABIN AND D. SCOTT. Finite automata and their decision problems. *IBM J. Res. Develop.* **3**, 198–220 (1959).
10. J. D. RUTLEDGE. On Ianov's program schemata. *J. Assoc. Comp. Mach.* **11**, 1–9 (1964).