MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Computation Structures Group Memo 65-1

Concurrency in Software Systems

by

Jack B. Dennis

Notes Prepared for an Advanced Course on Software Engineering,
Technical University of Munich, February 1972

June 1972

This research was done at Project MAC, MIT, and was supported in part by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr N00014-70-A-0362, and in part by the National Science Foundation under grant GJ-432.

Concurrency in Software Systems

1. INTRODUCTION

A large program such as an operating system, a compiler, or a real-time control program is a precise representation of a system composed of many interacting parts or modules. Due to the size of these programs, it is essential that the parts be represented in such a way that the descriptions of the parts are independent of the pattern in which they are interconnected to form the whole system, and so the behavior of each part is unambiguous and correctly understood regardless of the situation in which it is used. For this to be possible, all interactions between system parts must be through explicit points of communication established by the designer of each part.

If two parts of a system are independently designed, then the timing of events within one part can only be constrained with respect to events in the other part as a result of interaction between the two parts. So long as no interaction takes place, events in two parts of a system may proceed concurrently and with no definite time relationship among them. Imposing a time relation on independent actions of separate parts of a system is a common source of overspecification. The result is a system that is more difficult to comprehend, troublesome to alter, and incorporates unnecessary delays that may reduce performance. This reasoning shows that the notions of concurrency and asynchronous operation are fundamental aspects of software systems.

In this lecture we consider a model for systems viewed as collections of concurrently operating subsystems that interact with one another through specific disciplines of communication. In many cases, we desire that such a system have a behavior that is reproducible in separate runs when presented with the same input data. This property of systems is known as determinacy. We shall present and illustrate an important result that if interactions between subsystems obey certain natural conditions, then determinacy of the subsystems guarantees

determinacy of the whole system. We conclude by illustrating the application of this result to systems of concurrent processes that interact by means of semaphores using the primitives P and V of Dijkstra.

2. PETRI NETS

During the discussion we will illustrate concepts by reference to particular examples of systems. Since we have found it convenient to use the formalism of Petri nets to represent these examples of systems, we begin with a brief introduction to the notation and semantics of Petri nets.

A Petri net [1, 2, 3] is a directed graph with two types of nodes called places and transitions. Each arc must go from a place to a transition or from a transition to a place. In drawing a Petri net, places are represented by circles and transitions by bars as in the example shown in Figure 1. Places from which arcs are incident on a transition

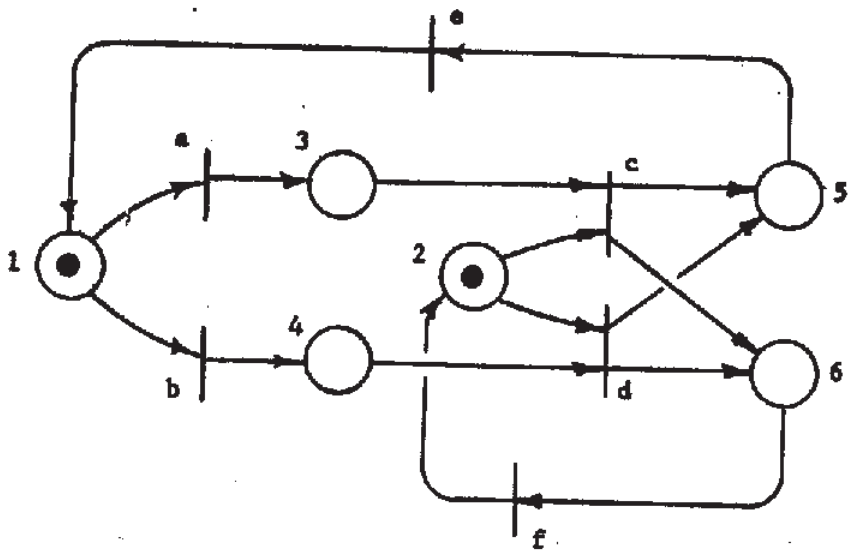


Figure 1. A Petri net.

are called input places of the transition, and places on which arcs from a transition terminate are called its output places. Each place may hold zero, one, or more markers or tokens. An arrangement of markers in a net is called a marking of the net. A Petri net may assume any series of markings consistent with the following simulation rule:

1. For a Petri net and a marking, each transition which has at least one token in each of its input places is enabled.
2. Any enabled transition may be chosen to fire.
3. Firing a transition consists of removing one token from each of its input places and adding one token to each of its output places.

Figure 2 shows a sequence of markings for a Petri net resulting from the firing of transitions in the sequence a,c,e,f. Note that this firing sequence returns the net to its original marking.

