10-1-1972

On the Decomposition of Asynchronous Systems

Robert M. Keller
Harvey Mudd College

ON THE DECOMPOSITION OF ASYNCHRONOUS SYSTEMS

Robert M. Keller
Department of Electrical Engineering
Princeton University
Princeton, N.J. 08540

Summary This paper reports results of part of a continuing investigation of parallel computation, in particular, efforts toward understanding the nature of different types of parallel control. The first section defines an "asynchronous system" to be a simple type of state machine. This was arrived at in an attempt to generalize from the types of control in parallel program schemata[1,2,3,4] and networks of asynchronous modules without bounded delays.[5,6,7,8] Asynchronous systems with output are also defined in a familiar way. The deviation from standard work comes in the definition of a parallel decomposition of asynchronous systems. Some preliminary work[4] on compositions of this type appears in[4]. Such definitions provide a useful analytic tool for discussion of related theories.

The first goal is to find necessary and sufficient conditions for a system to be parallel decomposable. Such conditions are found, and bear a relation to the "persistent," "permutable," and "commutative" conditions of[1]. These conditions provide a test for decomposability which is, in a sense, local to the states of the system.

The decomposition is applied to parallel program schemata in Section 2. The relation between decomposability and the previously defined concepts of determinacy, closure, etc. is pointed out. For example, for a sufficiently restricted class of schemata, the closure of the parallel composition of two schemata is precisely the parallel composition of the closures of the individual components.

Section 3 describes some other types of decompositions. The first are the series and quasi-series which are almost trivially testable. Finally the fork-join decomposition is introduced which properly generalizes both series and parallel. A criterion for fork-join decomposability is produced for a restricted class of schemata. It is also pointed out that there are schemata which are not fork-join decomposable. This concept is closely related to the "partial ordering" of tasks so prevalent in the literature and sheds some light on the generality of this representation.

1. Parallel Decompositions

Definition 1.1 An asynchronous system (or simply system) is a triple $M = (Q,\Sigma,f)$ where

(1) Q is a set of states

(2) Σ is a finite set called the alphabet

(3) $f: Q\times\Sigma \to Q$ is a partial function, the state-transition function

For $(q,\sigma)\in Q\times\Sigma$, if $f(q,\sigma)$ is defined, we write $\langle f(q,\sigma)\rangle$. In the customary manner, f is extended to $f: Q\times\Sigma^* \to Q$, where Σ^* is the set of all finite sequences of elements in Σ, by the following inductive definition, where o denotes the empty sequence:

(1) $f(q,o) = q$

(2) If $x\in\Sigma^*$, $\sigma\in\Sigma$, and $\langle f(q,x)\rangle$, then

$$f(q,x\sigma) = \begin{cases} f(f(q,x),\sigma) & \text{if defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Definition 1.2 The parallel composition of systems $M_1 = (Q_1,\Sigma_1,f_1)$ and $M_2 = (Q_2,\Sigma_2,f_2)$, written $M_1\times M_2$ where $\Sigma_1\cap\Sigma_2 = \phi$, is the system $M = (Q_1\times Q_2, \Sigma_1\cup\Sigma_2, f)$ where

$$\forall (q,q')\in Q_1\times Q_2 \qquad \forall\sigma\in\Sigma_1\cup\Sigma_2$$

$$f((q,q'),\sigma) = \begin{cases} (f_1(q,\sigma),q') & \text{if } \sigma\in\Sigma_1 \\ (q,f_2(q',\sigma)) & \text{if } \sigma\in\Sigma_2 \end{cases}$$

Note that this parallel composition is actually a proper sub-case of the standard one standard one for "synchronous" sequential machines (cf.[9]) since an input affects only one of the components. This is depicted in Figure 1.1.

Definition 1.3 A partition of Σ is a pair of sets $\{\Sigma_1,\Sigma_2\}$ such that $\Sigma_1\cup\Sigma_2 = \Sigma$ and $\Sigma_1\cap\Sigma_2 = \phi$.

Definition 1.4 A system $M = (Q,\Sigma,f)$ is said to have a parallel decomposition with respect to the partition $\{\Sigma_1,\Sigma_2\}$ if there exist systems $M_1 = (Q_1,\Sigma_1,f_1)$ and $M_2 = (Q_2,\Sigma_2,f_2)$ with M realizable by $M_1\times M_2$ in the sense that there exists a function $\zeta: Q\to Q_1\times Q_2$ with the following property:

$$\forall q\in Q \qquad \forall\sigma\in\Sigma$$

$$\zeta(f(q,\sigma)) = (f_1(\zeta_1(q),\sigma),\ f_2(\zeta_2(q),\sigma))$$

where ζ_i denotes the projection of ζ on Q_i, $i=1,2$.

This work was supported by NSF Grant GJ-30126.

This definition is not the most general, in that it does not take into account the possibility of state splitting and merging. However it will, with slight qualification, suffice for the applications we intend.

Definition 1.5 Let $M = (Q,\Sigma,f)$ be a system and $\Sigma_1 \times \Sigma_2 \subseteq \Sigma \times \Sigma$. M is called persistent with respect to $\Sigma_1 \times \Sigma_2$ iff

$\forall q \in Q \quad \forall (\sigma,\pi) \in \Sigma_1 \times \Sigma_2$

$<f(q,\sigma)>$ and $<f(q,\pi)>$ implies $<f(q,\sigma\pi)>$.

M is called permutable with respect to $\Sigma_1 \times \Sigma_2$ iff

$\forall q \in Q \quad \forall (\sigma,\pi) \in \Sigma_1 \times \Sigma_2$

$<f(q,\sigma\pi)>$ implies $<f(q,\pi)>$.

M is called commutative with respect to $\Sigma_1 \times \Sigma_2$ iff

$\forall q \in Q \quad \forall (\sigma,\pi) \in \Sigma_1 \times \Sigma_2$

if $<f(q,\sigma\pi)>$ and $<f(q,\pi\sigma)>$ then

$f(q,\sigma\pi) = f(q,\pi\sigma)$.

Obviously the last condition is symmetric with respect to $\Sigma_1 \times \Sigma_2$.

The terms "persistent", "permutable", and "commutative" were first used in [1]. In the present sense, these terms would be respectively translated "persistent with respect to $\Sigma \times \Sigma - I$" (where I denotes the identity relation on Σ), "permutable with respect to $\Sigma_i \times \Sigma_i$", and "commutative with respect to $\Sigma \times \Sigma$."

