MIT/LCS/TR-170

THE LOGIC OF SYSTEMS

by

Frederick Curtis Furtek

December 1976

This research was prepared with the support of the National Science Foundation under Grant DCR74-21822.

Massachusetts Institute of Technology

Laboratory for Computer Science

(Formerly Project MAC)

Cambridge Massachusetts 02139

THE LOGIC OF SYSTEMS

by

Frederick Curtis Furtek

Submitted to the Department of Electrical Engineering and Computer Science on July 31, 1978 in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

Abstract

We present a theory about the logical relationships associated with system behavior. The rules governing the behavior of a system are expressed by a Petri net. A set of assumptions about the modelling of a system permit us to separate system behavior into two components, what we refer to as information and control. Information is concerned with choices and how they are resolved. Control is concerned with the fixed, repetitive aspects of behavior - those aspects that are independent of choices.

We develop a concept of information that is nonprobabilistic. It is not inconsistent with Shannon's approach, but simply proceeds from a more basic idea: It deals with possibilities, rather than probabilities. Our approach embodies four common notions about information: (1) information distinguishes between alternatives; (2) it resolves choices; (3) it is transmitted and transformed within a system; (4) it says something about past behavior (memory or postdiction) and something about future behavior (prediction). We can identify those points at which information either enters or leaves a system, and we can trace information as it flows through a system.

The control component of system behavior is determined by a system's control structure, which is an event graph (marked graph). We show how the control structure of a system may be interpreted as the system's space/time framework.

When considered separately, the theories of information and control are of limited applicability. When brought together, they provide a technique for predicting and postdicting behavior.

Thesis Supervisor: Suhas S. Patil
Title: Associate Professor of Electrical Engineering and Computer Science

ACKNOWLEDGEMENTS

I wish to thank the following for their comments and suggestions: Anatol Holt, Suhas Patil, Ron Rivest, Robert Gallager, Mike Hack, and Fred Commoner. I also wish to thank Ellen Lewis for her perseverance in generating the text of this thesis, and Hannah Allen Abbott for her help in labelling the figures.

This research was prepared with the support of the National Science Foundation under Grant DCR74-21822.

TABLE OF CONTENTS

1. INTRODUCTION
1.1. System Models. 5
1.2. Petri Nets. 6
1.3. The Problem. 8
1.4. Background. 11
1.5. Outline of the Thesis. 14
2. PETRI NETS
2.1. Nets. 23
2.2. Petri Nets. 24
2.3. The Simulation Rule. 28
3. SYSTEMS
3.1. Assumptions. 39
3.2. A System. 40
3.3. The Parts. 43
3.4. The Control Structure. 48
3.5. The Modes. 55
3.6. An Initialized System. 59
3.7. Examples of Initialized Systems. 67
4. INFORMATION
4.1. Information Content. 75
4.2. Information Flow. 81
4.3. Resolving Choice. 90
4.4. Resolving Conflict. 92
5. CONTROL
5.1. Event Graphs. 101
5.2. Paths. 102
5.3. Cones. 113
5.4. System Space. 123
5.5. System Time. 129
6. PREDICTION AND POSTDICTION
6.1. Information and Control. 145
6.2. Transactions. 145
6.3. Extendible Simulations. 151
6.4. Prediction Graphs and Postdiction Graphs. 158
7. CONCLUSIONS
7.1. Evaluation. 170
7.2. Future Work. 171
References. 173
Index 175

CHAPTER I

INTRODUCTION

1.1 System Models:

The term system is taken to mean an assemblage of interacting elements. But that's not enough. Because the behavior of a system is always, to some extent, systematic or organized, there must exist rules that govern, at least partially, the interactions among elements. In many cases these rules are only implicitly understood, but when made explicit, they form a model[1] of the system. Some of the more common types of models are:

(a) a set of differential equations (where the interacting elements are 'infinitesimal')

(b) a finite-state machine

(c) a set of difference equations relating system levels and rates of flow (the model of 'System Dynamics')

(d) a description in plain English

Each of these has a domain to which it is particularly suited. The type of model we'll be working with is especially appropriate for those systems in which the transmission and transformation of information are the chief interests.

[1] 'Model' is used in two slightly different senses. It can refer to a detailed set of rules applicable to a particular system, or to a general set of rules applicable to a whole range of systems. A set of differential equations relating the electric and magnetic field intensity in a specific region of space is an example of the former sense, while Maxwell's equations are an example of the latter. Where the distinction is important, context will make clear the usage intended.

1.2. Petri Nets:

Petri nets are a class of system models. Like the first three types of models mentioned above, Petri nets are formal models, and they permit mathematical analysis of system behavior. However, unlike a set of differential equations, Petri nets are discrete, and only finitary methods are employed. Unlike finite-state machines, Petri nets make no attempt to describe a system in terms of a total, unstructured, system state. But rather they allow for a distributed system state in which many individual states may hold concurrently. Since Petri nets are a generalization of finite-state machines, they retain all the representational power of state machines - in particular, the ability to express alternatives. In relation to the model of System Dynamics, Petri nets are based on more primitive concepts and, therefore, permit a more general and powerful theory.

A Petri net is simply a bipartite, directed graph whose vertices are called states and events. (The term condition is interchangeable with 'state'.) In the graphical representation of a net, states are drawn as circles and events as rectangles. An example is given in Figure 1.1.[†]

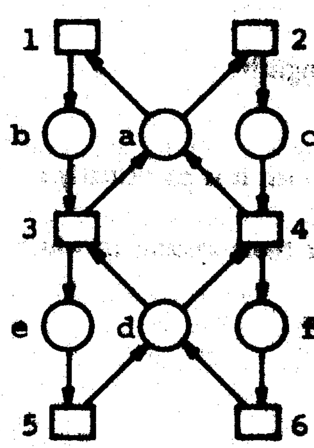


Figure 1-1 A Petri Net

[†] In Petri net examples, we shall adopt the convention of assigning lower-case letters to states and numbers to events.

We say that state *s* is a precondition of Event *e* if and only if there is an arc leading from *s* to *e*. Similarly, State *s* is a postcondition of Event *e* if and only if there is an arc leading rom *e* to *s*. Thus, in our example, states *b* and *d* are the preconditions of Event *3*, while states *a* and *e* are the postconditions.

Before we can use a net to simulate system behavior, we must first initialize it. This is done by designating certain states as initial conditions. These initial conditions are shown graphically by placing a 'token' on them:

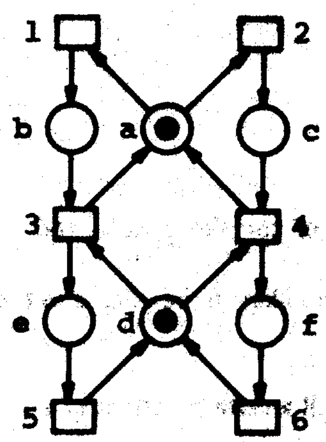


Figure 1.2 An Initialized Petri Net

The 'firing rule' for Petri nets states simply that when each precondition of an event holds (contains a token) then that event may occur (fire). The occurrence terminates a holding of (removes a token from) each precondition and initializes a holding of (places a token on) each postcondition. Notice that this is a strictly local operation involving only the event and its preconditions and postconditions. In general, there may be several events occurring independently and concurrently.

There is a special situation which deserves mention. When two events are concurrently enabled (their preconditions hold) but they have a common precondition, then we say that the two events are in <u>conflict</u>. Only one of them is permitted to terminate the holding of the shared state, and, therefore, the occurrence of either event will disable the other. In Figure 1.2, Events *1* and 2 are in conflict. Either one may occur. If Event 2 occurs, then the holding of State *e* is terminated and a holding of State *c* is initiated. This, in turn, permits Event *4* to occur, thereby producing yet another set of holdings. It should be apparent that there are, in general, many alternative simulations of a net (and that, in general, these simulations may be extended indefinitely).

1.3. The Problem:

For the <u>designers</u> and <u>users</u> of complex systems and for the <u>participants</u> in such systems, the bulk of the day-to-day problems are likely to be related to the following sorts of questions:

(1) Under what conditions will a certain pattern of behavior be produced?

(2) What are the consequences of a decision within the system?

(3) What are the effects of a system modification?

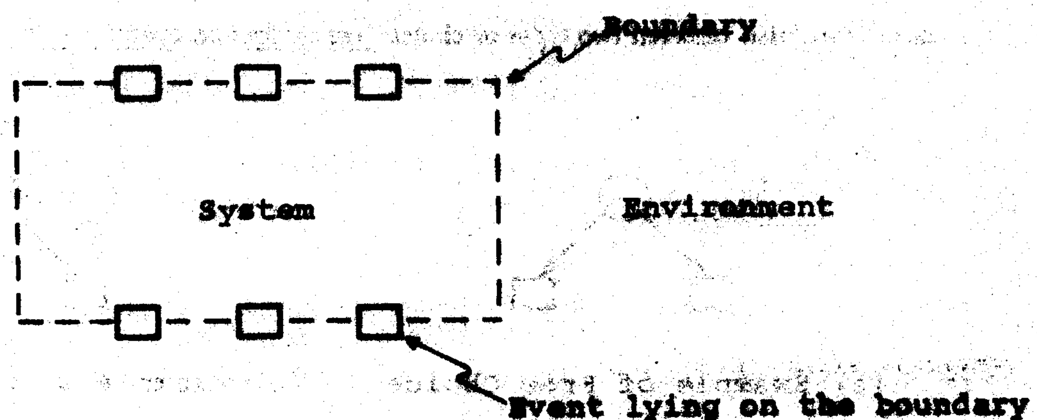
(4) How does behavior in one part of the system influence behavior in another part?

(5) How do the outputs of the system depend upon the inputs? (i.e., What is the 'function' of the system?)

Surprisingly, very little attention has been paid to these types of questions, which helps to account for the rather meager theoretical tools now available for getting answers.

The present work started out as an attempt to develop a technique for answering Question 5.

Petri nets had been chosen as the system representation. The problem, as it was then stated, was to find a way of determining the constraints that a net placed on the occurrences of a particular subset of events. If we view the system as embedded in some (unspecified) environment, then those events can be interpreted as lying on the system/environment boundary.



In this way we can arrive at a characterization of the external behavior of the system - a characterization that is independent of the particular mechanism by which that behavior is implemented. Defining the external behavior of a system as a set of constraints on the boundary events represents an attempt to get away from the restrictive notion of external behavior as a function from 'inputs' to 'outputs'.