A marking of a Petri net is said to be safe if no simulation of the net, starting from the given marking, yields a marking in which some place holds more than one token. In a net having a safe marking no transition is ever enabled when tokens are present in any of its output places. A marking of a Petri net is live if, for any marking reachable from the given marking, there is a firing sequence that will enable any transition of the net. Liveness of a marked Petri net requires that no part of the net ever reach a condition from which further activity of the part is impossible. The Petri net in Figs. 1 and 2 is both live and safe for all of the markings shown.

If simulation of a Petri net reaches a marking for which two transitions are enabled that share an input place, the two transitions are said to be in conflict over the token in the shared place. In Fig. 2a transitions a and b are in conflict at place 1. In this case, the conflict is resolved by making an arbitrary decision as to which transition is to receive the token in place 1, hence this sort of conflict is called a free choice.

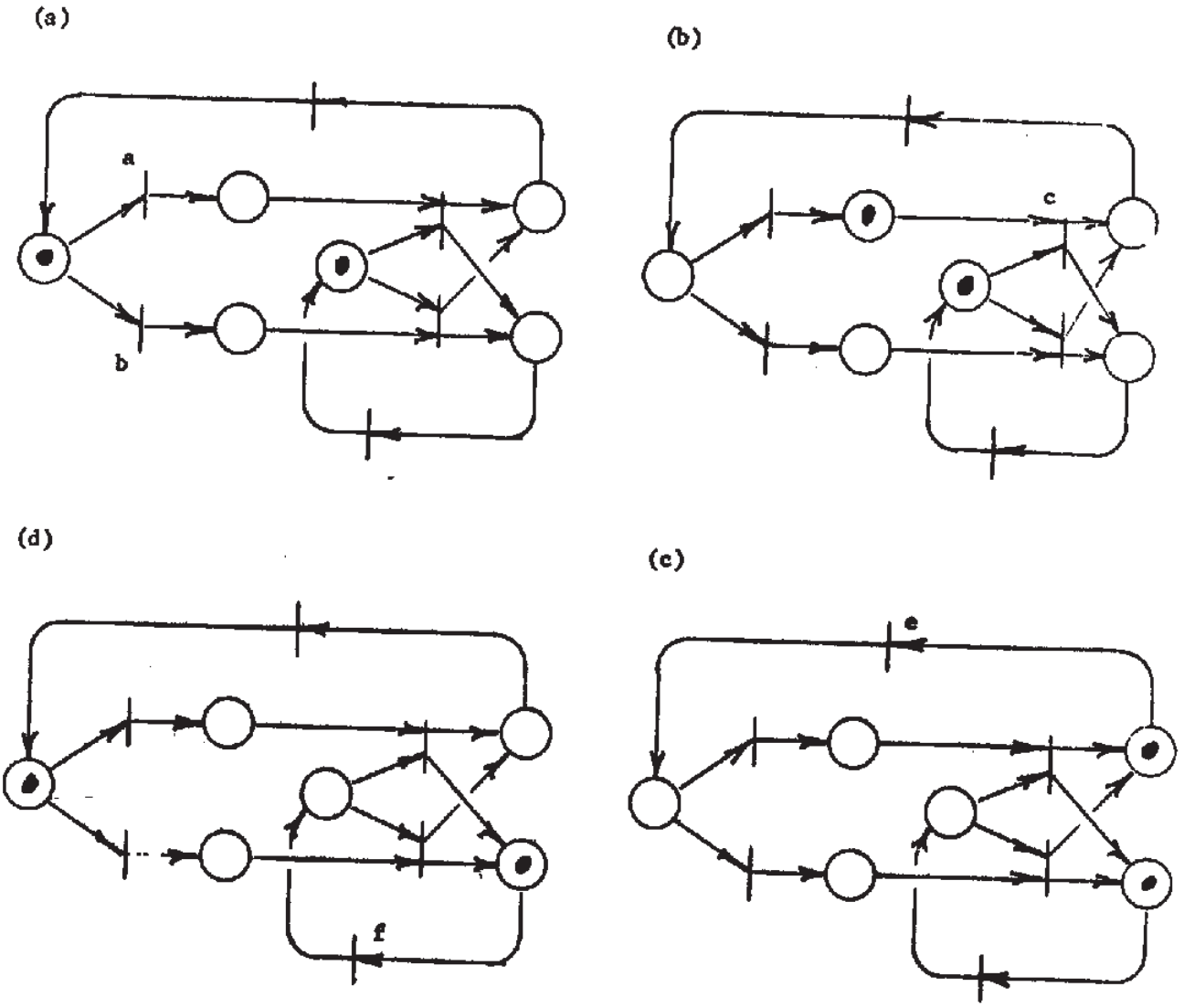


Figure 2. Simulation of a Petri net.