Theorem 1.1 Let $M = (Q,\Sigma,f)$ be a system with $\{\Sigma_1,\Sigma_2\}$ a partition of Σ. Then M possesses a parallel decomposition with respect to $\{\Sigma_1,\Sigma_2\}$ iff M is persistent, permutable, and commutative with respect to $\Sigma_1 \times \Sigma_2$ and $\Sigma_2 \times \Sigma_1$.

For the proof of Theorem 1.1, we shall require the following:

Definition 1.6 Let $M = (Q,\Sigma,f)$ be a system with $\Sigma_1 \subseteq \Sigma$. Let $\Sigma_2 = \Sigma - \Sigma_1$. An instance of Σ_1 is a maximal subset Q' of Q such that $\forall q,q' \in Q$ there exists a (possibly empty) sequence $q = q_0,q_1,q_2,\ldots,q_r = q'$ such that for each $j > 0$ $\exists \sigma \in \Sigma_2$ with $f(q_{j-1},\sigma) = q_j$ or $f(q_j,\sigma) = q_{j-1}$.
In other words, q and q' are in the same instance of Σ_1 if there exists an edge sequence in the graph of f, without regard to the direction of the edges, connecting q to q' with labels not in Σ_1.

Clearly, given Σ_1, the set of instances of Σ_1 partition Q. Hence there is a corresponding equivalence relation R_1. To continue with the proof of Theorem 1.1, suppose that M has the three properties stated. Let R_1 and R_2 be the equivalence relations corresponding to the instances of Σ_1 and Σ_2. Let Q_1 be the R_2 equivalence classes of Q and Q_2 be the R_1 equivalence classes of Q. Define $M_1 = (Q_1,\Sigma_1,f_1)$ and $M_2 = (Q_2,\Sigma_2,f_2)$ where f_1 and f_2 are given by the following rules, letting $R_i[q]$ denote the R_i equivalence class of q.

$\forall \sigma \in \Sigma_1 \quad f_1(R_2[q],\sigma) = R_2[f(q,\sigma)]$

$\forall \sigma \in \Sigma_2 \quad f_2(R_1[q],\sigma) = R_1[f(q,\sigma)]$

We proceed to show that f_1 and f_2 are well-defined. By symmetry, it suffices to show for f_1 only. Suppose that q and q' are such that $q\,R_2\,q'$. We must show that $\forall \pi \in \Sigma_1$ if $<f(q,\pi)>$ then $f(q,\pi)\ R_2\ f(q',\pi)$. Let $q_0,q_1,q_2,\ldots,q_r$ be a sequence such that $q_0 = q$, $q_r = q'$, and for each $j > 0$ $\exists \sigma \in \Sigma_2$ with $f(q_{j-1},\sigma) = q_j$ or $f(q_j,\sigma) = q_{j-1}$. It follows that if $<f(q_0,\pi)>$ then for each $j > 0$ $<f(q_j,\pi)>$. We show this inductively by supposing that $<f(q_{j-1},\pi)>$. Then if $f(q_{j-1},\sigma) = q_j$, then $<f(q_j,\pi)>$ by the persistent property. If $f(q_j,\sigma) = q_{j-1}$, then $<f(q_j,\pi)>$ by the permutable property.

By the commutative property, if $f(q_{j-1},\sigma) = q_j$ then $f(q_j,\pi) = f(f(q_{j-1},\pi),\sigma)$. Similarly if $f(q_j,\sigma) = q_{j-1}$ then $f(q_{j-1},\pi) = f(f(q_j,\pi),\sigma)$. Hence the sequence $f(q_0,\pi),f(q_1,\pi),\ldots,f(q_r,\pi)$ is such that $\forall j > 0 \quad \exists \sigma \in \Sigma_2$ with

$f(f(q_{j-1},\pi),\sigma) = f(q_j,\pi)$ or $f(f(q_j,\pi),\sigma) =$

$f(q_{j-1},\pi)$. Hence $f(q,\pi)\ R_2\ f(q',\pi)$.

We must now show that M is realizable by $M_1 \times M_2$. Let $\zeta: Q \to Q_1 \times Q_2$ be defined by $\zeta(q) = (R_2[q],R_1[q])$. Then $\zeta(f(q,\sigma)) = (R_2[f(q,\sigma)], R_1[f(q,\sigma)]) =$ [by definition of f_1,f_2] $(f_1(R_2[q],\sigma),\ f_2(R_1[q],\sigma)) =$ [by definition of ζ] $(f_1(\zeta_1(q),\sigma),\ f_2(\zeta_2(q),\sigma))$. The first and last terms above give the desired equality.

Conversely, suppose M is parallel decomposable into M_1, M_2 with respect to the partition $\{\Sigma_1,\Sigma_2\}$. The reader may verify, by applying the definitions, that M is indeed persistent, permutable, and commutative with respect to $\Sigma_1 \times \Sigma_2$ and $\Sigma_2 \times \Sigma_1$.

Definition 1.7 An asynchronous system with output is the generic name for the following class of objects: $M = (Q,\Sigma,f,g,\Delta)$, where (Q,Σ,f) is an asynchronous system as in Definition 1.1, Δ is a finite output alphabet, and g is the output function of one of the following forms:

(1) $g: Q \to \Delta$

or (2) $g: Q \times \Sigma \to \Delta$

Output functions of the second type seem to be most applicable to asynchronous modules, is discussed in [5]. We will not discuss this

type further here. Output functions of the first type seem to be relevant to parallel program schemata, as discussed in the next section.

Definition 1.8 A system with output, $M = (Q,\Sigma,f,g,\Delta)$, is parallel decomposable into $M_1 = (Q_1,\Sigma_1,f_1,g_1,\Delta_1)$ and $M_2 = (Q_2,\Sigma_2,f_2,g_2,\Delta_2)$ if (Q,Σ,f) is decomposable into $(Q_1,\Sigma_1,f_1) \times (Q_2,\Sigma_2,f_2)$ with map $\zeta: Q \to Q_1 \times Q_2$ and

(1) if case (1) of Definition 1.7 holds then there is a function $h: \Delta_1 \times \Delta_2 \to \Delta$ such that

$$\forall q \in Q \quad g(q) = h(g_1(\zeta_1(q)), g_2(\zeta_2(q)))$$

(2) if case (2) of Definition 1.7 holds then there is a function $h: \Delta_1 \cup \Delta_2 \to \Delta$ such that $\forall q \in Q \quad \forall \sigma \in \Sigma$

$$g(q,\sigma) = \begin{cases} h(g_1(\zeta_1(q),\sigma)) & \text{if } \sigma \in \Sigma_1 \\ h(g_2(\zeta_2(q),\sigma)) & \text{if } \sigma \in \Sigma_2 \end{cases}$$

2. Parallel Decomposition of Parallel Program Schemata

The concept of a parallel program schema has been defined in various ways[1,2,3,4] and parallel compositions of schemata were discussed in [4]. For the purpose of this paper, we use the definitions below which correspond most closely to [3]. To avoid confusion, the reader is forewarned that what is called a "schema" in [2] corresponds to a "realization" in the present work, and what corresponds to a "schema" in the present work is represented by the function φ in [2]. Several terms concerning schemata will be alluded to here, but not formally defined. The reader is referred to [2] or, preferably, [3].