There is, of course, a way of answering any question whatsoever about the behavior of a (bounded) system: exhaustive simulation. This involves cataloging all the different patterns of behavior. For very simple systems this is a practical approach and is, in fact, the usual approach. However, it becomes painfully clear that this method is extremely impractical for anything but the most elementary systems. A more desirable approach would be one that answered questions about behavior by examining the logical structure of the system, which, in our case, is the net representation.

It soon became apparent that there was one absolute requirement for success in our endeavor. Reduced to its simplest terms, it was this: we had to be able to determine how one choice influenced another. In a Petri net, a choice is represented by a shared state. For our purposes, we can distinguish between two types of choice: free choice and constrained choice.

(a) Example of Free Choice (b) Example of Constrained Choice

Figure 1.3 Choice

In the first case, the net in no way specifies how the choice is to be resolved, and this is how conflict, or nondeterminacy, is represented. In the second case, a holding of the shared state is paired with a holding of one of the two outside states to resolve the choice. Depending on whether the left or right hand state 'receives a token', either the left or right hand event will 'fire'. (We preclude the case where both outside states may hold concurrently.) Nets containing only free choices are, naturally enough, known as free-choice Petri nets. They are very amenable to analysis and their properties are well understood. This is because, in a free-choice net, no choice influences any other. As might be expected, free-choice nets were not general enough for our purposes. We needed constrained choices.

At a constrained choice, how a decision is resolved will depend upon some previous pattern of decisions at the free choices and other constrained choices. The problem was to determine this dependency. The situation was greatly complicated by the fact that we had to distinguish between repeated decisions at the same choice. The only hope appeared to lie in discovering the mechanism by which 'influence' propagates from one choice to another.

Although unrestricted nets are enormously general, they are also mathematically intractable, and there was no way of achieving our goals using unrestricted nets. So we had to find a set of restrictions that permitted us to trace the flow of 'influences' while still maintaining generality. Turing machines had shown that a model could be severely restricted without reducing its representational power.

1.4. Background:

Petri nets originated with the work of Carl Adam Petri who published the seminal paper Kommunikation mit Automaten in 1962. The model introduced there was later refined by Anatol Holt [10], and the modified structures were given the name 'Petri nets'. With Holt's work there has been a steadily increasing interest in nets, the key attraction being their ability to represent both concurrency and nondeterminacy.

Unrestricted nets have come to be recognized as an extremely general, systems model. But because unrestricted nets are mathematically intractable, the approach has been to study restricted classes of nets. If we eliminate either concurrency or choice, the problem becomes quite manageable.

State machines are a class of nets in which behavior is strictly sequential - there is no concurrency. In one form or another, state machines have been around for many years. They've been studied quite extensively and a great deal is known about their properties. State machines will be used in our theory to provide a notion of 'alternativeness'.

Marked graphs are the dual of state machines. They permit concurrency but exclude choice. Although they've not been studied quite as extensively as state machines, their properties are fairly well understood [3, 4, 7, 12, 16]. The most notable characteristic of a marked graph is that it can describe only a fixed, repetitive pattern of behavior. We'll make considerable use of this fact in our theory.

Free-choice nets are a generalization of both state machines and marked graphs. They permit both concurrency and choice. Some significant results have been obtained [3, 8], but, as we mentioned in the preceding section, the exclusion of constrained choices restricts generality.

An even larger class than free-choice nets are the simple nets. The mathematical analysis of this class of nets is considerably more difficult than for free-choice nets. Only a few results have been obtained [3, 17]. Patil's result [17] is significant since it shows that in spite of their power, simple nets are still not completely general. There is a class of coordination problems that they cannot model.

In working with this hierarchy of net subclasses it became increasingly clear that the key to the problem was the ability to trace the propagation of 'influences' within a net. Over a period of time, the 'propagation of influences' began to look suspiciously like the flow of information. This was an area in which Petri [19, 20, 21, 22] and Holt [13, 14, 15] had developed many ideas. Their major points may be summarized as follows:

(1) Information is (or at least ought to be) a system-relative concept.

(2) Information is what resolves choices. (This is a traditional view and the one adopted by Shannon.)

(3) Information is <u>gained</u> by a system at those points where there is 'forwards conflict' (nondeterminacy when the system is considered to be running forwards in time.) The information gained in a conflict situation specifies how the conflict is to be resolved.

(4) Information is <u>lost</u> by a system at those points where there is 'backwards conflict' (nondeterminacy when the system is considered to be running backwards in time). The information lost in a conflict situation specified how to back up one step.

Holt also recognized that information could be used as a tool for predicting and postdicting behavior. He and Commoner were successful in applying this idea to state machines [13]. Within the context of a state machine, they were able to identify 'information quanta' and trace their histories. The theory is consistent with all the points mentioned above. The significance of this work is that it established the principle that information could be formalized as a system-relative concept and that it could be used for predicting[†] and postdicting behavior. Of course, state machines are a very restricted class of structures, and there was clearly the need to apply notions of information flow to a much larger class of structures. Unfortunately, Holt and Commoner's work could not be generalized, at least not in a straightforward way. There were two key ideas missing.

One of these was recognizing the need to separate system behavior into two components, what we call <u>information</u> and <u>control</u>. The information component has to do with choices and how choices are resolved. The control component has to do with the fixed, repetitive aspects of behavior, those aspects that are unaffected by how choices are resolved. Holt very early recognized the need for making this distinction. What we refer to as the control component, he has referred to as the 'marked graph' aspects of behavior.[‡]

[†] Holt and Commoner's work actually deals only with postdiction. However, although it's not mentioned in the paper, there is a dual side to the theory dealing with prediction.

[‡] Private communication

Holt has continued his work in the area of nets, but it's too early to say how our approach relates to his since his new theory [34] is still in its formative stages and no new body of mathematics yet exists.

1.5. Outline of the Thesis:

Chapter 2 - Petri Nets

The basic structure we'll be working with is called a net. It's just a bipartite, directed graph. A Petri net is a net to which we attach a special interpretation and for which we define a 'simulation rule'. A state graph (state machine) is a Petri net in which each event has exactly one incident arc and one emergent arc. An event graph (marked graph) is a Petri net in which each state has exactly one incident arc and one emergent arc. A state component of a Petri net is a state-graph subnet in which all arcs connected to a participating state are used. An event component of a Petri net is an event-graph subnet in which all arcs connected to a participating event are used. A Petri net covered by state components (event components) is said to be state-graph decomposable (event-graph decomposable). We prove that if a Petri net is both SGD and EGD, then every state component and every event component is strongly connected.

The simulation rule generates a set of partial orders (simulations), each defining a causality relation among a set of state holdings and event occurrences. There are four ways in which two instances (holdings or occurrences) x and y may be related: x and y may be coincident (*i.e.*, the same instance), x may precede y, x may follow y, and x and y may be concurrent. We show that in a simulation, the instances associated with a '1-token' state component form a totally-ordered strand.

Chapter 3 Systems

Some basic assumptions about the modelling of a system are presented. These are translated into the following approach.

A system is is defined as a Petri net - the system net - together with (1) a set of subsets of states and (2) a set of subsets of events. With the help of five axioms (reflecting our assumptions), we're able to establish all the features of our model. The subsets of states are used to generate a covering of '1-token' state components, the parts of the system. The subsets of events are used to generate a covering of event components, the modes of the system. The parts are local structures associated with strictly sequential behavior. The modes are global structures associated with steady state behavior. The modes may be viewed as the 'natural frequencies' of the system.

To extract the control component of system behavior, we first generate the alternative classes of the system. This is done by 'collapsing' each of the parts. The alternative classes partition the states and events of the system net. There are two types of alternative classes: those that contain just states - they're called links - and those that contain just events - they're called meetings. The alternative classes induce a quotient net that is an event graph. This is the control structure of the system. The meetings form the events of the control structure, while the links form the states. As with any quotient system, the control structure loses certain features of the original system. What's lost is just the ability to distinguish between alternatives. For each system simulation (simulation of the system net), there is a corresponding control simulation (simulation of the control structure), and the two simulations are isomorphic. The second simulation is obtained from the first by replacing each instance of an element with an instance of the alternative class to which that element belongs.

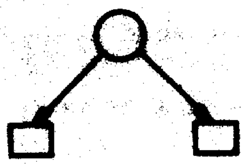
Distinguishing between alternatives is the domain of information. That's the topic of

Chapter 4, and in that chapter modes play a fundamental role. We note here an important property of modes: each mode intersects each alternative class in exactly one element. As a direct result of this, each mode is isomorphic to the control structure, and, therefore, the modes are isomorphic to each other. This means that a system net can be viewed as an interconnection of isomorphic event graphs.

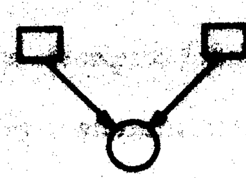
Chapter 4 Information

The control structure provides a framework for a system. On that underlying framework is 'superimposed' the informational component of behavior. To distinguish between alternatives we introduce the notion of 'informational content'. The <u>informational content</u> of a system element (state or event) is just the set of modes <u>excluded</u> from that element (i.e., those modes that do not contain that element). No two (distinct) elements in the same alternative class can have the same information content. So now a system element can be uniquely identified by specifying two things: (i) the <u>alternative class</u> to which it belongs and (ii) its <u>information content</u>. If we associate a color with each mode, then we can think of information as colors assigned to the 'tokens' on the control structure. By defining appropriate color transformations for each event, we can recapture the complete behavior of the system.

Our peculiar way of defining the concept of information pays off in allowing us to associate information with the resolution of choices. Since our theory is symmetrical with respect to the forwards and backwards directions of time, we distinguish between <u>forwards choices</u> and <u>backwards choices</u>.



A Forwards Choice



A Backwards Choice

Certain choices will involve *conflict*, which, in our case, is equivalent to free choice. We define the *information gain* of an event to be the information content of the event minus the combined information content of the event's *preconditions*. (The information content of an event will always contain the information contents of its preconditions and postconditions). The *information loss* of an event is defined to be the information content of the event minus the combined information content of the event's *postconditions*. We have the following two theorems:

> The information *gain* of an event *e* is the set of modes covering those events in *forwards* conflict with *e*.
>
> The information *loss* of an event *e* is the set of modes covering those events in *backwards* conflict with *e*.