3. SYSTEMS

In Figure 3 we show a system S with m inlets and n outlets. The inlets are points at which the system receives signals from other systems or from the environment E in which the systems operate. The outlets

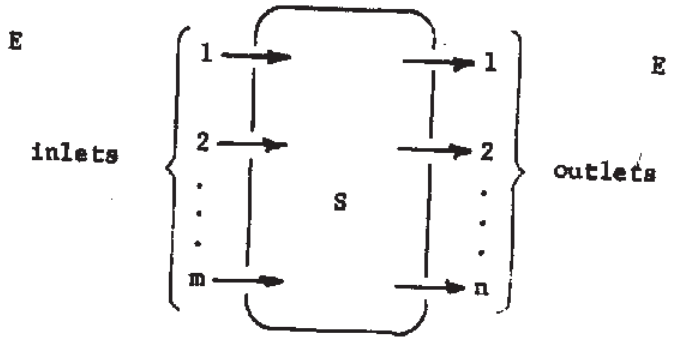


Figure 3. A system.

are points at which the system emits signals for reception by other systems or the environment. An alphabet of possible signals is associated with each inlet or outlet of the system.

Suppose system S begins operation from some internal configuration C_0 and makes transitions to successive configurations $C_1, C_2, \ldots, C_k, \ldots$. In some transitions symbols are absorbed at certain inlets; in other transitions symbols are delivered at outlets. Suppose the system has reached configuration C_i. During the activity from C_0 to C_i some definite sequence of signals was absorbed by S at each inlet, and some definite sequence of signals was delivered at each outlet, as shown in Figure 4. The array of input sequences U is called an input of the system; the array of output sequences V is a corresponding output of the system.

In this way, the behavior of a system is given by a binary relation R_S containing each pair (U, V) such that S has some finite activity, starting from configuration C_0, during which it absorbs the input array U and emits the output array V.

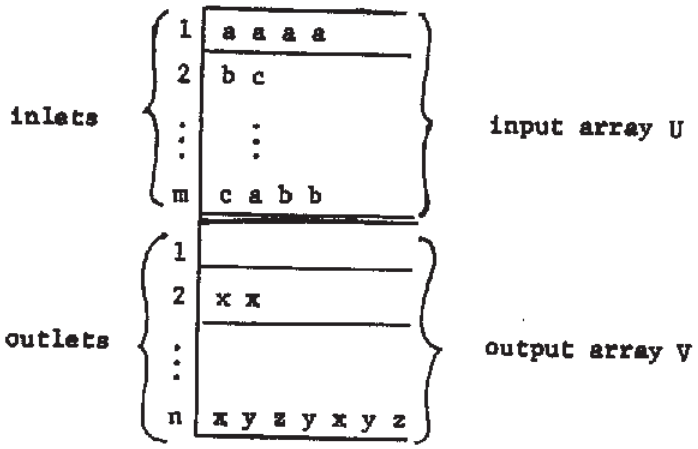


Figure 4

The domain of R_S consists of all inputs that can be absorbed by S during some activity starting from C_0. The domain does not include all possible arrays of finite sequences because S may cease absorbing signals at some inlets either temporarily or permanently. The range of the relation R_S contains each output S could emit for some input.

Let us consider some simple examples of systems to become familiar with the kinds of behavior that may occur. The system of Figure 5, shown in its initial configuration, transmits the pair of signals x,y for

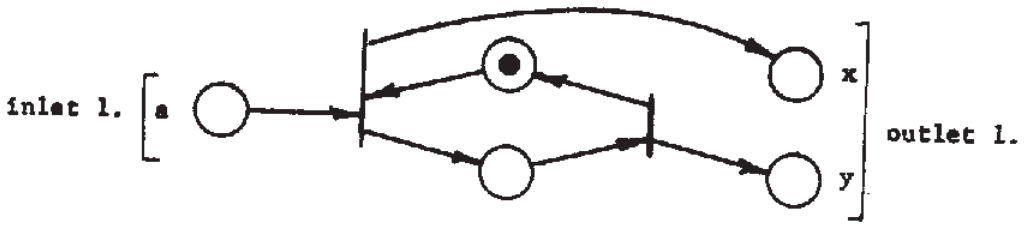


Figure 5.

each signal received at its inlet. Corresponding to the input

1. [a a]

the output may be any one of the three possibilities:

1. [x y] , 1. [x y x] or 1. [x y x y]

In this case, a bounded input can yield only bounded outputs. Figure 6 shows a system that can have unbounded outputs for certain bounded inputs.

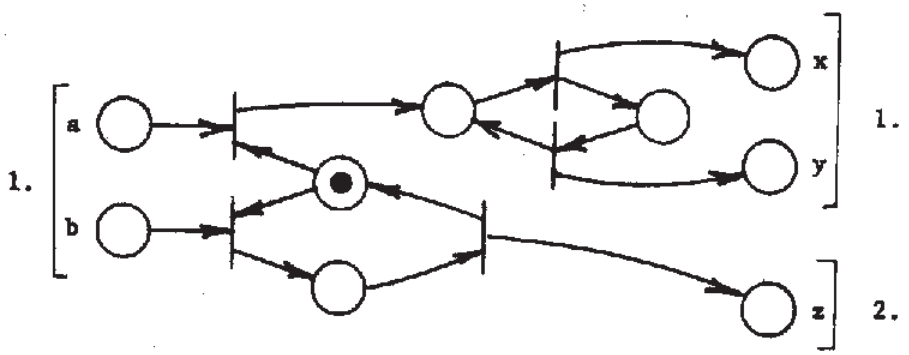


Figure 6.