Definition 2.1 An operation set is a finite set $B = \{b,c,d,\ldots\}$ of elements called operations, together with:

(1) for each $b \in B$, a non-empty set of unique symbols $\Sigma(b) = \{b_1,b_2,\ldots,b_{K(b)}\}$ called terminators of b.

and (2) a symmetric relation $\rho \subseteq B \times B$.

For any $C \subseteq B$ define $\Sigma(C) = \cup\{\Sigma(b) \mid b \in C\}$. If B is understood then Σ denotes $\Sigma(B)$. Σ will be the alphabet of interest in relating schemata to asynchronous systems. Also, for $C,D \subseteq B$, write $C \rho D$ iff $\exists (c,d) \in C \times D$ such that $c \rho d$. We write $\bar{\rho}$ for the complement of ρ.

Definition 2.2 A parallel program schema (or simply schema) over an operation set B is a partial function $\varphi: \Sigma(B)^* \to 2^B$ such that the following are satisfied:

Axiom 1 $\forall x \in \Sigma(B)^* \quad \forall b \in B \quad \forall \sigma \in \Sigma(b)$

$$\langle \varphi(x\sigma) \rangle \text{ iff } b \in \varphi(x)$$

(Recall that $\langle \varphi(x) \rangle$ means "$\varphi(x)$ is defined").

Axiom 2 $\forall x \in \Sigma(B)^* \quad \forall b \in B \quad \forall \sigma \in \Sigma(b)$

$$\text{if } \langle \varphi(x\sigma) \rangle \text{ then } (\wp(x) - \{b\}) \subseteq \wp(x\sigma).$$

The function $\wp$ is interpreted as the parallel control of the set of operations B. An element $x \in \Sigma(B)^*$ represents the sequence of terminators of operations which have occurred and $b \in \wp(x)$ indicates that b is "enabled" in the current state as a result of the sequence x.

We will further assume throughout this paper that B is irredundant in the sense that for each $b \in B$ there exists an $x \in \Sigma(B)^*$ such that $b \in \varphi(x)$.

Definition 2.3 A realization of a schema φ over B (or simply "realization over B") is a quadruple $\psi = (Q, q_o, f, g)$ where $(Q,\Sigma,f,g,2^B)$ comprises an asynchronous system with output of the first type, and where $\forall x \in \Sigma^* \ \varphi(x)$ and $g(f(q_o,x))$ are either both undefined or are equal.

With any schema we may associate a countable-state free realization in a rather obvious way, cf.[3]. Furthermore, Axioms 1 and 2 may be translated into Axioms 1' and 2' below so that when any asynchronous system with output of the first type satisfies these axioms, there is necessarily a corresponding schema.

Axiom 1' $\forall q \in Q \quad \forall b \in B \quad \forall \sigma \in \Sigma(b)$

$$\langle f(q,\sigma) \rangle \text{ iff } b \in g(q).$$

Axiom 2' $\forall q \in Q \quad \forall b \in B \quad \forall \sigma \in \Sigma(b)$

$$\text{if } \langle f(q,\sigma) \rangle \text{ then } (g(q) - \{b\}) \subseteq g(f(q,\sigma)).$$

We will assume in the remainder of the paper that every state q is reachable from q_o. in the sense that $\exists x \in \Sigma^* \quad f(q_o,x) = q$.

The conditions for parallel-decomposability of realizations is now apparent. Given a realization (Q,q_o,f,g) over B, we must find a partition $B = B_1 \cup B_2$ with $B_1 \cap B_2 = \phi$ such that (Q,Σ,f) satisfies the conditions of Theorem 1.1 with respect to the partition $\Sigma(B_1) \times \Sigma(B_2)$ and furthermore Condition 1 of Definition 1.7 holds for the resulting decomposition, if any, where $\Delta_1 = 2^{B_1}, \Delta_2 = 2^{B_2}$, and $\Delta = 2^B$. An example is demonstrated in Figure 2.1.

Definition 2.4 Let φ be a schema over B. Define a relation $\approx$ on Σ^* by $x \approx y$ iff $\forall z \in \Sigma^* \quad \varphi(xz) = \varphi(yz)$. φ is called commutative if

$$\forall x \in \Sigma^* \quad \forall (\sigma,\pi) \in (\Sigma \times \Sigma) - I$$

$$\text{if } \langle \varphi(x\pi) \rangle \text{ and } \langle \varphi(x\sigma) \rangle \text{ then } x\sigma\pi \approx x\pi\sigma.$$

Definition 2.5 Let $\psi = (Q,q_o,f,g)$ be a

realization over B. Define the relation $\approx$ on Q by $q \approx q'$ iff $\forall x \in \Sigma^*$ $g(f(q,x)) = g(f(q',x))$. A realization is called commutative if $\forall q \in Q$ $\forall \sigma,\pi \in \Sigma$ $f(q,\sigma\pi) \approx f(q,\pi\sigma)$ whenever both are defined. A realization is called reduced if $q \approx q'$ implies $q = q'$.

It is not difficult to see that if φ is commutative then so is every realization of φ. Furthermore, every schema has a unique reduced realization, since Axiom 1' requires that realizations be treated as "completely specified" machines. Hence if we are dealing with reduced realizations, Definition 1.4 is completely adequate to model any conceivable notion of parallel control decomposition.

By a slight abuse of notation, we may write $\varphi_1 \times \varphi_2$, when φ_1 and φ_2 are schemata, to denote the schema which corresponds to the realization obtained by forming the parallel composition of two realizations, ψ_1 for φ_1 and ψ_2 for φ_2.