This means that information is *gained* by a system at precisely those points where there is *forwards* conflict, and is *lost* by a system at precisely those points where there is *backwards* conflict. Furthermore, the information gained or lost in a conflict situation is equivalent to specifying how the choice is resolved. The same sorts of ideas apply to constrained choices, except now the information to resolve the choice is supplied by the system.

Chapter 5 Control

The control component of system behavior is entirely determined by the control structure. Since the control structure is an event graph, the theory of control is the theory of event graphs. We should mention that virtually all of our results pertain to event graphs covered by basic circuits (elementary circuits containing exactly one token). This is not a limitation for us since the images of the system parts within the control structure form a covering of basic circuits.

We begin by establishing the cyclic nature of event-graph simulations. We then show an important relationship between the paths in an event graph and the paths in a corresponding simulation†. This permits us to expose the symmetrical relationship that exists between two ordered occurrences in an event-graph simulation: If q_1 is an occurrence of e_1 and q_2 is an occurrence of e_2, then,

q_1 is the k'th occurrence of e_1 preceding q_2

iff

q_2 is the k'th occurrence of e_2 following q_1

(Note that because we're dealing with event graphs covered by basic circuits, all instances of the same event are totally ordered.) The value of k in this result is related to the structure of the event graph through the concept of synchronic delay. The synchronic delay of a path μ connecting two events in an event graph is the 'token loading' on μ minus the minimal token loading on those paths having the same endpoints as μ. The following proposition is equivalent to the two given above.

> There exists a path from q_1 to q_2 whose 'image' in the event graph is a path with a synchronic delay of k-1.

† Remember that a simulation is a partial order

Because synchronic delay is not a convenient concept to work with, we introduce the concepts of 'back cone' and 'front cone.' The back cone of an event e in an event graph is the set of states s such that: there does not exist a path of delay zero terminating at e and whose first state is s. The front cone of e is the set of states s such that: there does not exist a path of delay zero originating at e and whose last state is s. There is a simple relationship between synchronic delay and back and front cones:

> The synchronic delay of a path μ (in an event graph) is equal to the number of times the back boundary of μ's head is crossed.
>
> The synchronic delay of a path μ is equal to the number of times the front boundary of μ's tail is crossed.

Cones have an extremely interesting connection to the simulations of an event graph. The states of a particular cone define a series of cone-like slices in each simulation. If it's a back cone, then these cone-like surfaces point forwards, and if it's a front cone, then they point backwards. At the tip of each 'cone' is an occurrence of the related event. For that occurrence, the 'cone' provides a boundary between the past and 'not past' - if it's a back cone - or the future and 'not future' - if it's a front cone. Between any two consecutive 'cones', there is exactly one occurrence of each event.

System space is associated with the notion of 'synchronic distance'†, which is a measure of the 'slack' between two events. The synchronic distance between two events in an event graph is the minimal token loading on those circuits containing both events. When an event graph is strongly connected and free of blank circuits (circuits without any tokens), synchronic distance defines a metric on the set of events.

† Do not confuse synchronic delay and synchronic distance.

System time can be introduced for those event graphs satisfying three simple properties: The event graph must (i) be connected, (ii) be covered by basic circuits, and (iii) satisfy the *synchronic property*: the length of each elementary circuit is proportional to the number of tokens on it. For such an event graph we define the *phase relation*, an equivalence relation on the states and events. The elements within an equivalence class are said to be *in phase*, and an equivalence class of states is called a *phase*. The phase relation induces a quotient net that is an elementary circuit. This *fundamental circuit* may be viewed as the 'system clock'. For each simulation of a 'synchronous' event graph, we can partition the occurrences into slices called *instants of time* - and partition the holdings into slices called (elementary) *intervals of time*. These two types of slices strictly alternate. Occurrences within the same time slice are said to be *simultaneous*. The *time interval* between two occurrences is simply the number of elementary time intervals separating the two.

Chapter 6: Prediction and Postdiction

As mentioned earlier, for each system simulation there is a corresponding control simulation, and the two are isomorphic. We consider a pair of corresponding simulations. Because the control structure is an event graph, the control simulation has the regular properties described in Chapter 5. Since the system simulation is isomorphic to the control simulation, it too has these regular properties. Of course, the system simulation has certain 'irregular' properties, but these are describable using the concepts of information flow. We take advantage of the regular properties and the properties of information flow to develop a technique for predicting and postdicting the irregular aspects of the system simulation.

Within the control simulation, there is a total ordering among occurrences of the same meeting. For each such total ordering, there is a corresponding total ordering in the system simulation among occurrences of those events belonging to the related meeting. If, within the system simulation, the *n*'th occurrence associated with Meeting *m* is an occurrence of Event *e*, then we say that *e* is the <u>*n*'th transaction at Meeting *m*</u> (for that system simulation). If *q* is a holding or occurrence in the system simulation, then we can speak of the <u>*n*'th transaction at Meeting *m* relative to *q*</u>. In this case, *n* will be either negative or positive, depending upon whether the occurrence associated with the transaction is before or after *q*.

We now consider the following problem:

> We know that *q* is an instance in some system simulation and that *q* is associated with the alternative class *c*. If we also knew which element in *c* *q* was an instance of, what would this additional knowledge tell us about the possible patterns of transactions prior to *q* and subsequent to *q*?

The additional knowledge allows us to identify an element from among its alternatives. This is exactly the same thing that our notion of information content does. So the 'information content' of the additional knowledge is equivalent to the information content of the element identified. Anything that can be deduced from one can be deduced from the other, and vice versa. Of course, we can't say anything about how far the simulation containing *q* extends, either forwards or backwards, but we can say that the patterns of transactions must be <u>consistent</u> with certain requirements. Our approach to the problem is to try to characterize all the ways in which the information associated with *q* could have gotten there (postdiction) and all the ways in which it could have emanated from there (prediction). Information content consists of a set of excluded modes. Since information is 'additive', we can treat each mode in the information content of *q* separately and then merge the results.

For each excluded node, we know that by tracing the evolution backwards (forwards) we're going to generate a subnet of the simulation radiating into (out of) φ. This subnet defines a partial history of the transactions prior (subsequent) to φ. In general, there will be several such partial histories possible. In fact, some of the partial histories may be extendable arbitrarily far, in which case, there may be an infinite number of distinct histories possible. But since we're dealing with finite systems, there is a finite way of characterizing the set of (forwards and backwards) partial histories associated with each excluded node. The cones described in Chapter 5 may be used to 'slice up' the forwards and backwards partial histories -- back cones for backwards histories and front cones for forwards histories. This produces a finite set of 'history segments'. To characterize the set of forwards or backwards partial histories, we need only describe how the appropriate segments may be connected together. We do this with a state graph, each 'state' corresponding to a particular segment. For a postdiction net, each path terminating at a 'finishing state' defines a possible backwards partial history. For a prediction net, each path originating at a 'starting state' defines a possible forwards partial history. In both cases, the transactions associated with the n'th node in a path will be the n'th transactions in the corresponding partial history.

Chapter 7 Conclusions

We discuss aspects of the theory and the metatheory. The discussion of theory is mainly concerned with the future direction of the mathematics. In the area of metatheory, we touch upon four related topics:

(1) foundations (Is there a set of 'self-evident' axioms for the theory?)
(2) semantics (What meanings can we attach to the concepts of the theory?)
(3) methodology (How is the theory to be applied?)
(4) scope (For what problems is the theory suited?)

CHAPTER 2

PETRI NETS

2.1. Nets:

Nets are the basic structures of our theory.

Definition: A net is an ordered triple of sets $\langle A, B, C\rangle$ in which $A \cap B = \phi$ and $C \subseteq A \times B \cup B \times A$. $A \cup B$ is the set of elements of N. (N will be viewed as a bipartite, directed graph with $A \cup B$ as the vertices and C as the arcs.)

Notation: In the context of a net structure $\langle A,B,C\rangle$, we write $x \to y$ to mean $\langle x, y\rangle \in C$. For $x \in A \cup B$,

$$^{\bullet}x = \{y \mid y \to x\}$$
$$x^{\bullet} = \{y \mid x \to y\}$$

Definition: If N is the net $\langle A,B,C\rangle$, then R is a subnet of N, written $R \subseteq N$, iff $R = \langle A', B', C'\rangle$ where,

$$A' \subseteq A$$
$$B' \subseteq B$$
$$C' \subseteq C \cap (A' \times B' \cup B' \times A')$$

Property 2.1: A subnet is a net.

Notation: If R is a subnet of the net $\langle A,B,C\rangle$ and $R = \langle A', B', C'\rangle$, then,

for $X \subseteq A \cup B$: $X_R = X \cap (A' \cup B')$

for $Y \subseteq C$: $Y_R = Y \cap C'$

Q_R is the restriction of Q to R.

2.2. Petri Nets:

The rules governing the behavior of a system are expressed in terms of a Petri net.

Definition: A Petri net is a net <S,E,F> where,

S is a finite nonempty set of states
E is a finite nonempty set of events
F is the flow relation

If $s \in S$, then the elements of ${}^\bullet s$ are called the input events of s, and the elements of $s^\bullet$ the output events of s. If $e \in E$, then the members of ${}^\bullet e$ are called the input states, or preconditions, of e, and the members of $e^\bullet$ the output states, or postconditions of e.

Definition: An initialized Petri net is a quadruple <S, E, F, I> where,

<S, E, F> is a Petri net
$I \subseteq S$ is the set of initial conditions

Definition: A state graph (state machine) is a Petri net <S,E,F> in which,

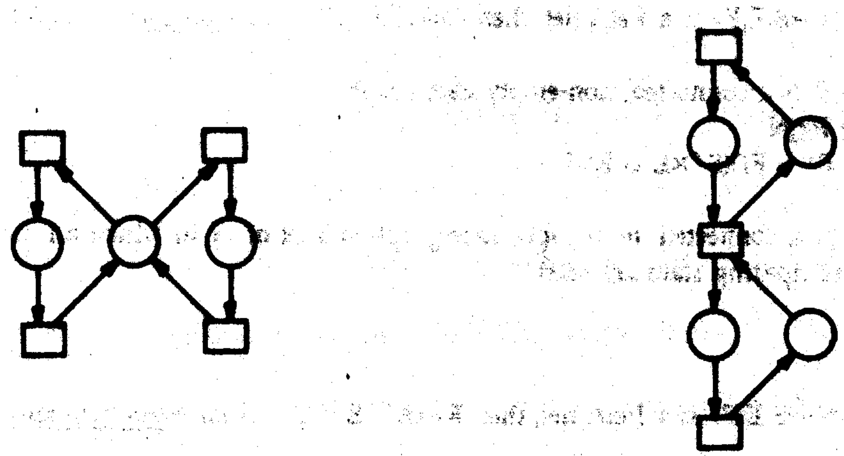
$\forall e \in E$: $|{}^\bullet e| = |e^\bullet| = 1$

'Each event has exactly one precondition and one postcondition.