For the input

1. [b a]

the output may be any member of the infinite set of arrays

1. 2. [z] 1. [x] 2. [z] 1. [x y] 2. [z] 1. [x y x] 2. [z] . . .

The example in Figure 7 shows how the order in which input signals are absorbed at different inlets may vary for different runs of a system

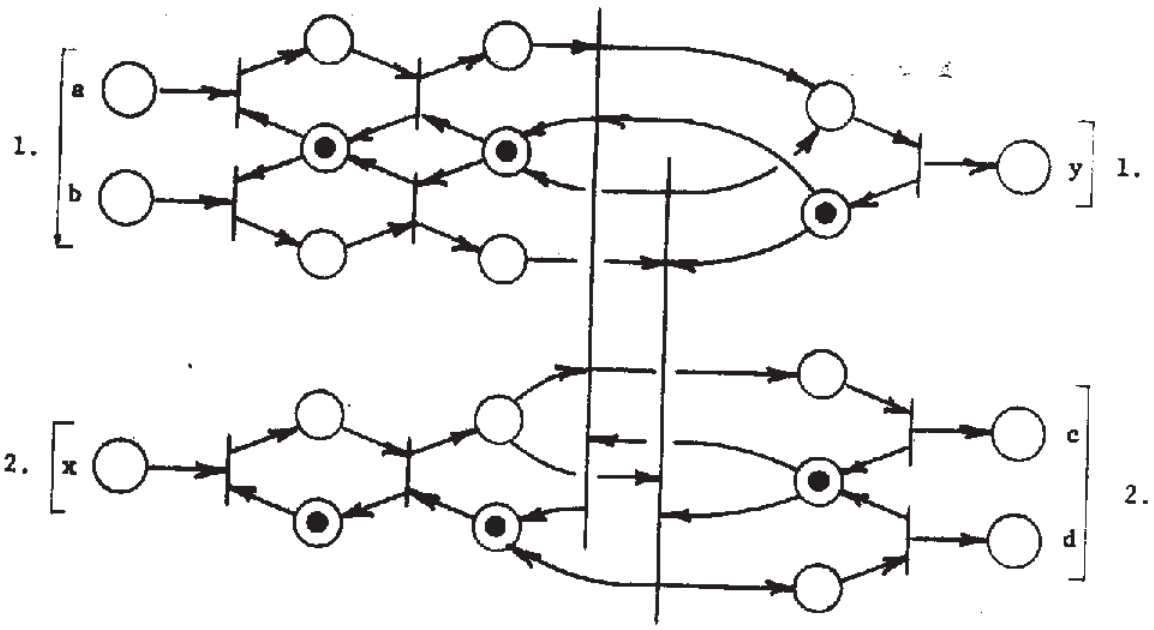
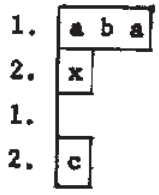
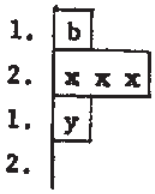


Figure 7.

without affecting the sequences that are emitted at each outlet. Two input-output pairs are shown below:





4. DETERMINACY

Next we introduce the ultimate output Y of a system S for some presented input X. We imagine that the symbol sequences of the array X are made available for absorption at the inlets of S. Some of these sequences may be infinite. Then an associated ultimate output of S is an output array Y emitted by advancing the activity of S as far as possible without absorbing input symbols beyond those in X. (In this activity, it may be that S does not absorb all of X.) To state this precisely, let us say that two sequences (possibly infinite)

$$x_1, x_2, \ldots, x_k, \ldots$$

$$y_1, y_2, \ldots, y_k, \ldots$$

are similar if $x_i = y_i$ for each index such that both x_i and y_i exist. Two arrays are similar if corresponding rows are similar. Then Y is an ultimate output of S for X if and only if

$$\left.\begin{array}{l}(U, V) \in R_S \\ U \text{ a prefix of } X \\ V \text{ similar to } Y\end{array}\right\} \text{ implies V is a prefix of Y}$$

Some examples of ultimate outputs for the systems shown in Figures 5, 6 and 7 are given below:

figure	presented input X	ultimate output Y
5	1. [a a]	1. [x y x y]
6	1. [b]	1. [ ] 2. [z]
6	1. [b a]	1. [x y x y x y .....] 2. [z]
7	1. [a b] 2. [x]	1. [y] 2. [c]

In these systems, there is a unique ultimate output for each presented input. We call any system having this property a determinate system. Some examples of systems that are not determinate are shown in Figures 8, 9, and 10.