The concept of a "repetition-free" (or simply free) schema or a corresponding realization has been defined elsewhere[1,2,3] and this definition will not be repeated here. It should suffice to say that the "free" assumption is very useful in characterizing certain properties of schemata, for example "determinacy" as pointed out in Lemma 2.1 below. The reason for this is that a free schema has the property that any control sequence x for which $\varphi(x)$ is defined corresponds to a sequence which can occur as a computation in an interpreted schema.

Definition 2.6 A schema φ over B is called conflict-free if $\forall x \in \Sigma^*$ $\forall (b,c) \in \rho$ $\{b,c\} \not\subseteq \varphi(x)$.

Lemma 2.1[3] A commutative free schema is determinate iff it is conflict-free.

Definition 2.7 Let $x,y \in \Sigma^*$. By $x \vee y$ we mean the set of $z \in \Sigma^*$ such that z is obtained by interleaving all components of x and y in an arbitrary way, so long as the original order in each sequence is preserved. Also if $S_1, S_2 \subseteq \Sigma^*$ then

$$S_1 \vee S_2 = \cup\{x \vee y \mid x \in S_1,\ y \in S_2\}.$$

For example, if $S_1 = \{b_1c_1, b_1c_2\}$ and $S_2 = \{d_1e_2\}$, then $S_1 \vee S_2 = \{b_1c_id_1e_2, b_1d_1c_ie_2, d_1b_1c_ie_2, d_1b_1e_2c_i, d_1e_2b_1c_i, b_1d_1e_2c_i \mid i = 1,2\}$.

Lemma 2.2 If φ_1 and φ_2 are schemata over B_1 and B_2 respectively, and $B_1 \bar{\rho} B_2$ then

$$\{x \mid <\varphi_1(x)>\} \vee \{y \mid <\varphi_2(y)>\} = \{z \mid <(\varphi_1 \times \varphi_2)(z)>\}$$

Furthermore, if $z \in x \vee y$ then $\varphi(z) = \varphi(x) \cup \varphi(y)$.

Theorem 2.1 Suppose φ, φ_1, φ_2 are free schemata over B, B_1, B_2 respectively where $B = B_1 \cup B_2$ and $B_1 \cap B_2 = \phi$, and $\varphi = \varphi_1 \times \varphi_2$. Then φ is conflict-free iff φ_1 and φ_2 are conflict-free and $B_1 \bar{\rho} B_2$.

Proof Follows from Lemma 2.2.

Corollary 2.1 If φ is commutative then φ is determinate iff φ_1 and φ_2 are determinate and $B_1 \bar{\rho} B_2$.

Proof Theorem 2.1 and Lemma 2.1.

In[2,3] the concept of a closed schema was introduced. Such a schema corresponds to the notion of one which is maximally parallel. The closure of a schema is one which is equivalent (as defined in[2,3]) to it and closed.

Definition 2.8 Let φ be a schema over B. φ is called locally complete if

$$\forall x \in \Sigma^* \quad \forall (b,c) \in \bar{\rho} \quad \text{if} \quad b \in \varphi(x) \text{ and}$$

$$\forall \sigma \in \Sigma(b) \quad c \in \varphi(x\sigma) \text{ then } c \in \varphi(x).$$

We recall the following from previous papers.

Lemma 2.3[2,3] A free determinate schema is closed iff it is locally complete.

Lemma 2.4[3] For any free determinate schema there exists a unique closure.

Theorem 2.2 Suppose $\varphi, \varphi_1, \varphi_2$ are free determinate schemata and φ is decomposable into $\varphi_1 \times \varphi_2$. Then φ is closed iff φ_1 and φ_2 are closed.

Proof Lemma 2.2 and 2.3.

Theorem 2.3 Suppose $\varphi, \varphi_1, \varphi_2$ are free determinate schemata and $\varphi = \varphi_1 \times \varphi_2$. Suppose $\tilde{\varphi}, \tilde{\varphi}_1, \tilde{\varphi}_2$ are the respective closures of these schemata. Then $\tilde{\varphi} = \tilde{\varphi}_1 \times \tilde{\varphi}_2$.

Proof Theorem 2.2 and Lemma 2.4.

3. Other Types of Decomposition

Definition 3.1 Let $\psi = (Q, q_o, f, g)$ be a realization over B. A state $q \in Q$ is called a ϕ-state if $g(q) = \phi$. ψ is called separable if it has a unique ϕ state.

Definition 3.2 Let $\psi_1 = (Q^1, q_o^1, f^1, g^1)$ and $\psi_2 = (Q^2, q_o^2, f^2, g^2)$ be realizations over B with q a ϕ state of Q^1. Then a series composition of ψ_1 followed by ψ_2 (with respect to q) is a realization which is the disjoint union of ψ_1 and ψ_2, except that states q and q_o^2 are merged together in the obvious way (see Figure 3.1).

Clearly if ψ_1 is separable with ϕ state

Q_i there is a unique series composition of ψ_1 followed by ψ_2.

Definition 3.3 Let $\Omega = \{\psi_1, \psi_2, \ldots, \psi_n\}$ be a collection of realizations. A quasi-series composition of realizations in Ω is a realization ψ obtained by merging all but one initial state in Ω with some unique ϕ-state. The unmerged initial state becomes the initial state of ψ.

Quasi-series compositions, of course, include series compositions and others, such as the "iterate" described in [4].

Definition 3.4 Let $\psi = (Q, q_0, f, g)$ be a realization over B. An instance of an operation $b \in B$ is defined to be an instance (as in Definition 1.6) of $\Sigma(b)$.

In other words, an instance of b is a set of states which corresponds to a single enabling of b, since any states within the instance can be reached by strings not containing elements of $\Sigma(b)$.

Observation 3.1 To determine whether a realization $\psi = (Q, q_0, f, g)$ has a quasi-series decomposition, it is a simple matter of checking whether there is a disjoint collection of sets $Q^1 \cup Q^2 \cup \ldots \cup Q^n = Q$ such that each Q^i completely contains all instances which it intersects. An example is shown in Figure 3.2.

Observation 3.2 The analog of Theorem 2.3 does not hold for series compositions. For example, if ψ_1 and ψ_2 are two acyclic realizations over disjoint decision-free* operation sets, then the closure of the schema corresponding to the series decomposition of ψ_1 followed by ψ_2 is not the same schema as the one corresponding to the series composition of the closure realizations, $\tilde{\psi}_1$ followed by $\tilde{\psi}_2$. An example is shown in Figure 3.3.