Definition: An event graph (marked graph) is a Petri net <S,E,F> in which,

$\forall s \in S$: $|{}^\bullet s| = |s^\bullet| = 1$

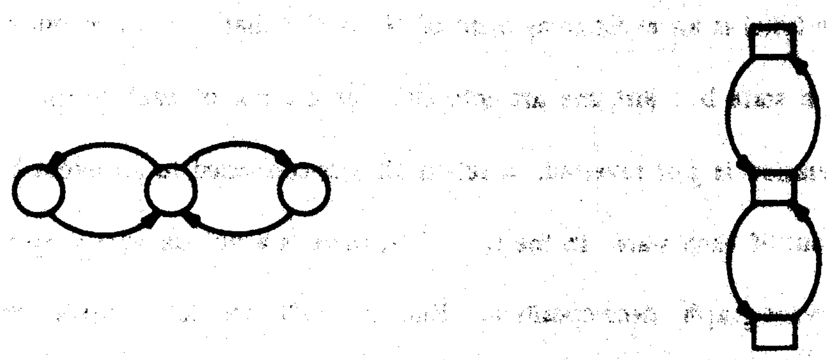
'Each state has exactly one input event and one output event.'



(a) A State Graph (b) An Event Graph

Figure 2.1

Since in a state graph and an event graph one type of node has exactly one incident arc and one emergent arc, it becomes superfluous in the graphical representation to explicitly show that type of node. We therefore adopt an abbreviated representation in which only the states in a state graph and only the events in an event graph are explicitly shown. Thus, the two nets in Figure 2.1 are now drawn as in Figure 2.2. It should be noted that this practice in no way affects the <u>formal</u> representation of state graphs and event graphs.



(a) A State Graph (b) An Event Graph

Figure 2.2 Abbreviated Representation

Definition: If N=<S,E,F> is a Petri net, then R=<S′,E′,F′> is a <u>state component</u> of N iff,

(a) R is a connected, non-empty state graph
(b) $R \subseteq N$
(c) $F' = F \cap (S' \times E \cup E \times S')$

'R is a connected, non-empty state-graph subnet of N in which all arcs connected to a participating state are used.'

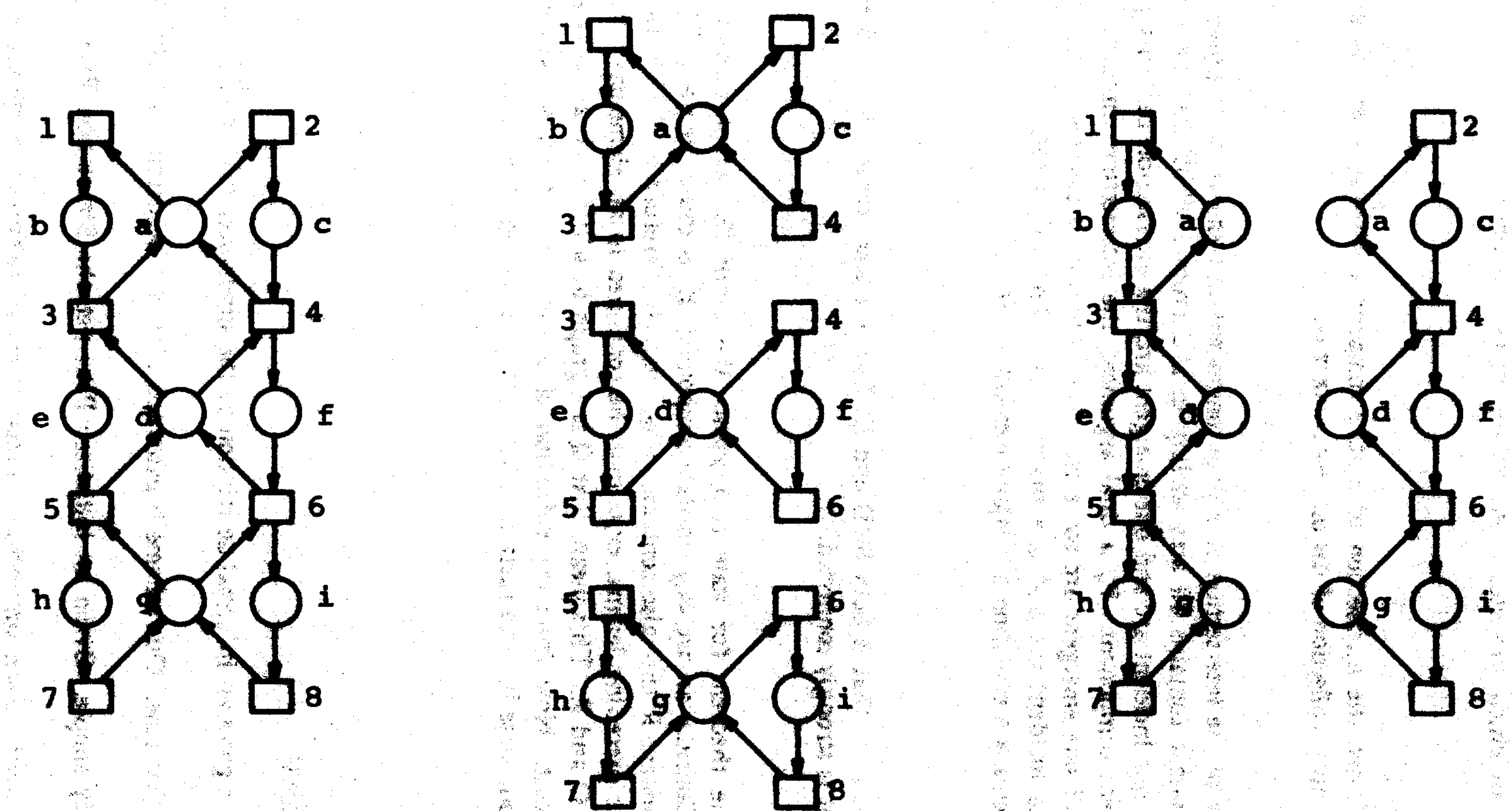
Definition: If N=<S,E,F> is a Petri net, then R=<A′, B′, C′> is an <u>event component</u> of N iff,

(a) R is a connected, non-empty event graph
(b) $R \subseteq N$
(c) $F' = F \cap (S \times E' \cup E' \times S)$

'R is a connected, non-empty event-graph subnet of N in which all arcs connected to a participating event are used."

Definition: A Petri net is said to be <u>state-graph decomposable</u> (SGD) iff each element (and thus each arc) is contained in at least one state component. A Petri net is said to be <u>event-graph decomposable</u> (EGD) iff each element (and thus each arc) is contained in at least one event component.

In Figure 2.3, we show a state-graph decomposition and an event-graph decomposition for the Petri net N. Each of the two nets in Figure 2.3(b) is a state component of N, and each of the two nets in Figure 2.3(c) is an event component of N. Notice that each state component selects all arcs connected to a state but just one arc into and one arc out of each event. With an event component the situation is just reversed: it selects all arcs connected to an event but just one arc into and one arc out of each state. In the case of N, there is a unique state-graph decomposition and a unique event-graph decomposition. But, as we'll see later, there may be several decompositions of each type due to overlap of components.



(a) N

(b) A State-Graph Decomposition of N

(c) An Event-Graph Decomposition of N

Figure 2.3

Definition: A <u>circuit</u> is a (directed) path whose two endpoints are the same. An <u>elementary circuit</u> is a circuit in which no vertex is encountered more than once.

Lemma 2.1: In a Petri net, the intersection of a state component with an event component is a (possibly empty) collection of disjoint elementary circuits.

Proof: Let s be a state in the intersection of a state component and an event component. Because the state component contains all arcs leading to and from s while the event component chooses exactly one incident arc and one emergent arc, the intersection contains just the two arcs of the event component. By a dual argument, the intersection contains exactly one incident arc and one emergent arc for each event. Thus, each element in the intersection has exactly one incident arc and one emergent arc. The only finite structure satisfying these requirements is a collection of disjoint elementary circuits. □

Theorem 2.1: In a Petri net that is both SGD and EGD, every state component and every event component is strongly connected.

Proof: We prove the theorem for state components, the proof for event components being symmetrical. Consider an arc $<x,y>$ in state component Q. Because the set is EGD, there is an event component R that also contains $<x,y>$. $<x,y>$ is therefore in the intersection of Q and R. By the preceding lemma, $<x,y>$ must be contained in an elementary circuit in the intersection of Q and R. This means that Q is a graph in which each arc is covered by an elementary circuit. Now from the definition of a state component, we know that Q is connected. From these last two facts, it follows that Q is strongly connected. □

2.3. <u>The Simulation Rule:</u>

In Section 1.2, we gave an informal description of the simulation rule for Petri nets. In this section, we formalize that rule. Our scheme is patterned after the 'occurrence graphs' of Holt [10]. The basic idea is very simple. Given an initialized Petri net, the Simulation Rule generates all possible finite 'simulations'. Each simulation expresses a causal relationship among a set of state 'holdings' and event 'occurrences'. The causal relationship reflects the pattern of 'terminations' and 'initiations' of holdings by occurrences.

A simulation is represented as a net $\langle H, O, C\rangle$ in which H is the set of holdings, O is the set of occurrences, and C is the causality relation. In order to distinguish between repeated instances of the same element, an instance (either a holding or an occurrence) is represented as an ordered pair $\langle x,n\rangle$ where x is an element - the 'instance type' - and n is a positive integer - the 'instance number'. The Simulation Rule is defined recursively. The initial simulation is a collection of isolated holdings of the form $\langle s,1\rangle$ where s is an initial condition. In a simulation, the set of unterminated holdings is referred to as the 'front boundary' of the simulation. When there exists a set of holdings in the front boundary of an existing simulation consisting of one holding for each precondition of Event e, then a new simulation can be generated. The new simulation contains one new occurrence for Event e and one new holding for each of e's postconditions. The new occurrence of e 'terminates' the previously unterminatd holdings of e's preconditions and 'initiates' the new holdings of e's postconditions.