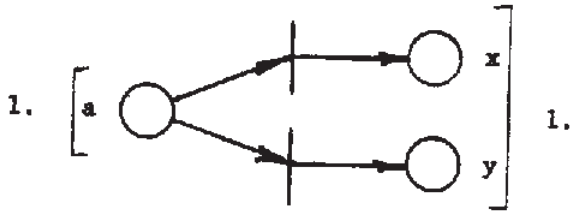


Figure 8

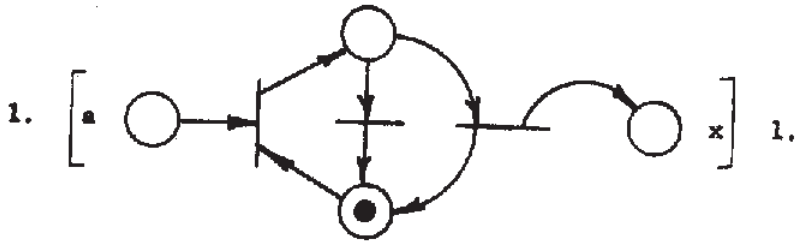


Figure 9

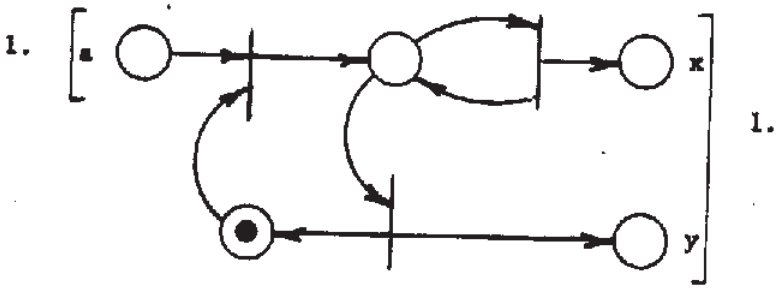


Figure 10

figure	presented input	ultimate outputs	
8	1. [a]	1. [x]	1. [y]
9	1. [a]	1. |	1. [x]
10	1. [a]	1. [x x y]	1. [x x x x ... .

5. INTERCONNECTED SYSTEMS

In the examples of systems used above, arrival of a signal at an inlet occurs when a marker is put in one of the places of the inlet. We assume that no further input signals arrive until the system absorbs the signal by removing the marker from the place. An output signal is emitted when the system puts a marker in one of the places of an outlet. We assume the marker is immediately removed by the environment in which the system operates.

Suppose a finite collection of systems $\{S_i\}$ are assembled to form a larger system S by specifying associations of certain outlets and inlets, as illustrated by Figure 11. We may define the input-output relation R_S

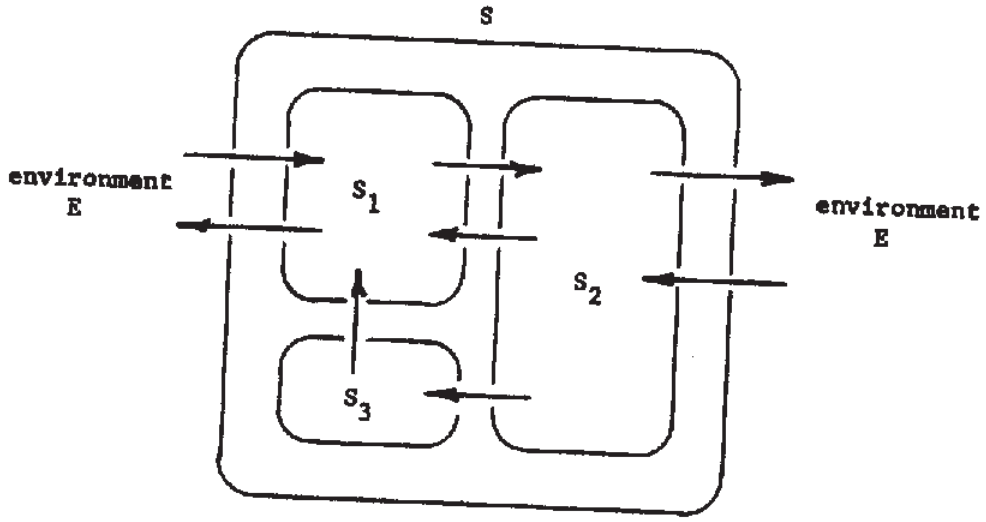


Figure 11

for the composite system by employing the following convention regarding the inputs and outputs of the constituent systems:

Suppose operation of S has reached a point where S has absorbed input U and emitted output V, and each subsystem S_i has absorbed input U_i and emitted output V_i. Then, if outlet p of S_i is associated with inlet q of S_j, the qth row of U_j must be a prefix of the pth row of V_i.