We now present another type of composition which we will see properly contains all combinations of parallel and quasi-series compositions. This composition is obtained by combining schemata with the "fork" and "join" primitives, as described in [10,11]. For example, the equivalents of parallel and series compositions are indicated in Figure 3.4 (the series composition does not require either fork or join). The following definition generalizes.

Definition 3.5 Let $\Omega = \{\psi_1, \psi_2, \ldots, \psi_n\}$ be a set of realizations. An FJ graph over Ω is a directed graph with 3 distinguished types of nodes:

(1) F - (Fork) labelled node - exactly one input arc.

(2) J - (Join) labelled node - exactly one output arc.

(3) R - (Realization) labelled node - Each node is labelled with a unique element ψ of Ω. There is one input arc and the output arcs are in one-one correspondence with ϕ-states of ψ.

For the present, we will be concerned with FJ graphs in which the operations are decision-free and each R node corresponds to a single operation. Furthermore we will require that each graph be acyclic and that there be unique input and output arcs of the graph. We summarize these requirements by giving these graphs the name simple, abbreviated SFJ graph. Our reason for introducing this restriction here is that the semantics of SFJ are simpler to describe and the theorems to be presented are only applicable to this case.

Each SFJ graph represents, subject to a slight qualification to be stated below, a realization of a schema. This realization may be described in several ways. For example, the "parallel flowcharts" of[1] suitably restricted provide a description in terms of counter machines. The description which will be employed here will be in terms of the "marked graphs" of[12] and the reader is referred there for a formal description. However an example should suffice for the reader not already familiar with the obvious interpretation of FJ graphs.

Definition 3.6 Given an FJ graph Γ, a marking is a subset of the arcs of Γ. For an SFJ graph, the initial marking is defined to be the single input arc of the graph. A node n in a marked graph is firable with respect to a marking M if all of the input arcs to n are in M. If node n is firable in marking M, and M' is the marking obtained by replacing all of the input arcs of n with the output arcs of n, we write $M \xrightarrow{n} M'$. We say that a marking is stable if no F or J nodes are firable (i.e. only R nodes are firable). We write $M \xrightarrow{*} M'$ if M is unstable and there exist nodes $n_1, n_2, \ldots, n_k$ and unstable markings $M_1, M_2, \ldots, M_{k-1}$, such that $M \xrightarrow{n_1} M_1, M_1 \xrightarrow{n_2} M_2, \ldots, M_{k-1} \xrightarrow{n_k} M'$.

Definition 3.7 Let Γ be an acyclic directed graph. If α and β are arcs, we write $\alpha < \beta$ if α occurs before β in some directed path, and similarly if α and β are nodes. We write $\alpha \leq \beta$ if $\alpha < \beta$ or $\alpha = \beta$. A cut in Γ is defined to be a maximal set of arcs which are pairwise unrelated by $<$. (Clearly every cut in an FJ graph is a marking.) If Q and Q' are cuts, we write $Q \leq Q'$ iff $\forall \beta \in Q' \; \exists \alpha \in Q \; \alpha \leq \beta$. We say Q' is a minimal cut with a certain property if there is no other cut Q with this property such that $Q < Q'$.

Lemma 3.1 Let M be an unstable cut in an SFJ

* - i.e. each operation has exactly one terminator.

graph. There exists a unique stable cut M' such that $M \overset{*}{\rightarrow} M'$.

Proof Omitted.

Definition 3.8 For a cut M, let L(M) denote M if M is stable, and the unique stable cut M' such that $M \overset{*}{\rightarrow} M'$ otherwise. Given an SFJ graph Γ over B; the realization corresponding to Γ is given by $\psi = (Q, q_o, f, g)$ where:

(1) Q is a subset of the stable cuts of Γ.

(2) q_o is $L(M)$ where M = initial marking of Γ.

(3) g(q) = the operations of those operation nodes which have input arcs in q.

(4) $f(q,\sigma) = L(q')$ where, letting n be the node corresponding to operation b, $q \overset{n}{\rightarrow} q'$ (i.e. q' is obtained by firing n).

We implicitly assume that each SFJ is well formed in the sense that each operation appears at most once in each reachable cut.

An example illustrating a number of these definitions is given in Figure 3.5.

Theorem 3.1 There exist a (reduced, commutative, decision free) realization which is representable by an SFJ graph, but which is not series-parallel decomposable.

Proof View the example of Figure 3.5a. That this realization is not series-parallel decomposable may be verified by testing exhaustively using the criteria previously presented.

Theorem 3.2 There exists a (reduced, commutative, decision-free) realization which is not representable by an SFJ graph.

Proof Such a realization is shown in Figure 3.6. The proof then follows from the next theorem.

Definition 3.9 Let $\psi = (Q, q_o, f, g)$ be a realization over B. ψ is called orthogonal if for any $b \in B$ and any instance Q' of b $\forall q, q' \in Q'$ $q \neq q'$ implies $\exists \sigma \in \Sigma(b)$ $f(q,\sigma) = q'$.

The "orthogonality" condition is introduced for avoiding a technical pathology. It can be shown that any commutative, reduced realization is orthogonal, so not too much generality is lost.

Theorem 3.3 Let ψ be a reduced, commutative, finite-state realization. ψ is representable by an SFJ graph iff each instance of an operation in ψ contains a unique minimal state. Here "instance" is defined as in Definition 3.4 and "minimal" means with respect to the ordering < of nodes in the state graph (not in the FJ graph).

For the proof, we require the following lemmas.

Lemma 3.2 Let Γ be an SFJ graph and α an input arc of an R-node. There is a unique minimal stable cut (see Definition 3.7) containing α.

Definition 3.10 Let x and y be elements of Σ^*. We say x and y are similar if they contain the same number of each element of Σ.

Lemma 3.3 Let ψ be an acyclic, commutative, orthogonal realization of a decision-free schema. Then any two paths from a state q to a state q' are similar.