To create new instances, we employ an auxiliary function.

Definition: $\psi(x,Q) = \{\langle x, |Q \cap (\{x\} \times N)|+1\rangle\}$

$\psi(x,Q)$ creates a set consisting of a new instance of Element x. The instance number is one greater than the number of instances of x in Q. As a result, instance numbers for an element are assigned in numerical order beginning with 1.

We're now ready for the formal simulation rule.

Definition (The Simulation Rule): If Z is the initialized Petri net <S, E, F, I>, then,

(1) <I×{1},φ,φ> is a __simulation__ of Z.

(2) If T is the existing simulation <H, O, C> and if A is a set of 'holdings' in the 'front boundary' of T consisting of one 'holding' for each precondition of Event *e*, then,

<H ∪ $\eta(e^*,H)$,
O ∪ $\eta(e,O)$,
C ∪ A×$\eta(e,O)$ ∪ $\eta(e,O)$×$\eta(e^*,H)$>[†] is a __simulation__ of Z.

(3) The only __simulations__ of Z are those given by (1) and (2).

Step (2) is illustrated in Figure 2.4.

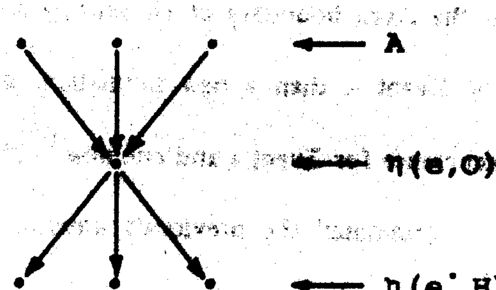


Figure 2.4 Extending a Simulation

Definition: If T is the simulation <H,O,C>, then,

H is the set of __holdings__ of T
O is the set of __occurrences__ of T
C is the __causality relation__ of T

[†] We're using a notational convention here. If the domain of the function *f* is Q, and if the elements in the range of *f* are sets, then for $X \subseteq Q$,

$$f(X) = \bigcup_{x \in X} f(x)$$

Thus, $\eta(e^*, H) = \bigcup_{s \in e^*} \eta(s, H)$.

The ordered pairs in C are the elementary causal connections of T. Occurrence q is said to terminate (initiate) Holding h iff there is an elementary causal connection leaning from h to q (q to h). The set of unterminated (uninitiated) holdings is called the front (back) boundary of T.

In the graphical representation of a simulation, the vertices are drawn as points. The simulation in Figure 2.5 is one of those generated from the net in Figure 2.3(a) with States a,d, and g designated as initial conditions. Because instance numbers are redundant in a graphical representation and because they might be confused with event names, they're usually omitted. The abbreviated form of the simulation in Figure 2.5 is shown in Figure 2.6. Note that this practice has no effect on the formal representation of a simulation.

The following four properties follow immediately from the simulation rule.

Property 2.2: A simulation is a net.

Property 2.3: An occurrence of an event terminates one holding of each precondition of that event and initiates one holding of each postcondition.

Property 2.4: A holding is initiated by no more than one occurrence and is terminated by no more than one occurrence.

Property 2.5: A simulation is circuit-free.

Definition: Consider the result of taking the transitive and reflexive closures of a simulation. The new structure is transitive, reflexive, and - because of Property 2.5 - antisymmetric. In short, it is a partial order. We write $x \leq y$ to indicate that x is related to y by this partial order, and $x<y$ to indicate that $x \leq y$ but $x \neq y$. We adopt the following terminology,

$x=y$ - x and y are coincident

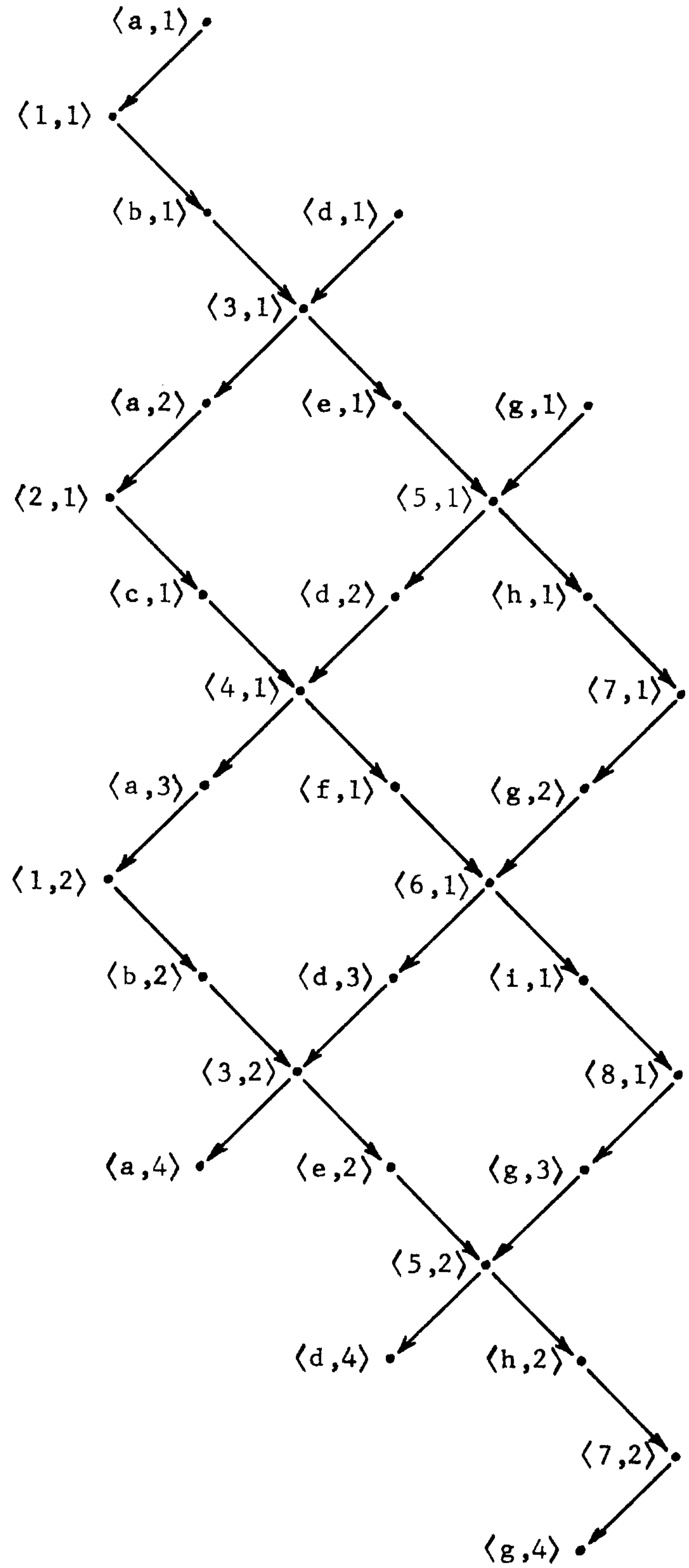


Figure 2.5 A Simulation

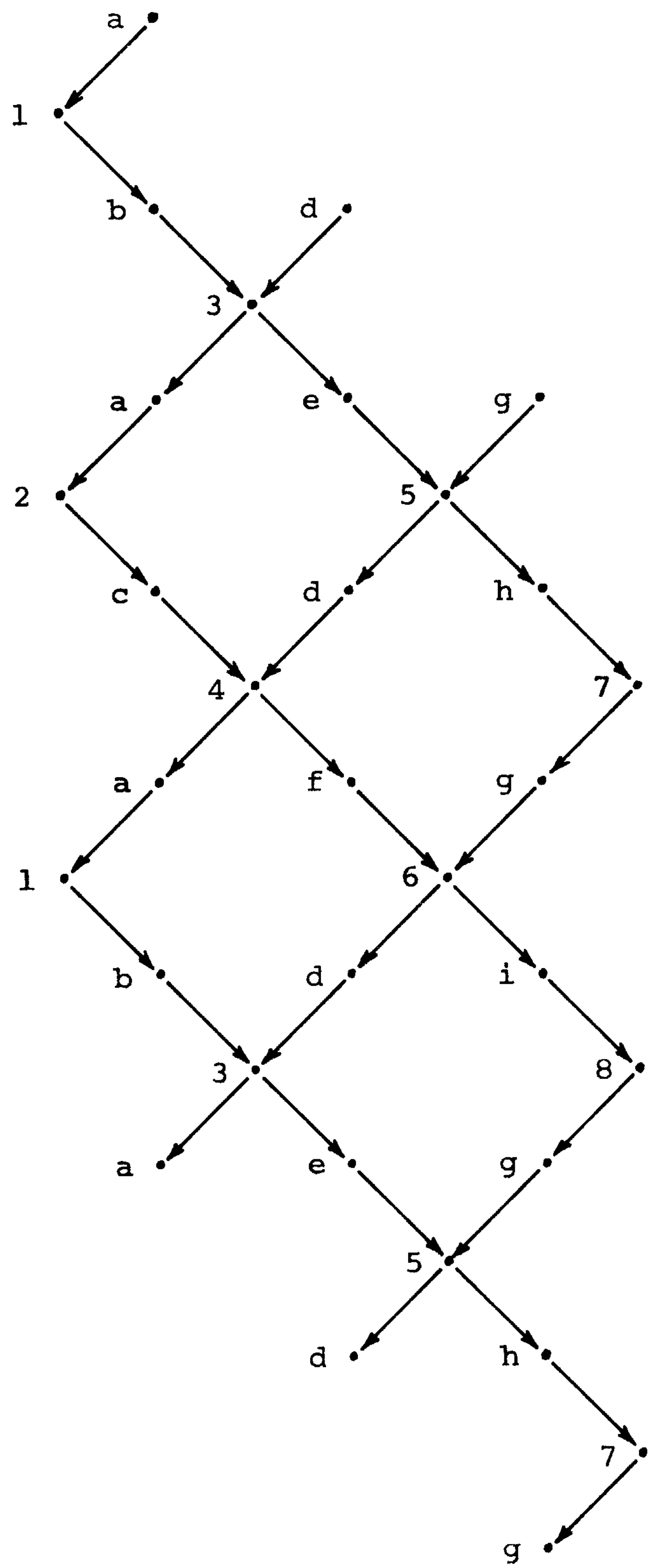


Figure 2.6 A Simulation (Abbreviated)

$x \leq y$ - *x* <u>precedes</u> *y*
y <u>follows</u> *x*

$x \not\leq y \wedge y \not\leq x$ - *x* and *y* are <u>concurrent</u>

(Note that by our convention an instance precedes itself and follows itself. This is not normal usage, but it's more convenient for our purposes. Notice that concurrency is nothing more than the absence of ordering.)