If the pth inlet of S_i is specified to be qth inlet of S, then row q of X and row p of X_i must be identical. If the pth outlet of S_i is specified to be the qth outlet of X, then row p of Y_i and row q of Y must be identical.

Using these conventions for defining the behavior of assembled systems, Patil [4] has established this important result:

Theorem A system S formed by the assembly of systems $\{S_i\}$ is determinate if each system S_i is determinate. That is, the class of determinate systems is closed under the operation of assembly.

If, in an assembly of systems, outlet p of S_i is associated with inlet q of S_j, then more signals may have been emitted by outlet p than have been absorbed by inlet q. Thus, to apply the above result, we must connect outlet p to inlet q in such a way that signals emitted by p are fed to q in exactly the same order, and no signals are lost. Two ways of accomplishing this are:

1. Insert an FIFO queue of unbounded capacity between outlet p and inlet q to hold signals emitted by p but not yet absorbed by q.
2. Prevent S_i from emitting a signal at outlet p until the previous signal emitted has been absorbed by S_j at inlet q.

Suppose outlets are connected to inlets by means of unbounded queues. Then an event that emits a signal at an outlet enters the signal in the associated queue; an event that absorbs a signal at an inlet removes a signal from the queue, and can only occur if the queue is not empty. Under this communication discipline, the Theorem shows that interconnections of determinate systems are necessarily determinate.

To prevent a system from emitting signals before a previous signal has been absorbed, it is sufficient that an assembly of systems satisfy the following condition:

α-condition: For each association of an outlet p of some S_i with an inlet q of some S_j, the assembly S must contain a path from inlet q to outlet p by way of systems in $\{S_i\}$ and the environment of S such that each signal emitted at outlet p requires the prior absorption of a signal at inlet q.

Figure 12 is an example of an assembly of systems that satisfies the α-condition. If it can be verified that an assembly S of systems satisfies the α-condition, then the Theorem guarantees that S is determinate.

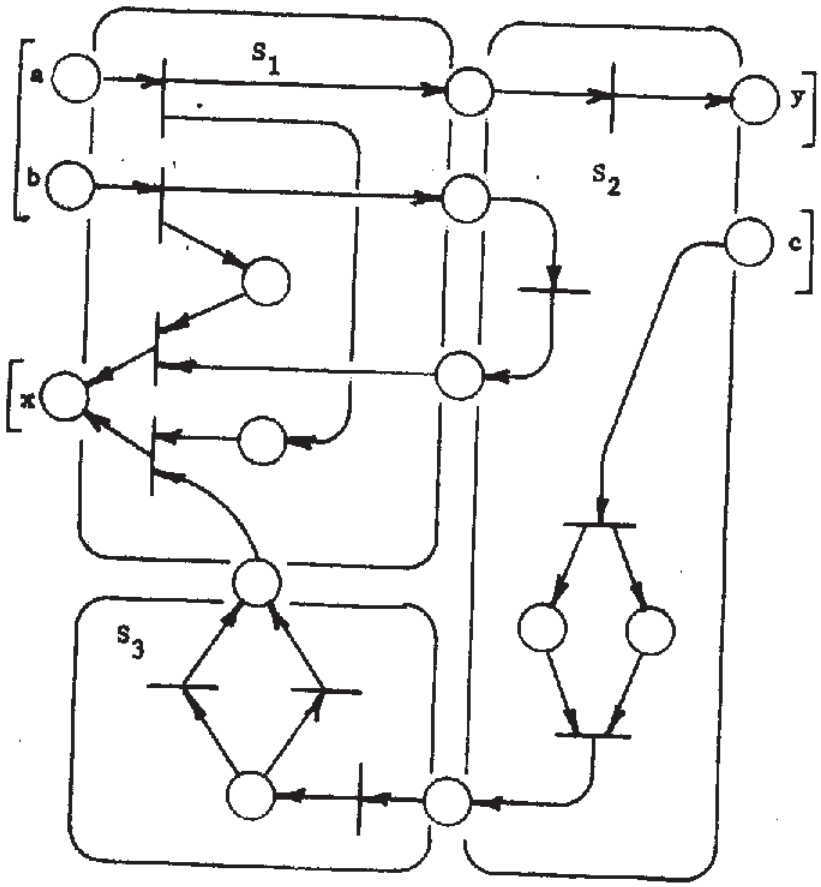


Figure 12

There is an important scheme for interconnecting systems that guarantees that the α-condition hold for the resulting system. The only kind of connection permitted between systems is a link that connects an output port of one system to an input port of another as shown in Figure 13. Each port consists of an inlet and an outlet. Systems are required to obey the discipline of emitting a signal at the outlet of a port only after receiving a signal at the associated inlet. In the initial configuration of a system, each output port is considered to have just received a (null) signal, and is prepared to emit a signal at the outlet of the port. Each input port is prepared to absorb a signal at the inlet, and will not emit a signal at the outlet until a signal arrives at the inlet. We call systems that communicate according to this discipline β-systems. Since any β-system satisfies the α-condition automatically, and any interconnection of β-systems is also a β-system, the Theorem shows that the class of determinate β-systems is closed under interconnection.