Proof Suppose that x and y are two different paths from q to q'. By induction on the length of x, $|x|$, we show that x and y are similar. If $|x| = 0$ the conclusion follows trivially, since the graph is acyclic and thus $x = y$. If $|x| > 0$ write $x = \sigma x'$. From the preceding statement, $|y| \neq 0$, so we write $y = \pi y'$. If $\sigma = \pi$ then applying the induction hypothesis, we have that x' and y' are similar. Hence x and y are similar. Suppose instead that $\sigma \neq \pi$. By an inelegant innumeration which involves the orthogonality assumption, all cases except $x' \neq o$ and $y' \neq o$ can be ruled out. Letting $x' = \alpha x''$ and $y' = \beta y''$, we have the diagram shown in Figure 3.7a. We may add the arcs shown in Figure 3.7b, where the existence of strings u and v may be determined by Axiom 2 and commutativity. By the induction hypothesis, the components of the pairs $(x'', \pi u)$, $(y'', \sigma v)$, and $(\alpha u, \beta v)$ are similar. Hence so are those of $(\sigma\alpha x'', \sigma\pi\alpha u)$, $(\pi\beta y'', \pi\sigma\beta v)$, and $(\sigma\pi\alpha u, \pi\sigma\beta v)$. But since similarity is an equivalence relation, $x = \sigma\alpha x''$ and $\pi\beta y'' = y$ are similar, which completes the induction step.

Proof of Theorem 3.3 Suppose that we have a realization representable by an AFJ graph. By Lemma 3.2, to each R-node there is a unique minimal stable cut containing the input arc of the R-node. Clearly the state corresponding to such a cut must be minimal with respect to other states in this particular instance of the operation corresponding to the R-node.

Suppose instead that we are given a realization such that every instance contains a unique minimal state. We demonstrate the construction of a corresponding FJ graph. From Lemma 3.3, any path from q_0 to a minimal state in an instance contains the same terminators, so we designate the instances by $b^1, b^2, \ldots$. Hence we precede the node corresponding to the instance in question by a J node, and connect to the output of each $b^1, b^2, \ldots$ an F node; and finally from each F node connect one arc to the J node. Some of these nodes and arcs may be redundant. Applying this process to each instance gives the desired FJ graph. We illustrate with Figure 3.8.

Theorem 3.4 Suppose B is a decision-free operation set and ψ is an acyclic finite-state,

free, determinate schema over B. Then the closure $\tilde{\ }$ of $\ $ has a realization which is representable by an SFJ graph.

Proof It is clear that $\tilde{\ }$ has an acyclic, finite-state, reduced, commutative realization. (The fact that commutativity can be assumed is demonstrated in [3].) It was shown in [3] that in a determinate schema any computation defines a partial order and for any two equivalent computations the partial orders must be the same. Since the closure of a schema is defined to be a schema which contains all equivalent computations, the computations of the closure are completely represented by the partial order. For any such partial order, there is an SFJ graph which bears a very close correspondence to the partial order. The corresponding realization contains all of the equivalent computations and therefore is a realization of the closure.

The use of a partial order to represent parallel computations pervades much of the current folklore. In many cases no justification for it is given. The results of this section present justifications of it since partial orders correspond closely to SFJ graphs, and also grounds for rejecting it. For example, Figure 3.6 demonstrates a meaningful parallel program schema which has been proved not to have a partial order representation (the state graph is a partial order, but this is a different sense). Such an example might arise if each operation requires one unit of some resource, but only two units are available. However we show that every suitably restricted closure, (or maximally parallel equivalent) is representable by a partial order.

Another area of interest concerns the removal of the assumption that the SFJ graph be simple. Then nodes can correspond to decisions, or realizations with more than one ϕ state. In this case it is useful to introduce another type of node: M (for merge), which allows recombination of mutual exclusive arcs. We call the result an FJM graph. This generalization introduces several new problems. First, semantics of an FJM graph are more difficult to describe, and detecting well-formedness is far from obvious. Some results on the latter may be found in [13,14,15,16]. Second, the nice partial-order quality of an FJ graph is lost, even in the acyclic case. In fact, it appears that a generalized version of the J node, such as that described in [5], is essential for representing maximally parallel realizations.

References

1. Karp, R.M. and R.E. Miller. Parallel program schemata. J. Computer and System Sciences, Vol. 3, No. 2, May 1969.

2. Keller, R.M. On maximally parallel schemata. IEEE Switching and Automata Theory Symposium, October 1970, pp 33-50.

3. Keller, R.M. Parallel program schemata and maximal parallelism. Princeton University, Department of Electrical Engineering, Computer Science Laboratory TR-104, TR-105, January 1972. Submitted for publication.

4. Brinsfield, W.A. and R.E. Miller. On the composition of parallel program schemata. IEEE Switching and Automata Theory Symposium, October 1971, pp 20-23.

5. Keller, R.M. Asynchronous modules: Properties and universality. Princeton University, Department of Electrical Engineering, Computer Science Laboratory TR-102, November 1971.

6. Muller, D.E. Asynchronous logics and application to information processing, in Switching Theory in Space Technology. Stanford University Press, 1963.

7. Muller, D.E. and W.S. Bartky. A theory of asynchronous circuits. Proceedings of an international symposium on the theory of switching. Annals of the computation laboratory of Harvard University, Vol. 29, Part 1, 1959, pp 204-243.

8. Clark, W.A. Macromodular computer systems. AFIPS Conference Proceedings, Vol. 30, Spring 1967, pp 335-336, and adjacent papers.

9. Ginzburg, A. Lectures on the Algebraic Theory of Automata, ACM, Academic Press, 1968.

10. Dennis, J.B. and E.C. Van Horn. Program semantics for multiprogrammed computations. CACM, Vol. 9, No. 3, March 1966, pp 143-155.

11. Conway, M.E. A multiprocessor system design. AFIPS Conference Proceedings, Vol. 24, Fall 1963, pp 139-148.

12. Commoner, F., A.W. Holt, S. Even, and A. Pnueli. Marked directed graphs. J. Computer and System Sciences, Vol. 5, No. 5, October 1971.

13. Keller, R.M. Analysis of implementation errors in digital computing systems. Technical Report No. 6, Computer Systems Laboratory, Washington University, St. Louis, March 1968.

14. Bruno, J. and S.M. Altman. Asynchronous control networks. IEEE Trans., Vol. C-20, No. 6, June 1971, pp 629-638.

15. Baer, J.L., D.P. Bovet, and G. Estrin. Legality and other properties of graph models of computations. J. ACM, Vol. 17, No. 3, July 1970, pp 543-554.