The next property follows directly from the Simulation Rule.

Property 2.6: If the instances of Element x are totally ordered in Simulation T, and if $\langle x,m \rangle$ and $\langle x,n \rangle$ are instances in T, then $\langle x,n \rangle$ is the next instance of x following $\langle x,m \rangle$ iff $n=m+1$.

We now establish some basic structural relationships between Petri nets and their simulations.

Definition: For a directed graph G, $\Pi(G)$ denotes the paths of G. If G is a simulation, then each path represents a <u>causal connection</u>.

Notation: If q is the instance $\langle x,n \rangle$, then $\hat{q} = x$. This notation is extended to a sequence of instances in the obvious manner. If σ is either an instance or a path in a simulation, then $\hat{\sigma}$ is called its <u>image</u>.

Theorem 2.2: If T is a simulation of the initialized Petri net $\langle N,I \rangle$, then,

$$\sigma \in \Pi(T) \Rightarrow \hat{\sigma} \in \Pi(N)$$

'The image of a path in T is a path in N.'

Proof: Every arc in T is either of the form <holding, occurrence> or <occurrence, holding>. This together with Property 2.3 leads to the desired result. □

Definition: A <u>strand</u> of a simulation is a path originating in the back boundary of the simulation and terminating in the front boundary.

Definiton: If $T=\langle H,O,C\rangle$ is a simulation of the initialized Petri net $\langle S,E,F,I\rangle$ and if R is a subnet of $\langle S,E,F\rangle$, then

for $A \subseteq H \cup O$: $A_R = \{q \in A \mid \hat{q} \in (S_R \cup E_R)\}$

'A_R contains those instances in A that have images within R.'

for $B \subseteq C$: $B_R = \{\langle p,q\rangle \in B \mid \langle \hat{p}, \hat{q}\rangle \in F_R\}$

'B_R contains those arcs in B that have images within R.'

$T_R = \langle H_R, O_R, C_R\rangle$

Property 2.7: If T is a simulation of the initialized Petri net $\langle N,I\rangle$ and R is a subnet of N, then T_R is a subnet of T.

Property 2.8: If $T=\langle H,O,C\rangle$ is a simulation of the initialized Petri net $\langle S,E,F,I\rangle$ and R is a subnet of $\langle S,E,F\rangle$, then,

$$F_R = F \cap (S_R \times E_R \cup E_R \times S_R) \Rightarrow C_R = C \cap (H_R \times O_R \cup O_R \times H_R)$$

'If F_R consists of all arcs (in F) connecting two elements of R, then C_R consists of all arcs (in C) connecting two instances of T_R.'

Property 2.9: If $T = \langle H,O,C\rangle$ is a simulation of the initialized Petri net $\langle N,I\rangle$ and R is either a state component or an event component of N, then,

$$C_R = C \cap (H_R \times O_R \cup O_R \times H_R)$$

'C_R consists of all arcs in C connecting two instances of $H_R \cup O_R$.'

Theorem 2.3: If T is a simulation of the initialized Petri net <N,I> and R is a state component of N, then T_R consists of $|I_R|$ disjoint strands.†

Proof: If T is the initial simulation of <N,I>, then T_R consists of the holdings in $I_R \times \{1\}$. They form $|I_R|$ disjoint strands.

Now suppose that $\langle H_1, O_1, C_1 \rangle$ is a simulation of <N, I> for which the theorem is satisfied, and that $\langle H_2, O_2, C_2 \rangle$ is derived from $\langle H_1, O_1, C_1 \rangle$ through a single application of Step (2) of the Simulation Rule. Thus, there exists an event e and a set of holdings A in the front boundary of $\langle H_1, O_1, C_1 \rangle$ such that,

$$H_2 = H_1 \cup \psi(e^{\bullet},H_1)$$
$$O_2 = O_1 \cup \psi(e,O_1)$$
$$C_2 = C_1 \cup A\times\psi(e,O_1) \cup \psi(e,O_1)\times\psi(e^{\bullet},H_1)$$

From Property 2.9 we have,

$$(C_2)_R = (C_1)_R \cup A_R\times(\psi(e,O_1))_R \cup (\psi(e,O_1))_R\times(\psi(e^{\bullet},H_1))_R$$

There are now two possibilities: e is contained in R, or e is not contained in R. In the first case, $|A_R| = |(\psi(e,O_1))_R| = |(\psi(e^{\bullet},H_1))_R| = 1$. And in the second $|A_R| = |(\psi(e,O_1))_R| = |(\psi(e^{\bullet},H_1))_R| = 0$. Either way, the theorem is satisfied for $\langle H_2, O_2, C_2 \rangle$. □

In the net of Figure 2.7, there is a '2-token' state component consisting of those states, events, and arcs that lie on the outside ring. According to Theorem 2.3, there should be two strands associated with that state component in each simulation of the net. These strands are shown for the simulation in Figure 2.8. Notice that the net in Figure 2.7 contains another 2-token state component - the inside ring - and four 1-token state components. The reader may verify that the simulation of Figure 2.8 contains the appropriate number of strands for each of these.

† $|I_R|$ is the number of 'tokens' on R.

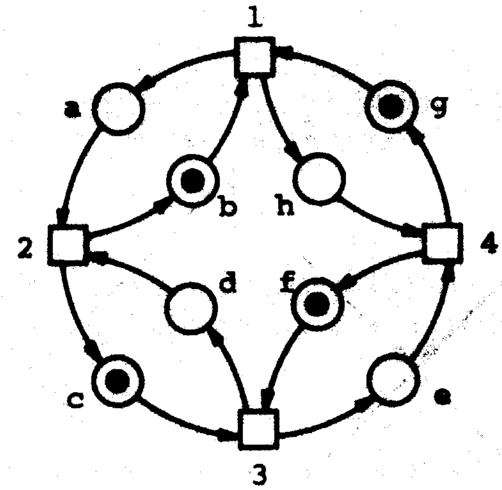


Figure 2.7

Corollary 2.1 If T is a simulation of the initialized Petri Net <N,I>, and R is a 1-token state component of N, then T_R consists of a single strand.

Corollary 2.2 If <H,O,C> is a simulation of the initialized Petri Net <N,I>, and R is a 1-token state component of N, then the instances in $H_R \cup O_R$ are totally ordered.

Corollary 2.3 If T is a simulation of an initialized Petri net covered by 1-token state components, then within T the instances of each element are totally ordered.

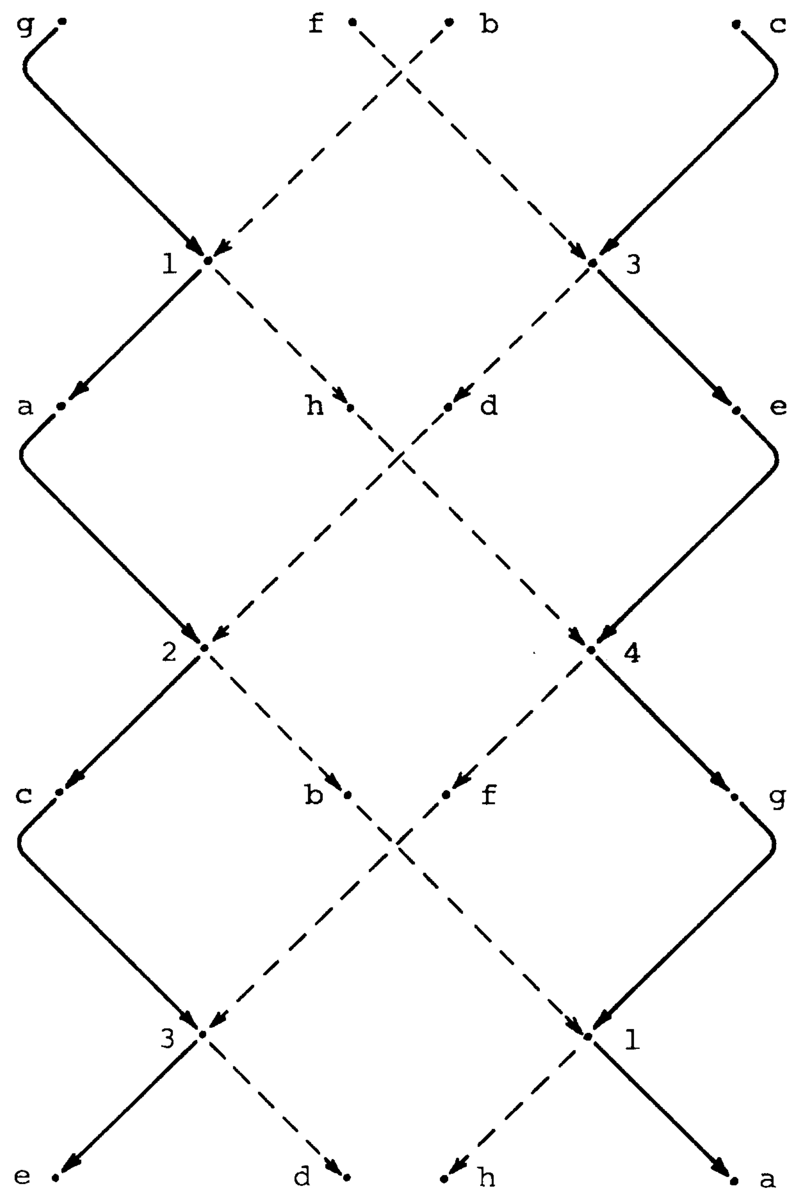


Figure 2.8 Strands of a State Component

CHAPTER 3

SYSTEMS

3.1. Assumptions:

Like any theory, the one presented here is based on certain assumptions. The major ones are the following:

(1) Associated with each system is a set of states (conditions) and events - the system elements.

(2) The logical aspects of system behavior can be completely expressed in terms of the possible patterns of state holdings and event occurrences.

(3) Petri nets are an appropriate tool for representing the logical constraints that a system places on the holdings and occurrences of its elements.