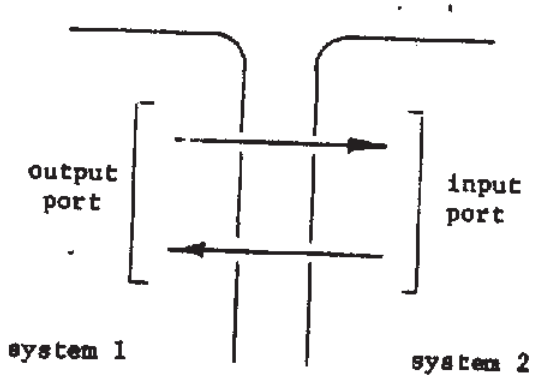


Figure 13

From Figure 14 we see that, since a FIFO queue is a determinate β-system, it is also true that determinate β-systems interconnected by queues yield determinate β-systems.

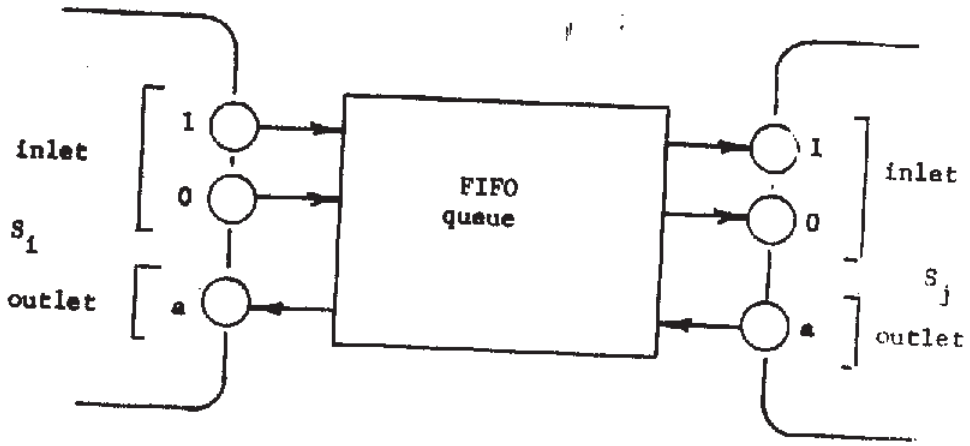


Figure 14

6. INTERPROCESS COMMUNICATION

A sequential process may be represented by a Petri net. An example is shown in Figure 15. Since there is one site of control, only one marker is ever present in the Petri net. Such Petri nets are called **state machines**. The location of the marker corresponds to the notion of "program counter" in a conventional computer.

(a) block diagram

(b) Petri net

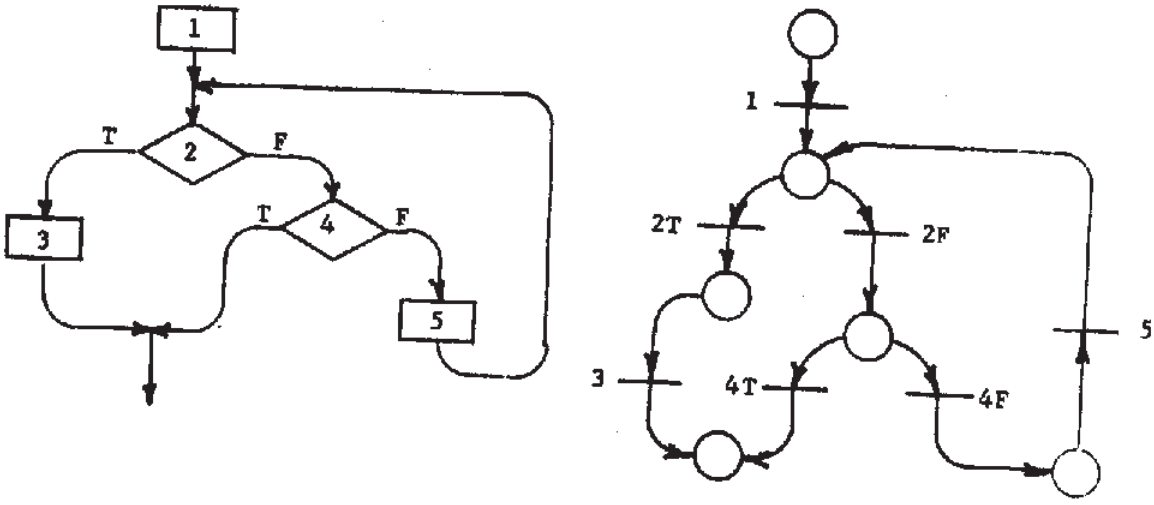


Figure 15

The synchronizing primitives of Dijkstra [5], as used to control the interaction of pairs of processes, may be represented as in Figure 16. The number of markers in place s represents the value of the semaphore.