16. Hack, M.H. Analysis of production schemata. MIT Project MAC Technical Report TR-94, February 1972.

List of Figures

{1}* (1 = initial marking)	
{2,3} = q_0	
{2,5}*	
{3,4} = q_1	
{2,6,7} = q_2	
{2,7,8} = q_4	
{2,6,11} = q_5	
{2,8,11} = q_8	
{4,5}*	
{4,6,7} = q_3	
{4,6,11} = q_6	
{4,7,8}*	
{4,8,11}*	
{7,9} = q_7	
{9,11} = q_{10}	
{7,10} = q_9	
{10,11}*	
{12} = q_{11}	

Table I

The correspondence between cuts and states in Figure 3.5. * designates unstable cuts.

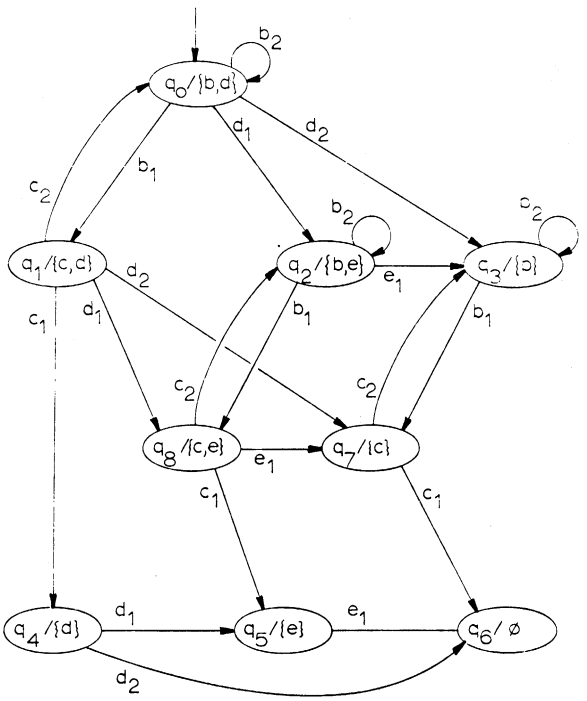


Figure 2.1(a)

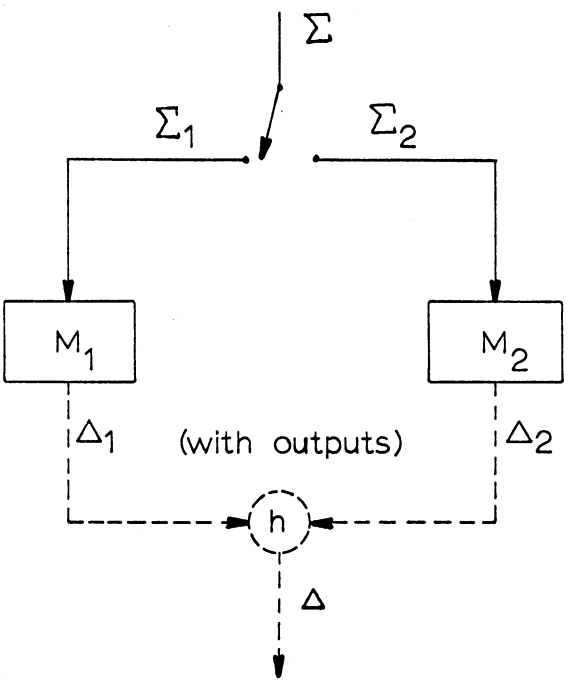


Figure 1.1

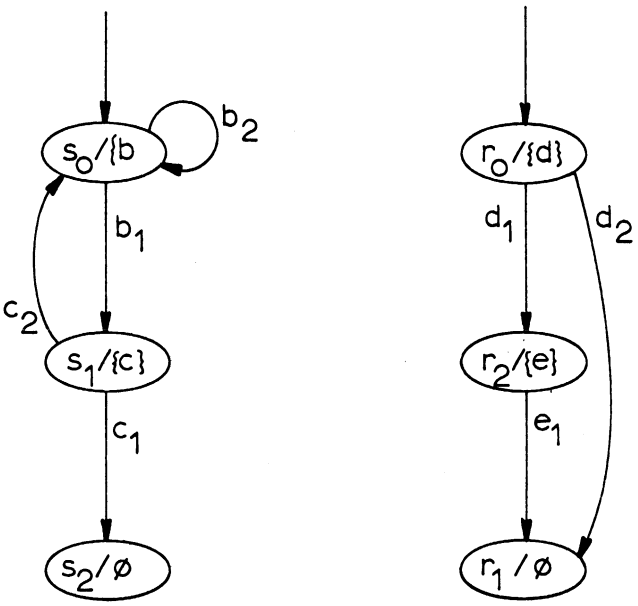


Figure 2.1(b)