(4) A system may be decomposed into sequential components, and the alternativeness relation induced by these components partitions the system elements into alternative classes.

(5) Every system element is part of at least one (potential) steady-state pattern of behavior.

Assumptions (1) and (2) represent an attempt to find a common ground for describing the myriad facets of system behavior. The notions of state, event, holding, and occurrence appear to be general enough to encompass everything that one might consider to be 'logical behavior'. Note that we are specifically excluding those aspects of reality not explicable in logical terms - for example, human emotions.

Because we're dealing with finite systems, there must be a finite way of characterizing the constraints that a system places on the holdings and occurrences of its elements. Experience with Petri nets has shown them to be ideally suited for characterizing such constraints. This is the basis for Assumption (3).

The most natural way of introducing the notion of alternativeness is by assuming the existence of sequential components. Each such component induces a natural partition of its elements into alternative classes. (To be explained below). By assuming that alternative classes from different components do not partially overlap, we get a partition of the entire set of system elements. This is the meaning of Assumption (4).

Assumption (5) introduces the notion of steady-state behavior. Steady-state has been an important concept in many different disciplines, but these disciplines have been based on <u>continuous</u> models while ours is based on a <u>discrete</u> model. The fact that steady-state appears in conjunction with a discrete model should not be too surprising though. After all, we're trying to describe the same reality whether we use a continuous model or a discrete model.

The five assumptions are embodied in our definition of a system. This definition incorporates five axioms. Although these axioms exclude many interesting and meaningful nets, it has so far been possible to convert such nets into the framework of our theory. Much more experience is needed to fully understand the implications of these axioms and the assumptions they embody.

3.2. <u>A System:</u>

In the preceding chapter we distinguished between a 'Petri net' and an 'initialized Petri net'. We do the same for systems. This section and the next three are concerned with the purely structural properties of a system. Section 3.[illegible] is concerned with the behavioral properties of an initialized system.

We begin by defining the <u>system</u> Σ. It consists of a Petri net together with a set of subsets of states and a set of subsets of events. The subsets of states are used to generate a covering of state components - the 'parts' of the system. The subsets of events are used to generate a covering of event components - the 'modes' of the system. We could, of course, have included the parts and modes themselves in the definition of a system, but there would have been a great deal of redundant information. We're taking advantage of the fact that a state component is uniquely identified by its states, while an event component is uniquely identified by its events. Note that for a given Petri net, there may be several coverings of both state components and event components. The constructs of the theory will, in general, depend upon which coverings are selected, but the implications of this are not fully understood.

Definition: $\mathcal{S} = \langle N, D_{\mathcal{P}}, D_{\mathcal{M}} \rangle$ where,

> $N = \langle S, E, F \rangle$ is a Petri net - the <u>system net</u>
> $D_{\mathcal{P}} \subseteq \mathcal{P}(S)$ is the <u>part decomposition</u>[†]
> $D_{\mathcal{M}} \subseteq \mathcal{P}(E)$ is the <u>mode decomposition</u>
>
> X denotes $S \cup E$, the <u>system elements</u>.

Axiom 1: N is connected.

This axiom merely prevents a system from having several disconnected components. This is not a real limitation since in such cases each connected component can be treated as a separate system.

Axiom 2: $S = \cup D_{\mathcal{P}}$

$$\wedge$$

$$\forall A \in D_{\mathcal{P}}: \langle A, \text{`}A \cup A\text{`}, F \cap (A \times E \cup E \times A) \rangle \text{ is a connected, non-empty state graph}$$

'The sets of states in $D_{\mathcal{P}}$ generate a covering of state components.'

[†] $\mathcal{P}(A)$ denotes the <u>power set</u> of A - i.e., the set of all subsets of A.

Definition: The state components generated by the sets in $D_{\mathcal{P}}$ are called the <u>parts</u> of $\mathcal{S}$. The set of parts is denoted by $\mathcal{P}$.

When the system $\mathcal{S}$ is initialized, we will require that each part be assigned exactly one initial condition. Thus, the parts will become 1-token state components and will be associated with strictly sequential behavior.

Axiom 3: $E = \cup D_{\mathfrak{M}}$

$\wedge$

$\forall B \in D_{\mathfrak{M}}$: $\langle {}^{\bullet}B \cup B^{\bullet}, B, F \cap (S \times B \cup B \times S) \rangle$ is a connected, non-empty event graph.

'The sets of events in $d_{\mathfrak{M}}$ generate a covering of event components.'

Definition: The event components generated by the sets in $D_{\mathfrak{M}}$ are called the <u>modes</u> of $\mathcal{S}$. The set of modes is denoted by $\mathfrak{M}$.

Each mode is to be associated with a steady-state pattern of behavior. The reasons for this interpretation are simple. If an event is involved in a steady-state pattern of behavior, then all of the states connected to that event are also involved. For a state that is part of a steady-state pattern the situation is different. Here, just one input event and one output event of the state are involved. So we see that steady-state behavior is naturally associated with event components. In Section 3.5, we'll strengthen the connection between steady-state behavior and those event components that comprise the modes.

Property 3.1: Every part and every mode is strongly connected.

This follows from the fact that N is both SGD and EGD. (See Theorem 2.1)

Property 3.2: N is strongly connected.

Since N is connected (Axiom 1) and covered by strongly connected components (Property 3.1), it must be strongly connected.

3.3. The Parts:

The main reason for including parts in our definition of a system is so that we can define the notion of alternativeness. We begin by assuming that concurrency and alternativeness are mutually exclusive. That is, if two system elements may either hold or occur concurrently, then they cannot be considered alternative. The most natural way of guaranteeing that two elements will never hold or occur concurrently is to require that they both be contained in a single 1-token state component - that is, a part. But we don't want to say that two elements are alternative just because they belong to the same part. For example, if the part consists of a single elementary circuit, then no two elements on the circuit can be said to be alternative. There is a situation, however, in which two elements would definitely be called alternative: when they are alternatives in a choice. Since our theory is intended to be symmetrical with respect to the forwards and backwards directions of time, we include both forwards and backwards choices. In Figure 3.1, Events e_1 and e_2 are alternative, as are Events e_3 and e_4. We carry this idea one step further by defining the 'alternative closure' of a part: If two elements in a part are alternative, then their immediate successors within the part are also alternative, as are their immediate predecessors. Thus in Figure 3.2, States s_1 and s_2, as well as States s_3 and s_4, are alternative.

Figure 3.1 Alternative Events

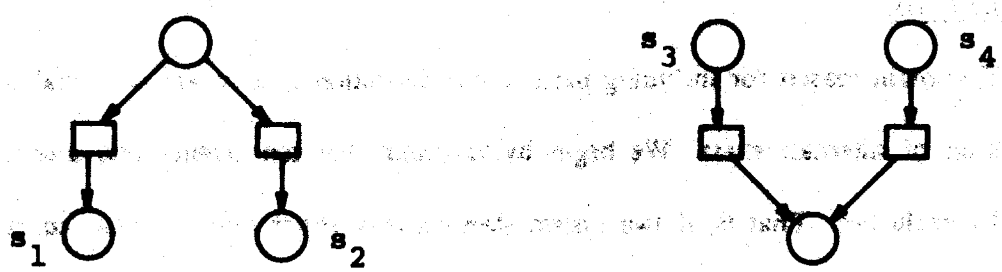


Figure 3.2 Alternative States

This idea is formalized as follows.

Definition: The <u>alternativeness relation for Part P</u> is the minimal relation $\alpha_P \subseteq (X_P)^2$ such that,

$$\forall x \in X_P:\ x \alpha_P x$$

$$\forall x_1,x_2,x_3,x_4 \in X_P:\ \ x_1 \alpha_P x_2 \wedge ((x_1 = x_3 \wedge x_2 = x_4) \vee (x_3 = x_1 \wedge x_4 = x_2)) \Rightarrow x_3 \alpha_P x_4$$

We say that x_1 and x_2 are <u>alternative</u> (in P) iff $x_1 \alpha_P x_2$ but $x_1 \neq x_2$. (We do not say that an element is alternative with itself.)

Theorem 3.1: For $P \in \mathcal{P}$, α_P is an equivalence relation on the elements of P, and,

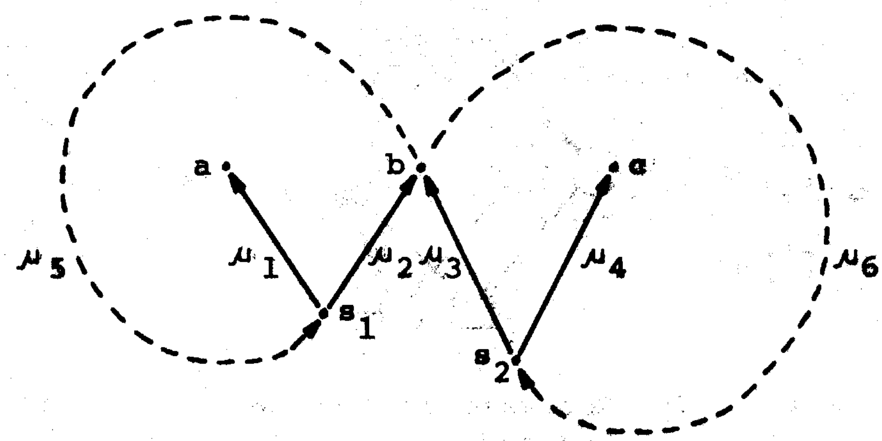
$$\forall A \in X_P/\alpha_P:\ A \subseteq S \vee A \subseteq E^{\dagger}$$

'Each equivalence class induced by α_P contains either exclusively states or exclusively events.'

---

† If $\equiv$ is an equivalence relation on the set X, then $X/\equiv$ denotes the set of equivalence classes induced by $\equiv$.

Proof: Reflexivity and symmetry follow directly from the definition of α_P. For transitivity and the second part of the theorem, we make use of the fact that two elements are related by α_P iff there is a state from (to) which there exist paths of equal length leading to (from) those two elements.[†] Thus if $a \alpha_P b$ and $b \alpha_P c$, we have the following situation in which $|\mu_1|=|\mu_2|$ and $|\mu_3|=|\mu_4|$.



Because a part is strongly connected (Property 3.1), there exist paths μ_5 and μ_6 as shown. This gives us paths of equal length leading from b to a and c - namely, $\mu_6\mu_3\mu_5\mu_1$ and $\mu_5\mu_2\mu_6\mu_4$. Thus, $a \alpha_P c$ and α_P is transitive. To see that a state and an event can never be alternative, it is only necessary to note that between any two states all paths are of even length while between a state and an event all paths are of odd length. □

Now since α_P is an equivalence relation on the elements of Part P, α_P induces a quotient net of P.