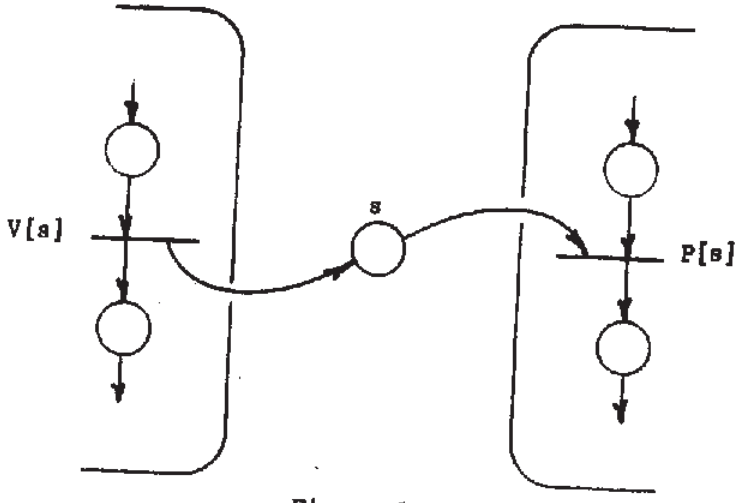


Figure 16

Suppose n sequential processes interact only in the two ways defined in Figure 17. Our development shows that such a system of processes is determinate.

(a) FIFO queue

(b) β-link

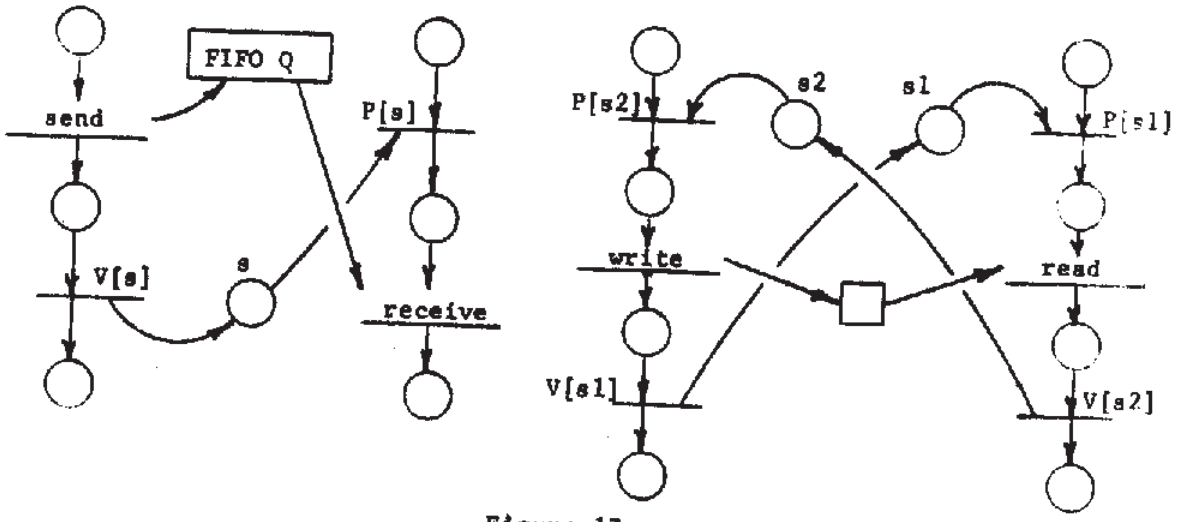


Figure 17

7. REFERENCES

1. C. A. Petri, Communication With Automata. Supplement 1 to Technical Report RADC-TR-65-377, Vol. 1, Griffiss Air Force Base, New York 1966. [Originally published in German: Kommunikation mit Automaten, University of Bonn, 1962.]

2. A. W. Holt and F. Commoner, Events and conditions. Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York 1970, pp 3-52.

3. A. W. Holt, F. Commoner, S. Even, and A. Pnueli, Marked directed graphs. J. of Computer and System Sciences, Vol. 5 (1971), pp 511-523.

4. S. S. Patil, Closure properties of interconnections of determinate systems. Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York 1970, pp 107-116.

5. E. W. Dijkstra, Co-operating sequential processes. Programming Languages, F. Genuys, Ed., Academic Press, New York 1968. [First published as Report EWD 123, Department of Mathematics, Technological University, Eindhoven, The Netherlands, 1965.]