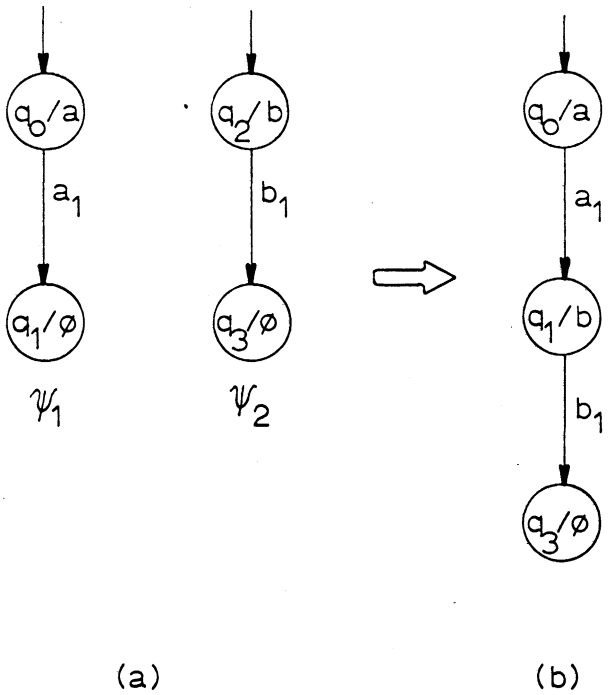


Figure 3.1

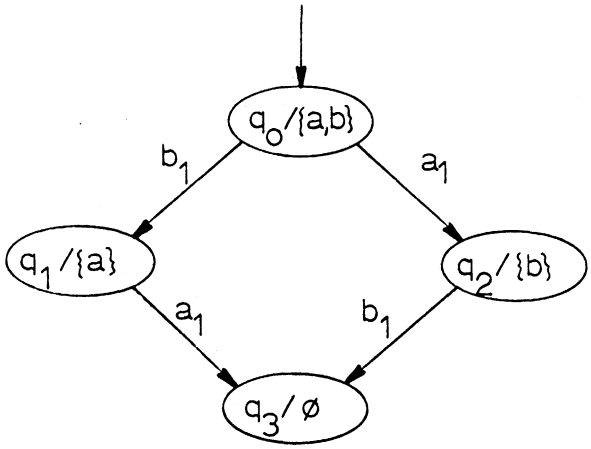


Figure 3.3

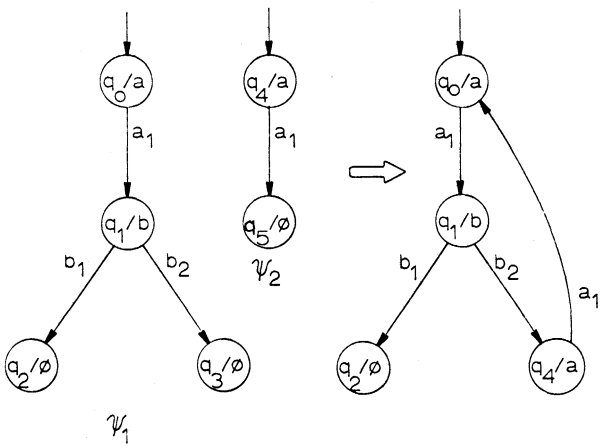


Figure 3.2

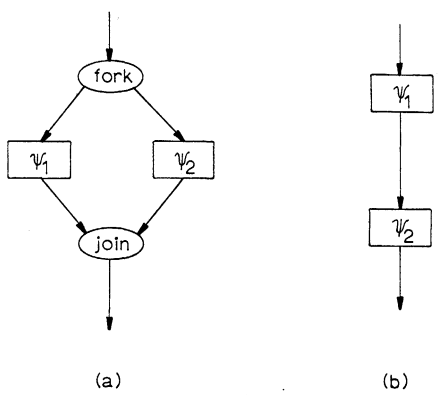


Figure 3.4

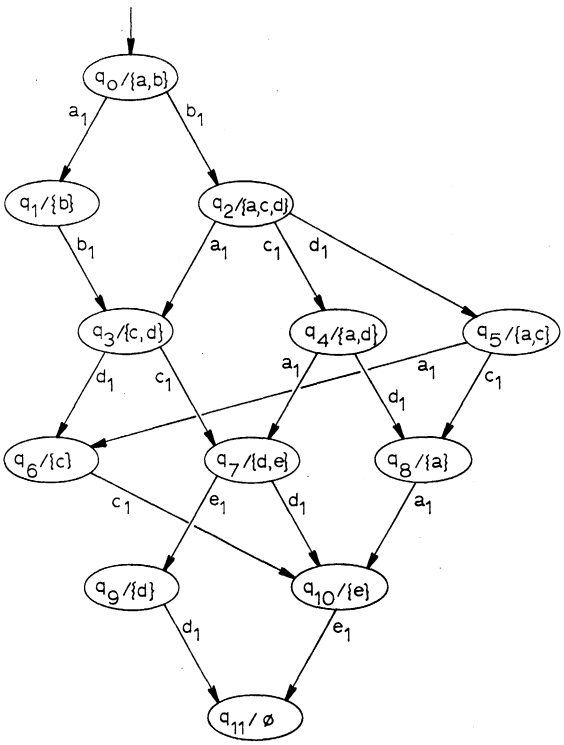


Figure 3.5(a)

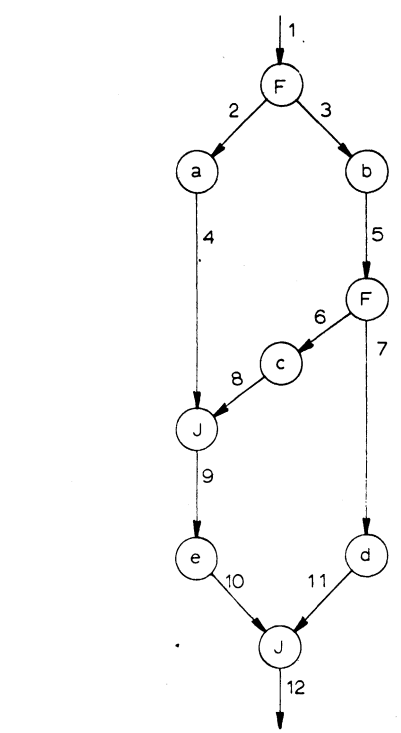


Figure 3.5(b)

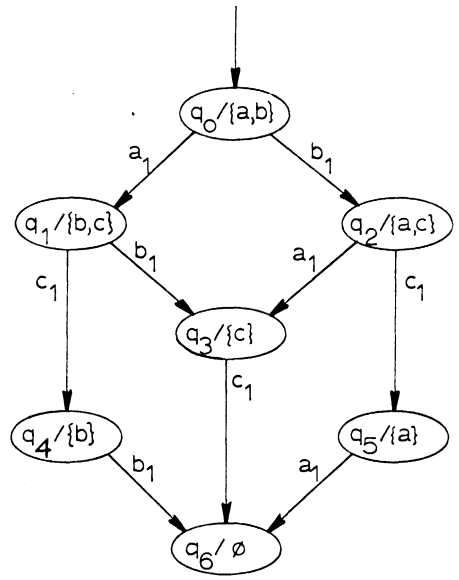
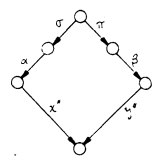


Figure 3.6



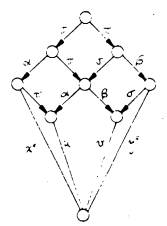


Figure 3.7

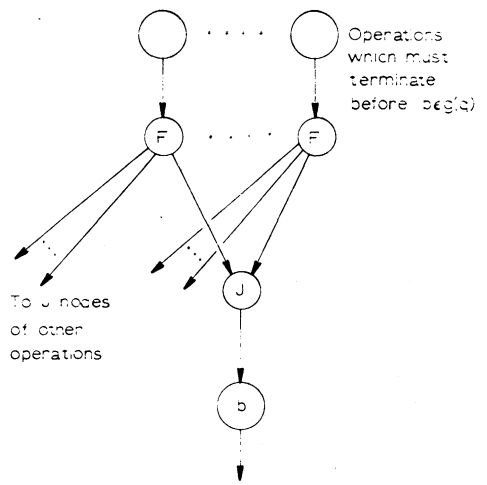


Figure 3.8