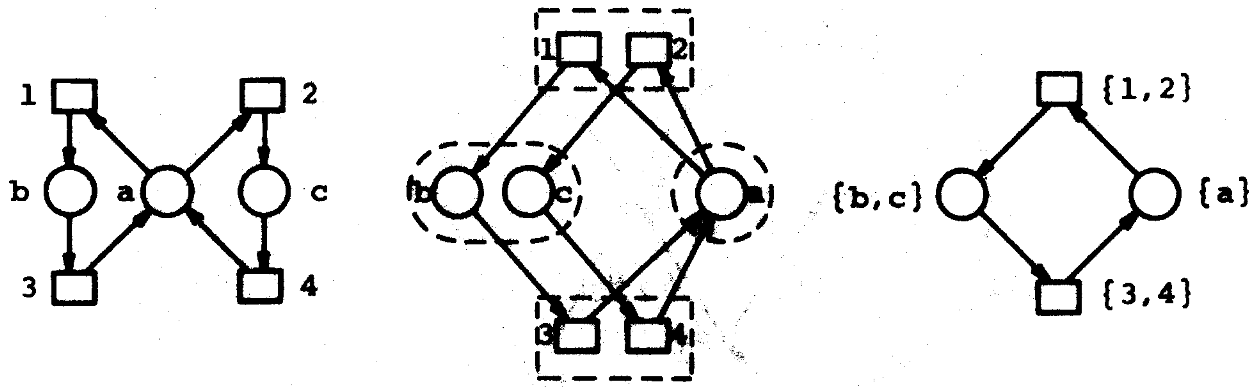
Definition: For $P \in \mathcal{P}$,

$$P^* = \langle \{[s]_{\alpha_P} \mid s \in S_P\},$$
$$\{[e]_{\alpha_P} \mid e \in E_P\},$$
$$\{\langle [x]_{\alpha_P}, [y]_{\alpha_P} \rangle \mid \langle x,y \rangle \in F_P\} \rangle$$

† This fact may be verified by the reader.

Property 3.3: P^* is a net.

The method of generating a quotient net is illustrated in Figure 3.3.



(a) Part P (b) Equivalence Classes Induced by α_P (c) P^*

Figure 3.3

Notice that the quotient net in Figure 3.3 (c) is an elementary circuit. This is always the case. What's more, the length of such a circuit is equal to the gcd (greatest common divisor) of the lengths of the elementary circuits in the corresponding part.

Definition: If G is a strongly-connected directed graph, then,

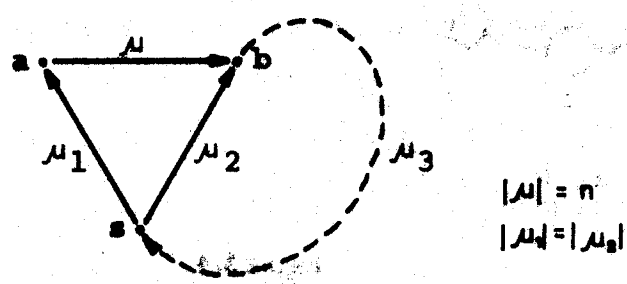
$$\gamma(G) = \gcd \{n | n \text{ is the length of an elementary circuit in N}\}$$

Theorem 3.2: For $P \in \mathcal{P}$, P^* is an elementary circuit of length $\gamma(P)$.

Proof: The definition of α_P eliminated all branching, both forwards and backwards, in P^*. Because P is strongly connected, so too is P^*. It follows that P^* is an elementary circuit.

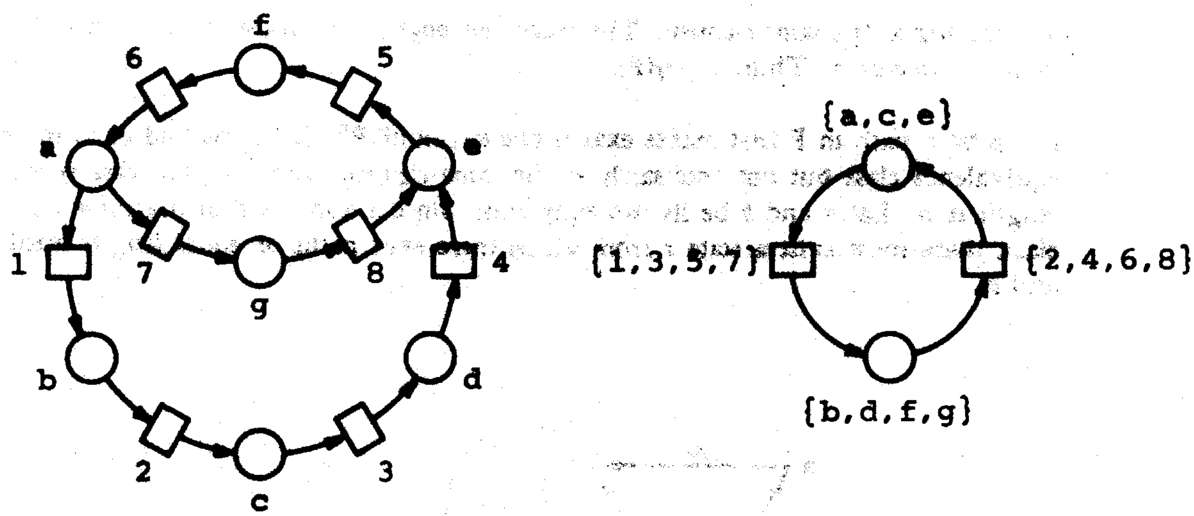
Let n be the length of the elementary circuit comprising P^*. We note that two elements belong to the same equivalence class iff the length of every path between the two is a multiple of n. Now each elementary circuit in P may be viewed as a path starting and terminating at the same element. Therefore, the length of each elementary circuit in P must be a multiple of n. Thus, $n \leq \gamma(P)$.

Let μ be a path in P that makes exactly one circuit of P^*. It begins and ends in the same equivalence class but not necessarily at the same element. Such a path clearly exists. Its length is n. Let a and b be its two endpoints. Since a and b are in the same equivalence class, there must exist a state s from which there exist paths of equal length leading to a and b.



Because P is strongly connected, there exists a path μ_3 from b to s. We now have two circuits: $\mu_3\mu_2$ and $\mu_3\mu_1\mu$. Since every circuit, elementary or otherwise, can be decomposed into elementary circuits, it follows that $\gamma(P)$ divides the lengths of all circuits in P. In particular, $\gamma(P)$ divides $|\mu_3\mu_2|$ and $|\mu_3\mu_1\mu|$. From number theory, we know that $\gamma(P)$ must also divide the difference between $|\mu_3\mu_2|$ and $|\mu_3\mu_1\mu|$. But this is just n. Therefore, $\gamma(P) \leq n$. Since we've already shown that $n \leq \gamma(P)$, we have $n = \gamma(P)$. □

The state graph in Figure 3.4(a) has two elementary circuits, one of length 8 and the other of length 12. The quotient net induced by this state graph is shown in Figure 3.4(b). It is an elementary circuit of length 4, which is the *gcd* of 8 and 12.



(a) P (b) P^*

Figure 3.4

3.4. The Control Structure:

What we've done so far is to generate a separate quotient net for each part. In order to construct a single quotient net for the entire system, we first have to look at the possible relationships between two alternative classes from two different parts. There are three possibilities: (1) the alternative classes are disjoint, (2) they partially overlap, or (3) they are identical. Since we need a partition of the system elements to construct a quotient system, we must exclude the second possibility, alternative classes that partially overlap. This is accomplished with the following axiom.

Axiom 4: $\forall P_1, P_2 \in \mathcal{P}$: $\forall q \in X_{P_1}/\alpha_{P_1}$: $\forall r \in X_{P_2}/\alpha_{P_2}$:

$$q \cap r = \phi \ \vee \ q = r$$

'Two alternative classes are either disjoint or identical.'

Figure 3.5 illustrates the type of situation that is prohibited by this axiom. Part P_1 generates an alternative class containing the events e_1 and e_2. Part P_2 generates an alternative class containing the single event e_1. The two alternative classes overlap but are not identical.

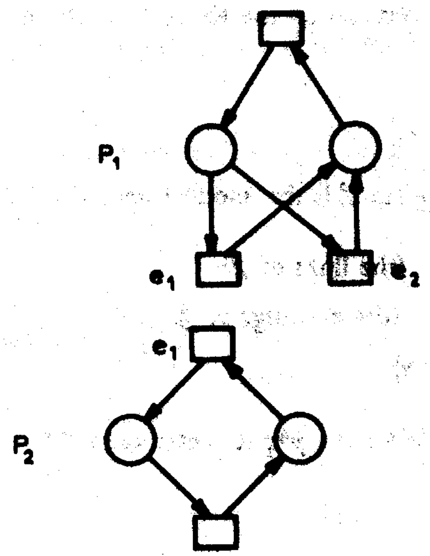


Figure 3.5 Partially Overlapping Alternative Classes

As a result of Axiom 4, we now have an equivalence relation on the set of system elements.

Definition: $\alpha = \bigcup_{P \in \mathcal{P}} \alpha_P$

α is the <u>alternativeness relation</u> for $\mathcal{S}$.

Property 3.4 $\propto$ is an equivalence relation on x, and,

$$\forall A \in X/\propto: \quad A \subseteq S \quad \vee \quad A \subseteq E$$

'Each equivalence class induced by $\propto$ contains either exclusively states or exclusively events.'

Definition: The equivalence classes induced by $\propto$ are called alternative classes. An alternative class containing all states is called a link. An alternative class containing all events is called a meeting.

Since $\propto$ is an equivalence relation on the elements of the net N, $\propto$ induces a quotient net of N.

Definition: The control structure for $\mathcal{B}$ is the quotient net $N^* = \langle S^*, E^*, F^* \rangle$ where,

$S^* = \{[s]_\propto \mid s \in S\}$ (the links of $\mathcal{B}$)

$E^* = \{[e]_\propto \mid e \in E\}$ (the meetings of $\mathcal{B}$)

$F^* = \{\langle [x]_\propto, [y]_\propto \rangle \mid x \cdot y\}$

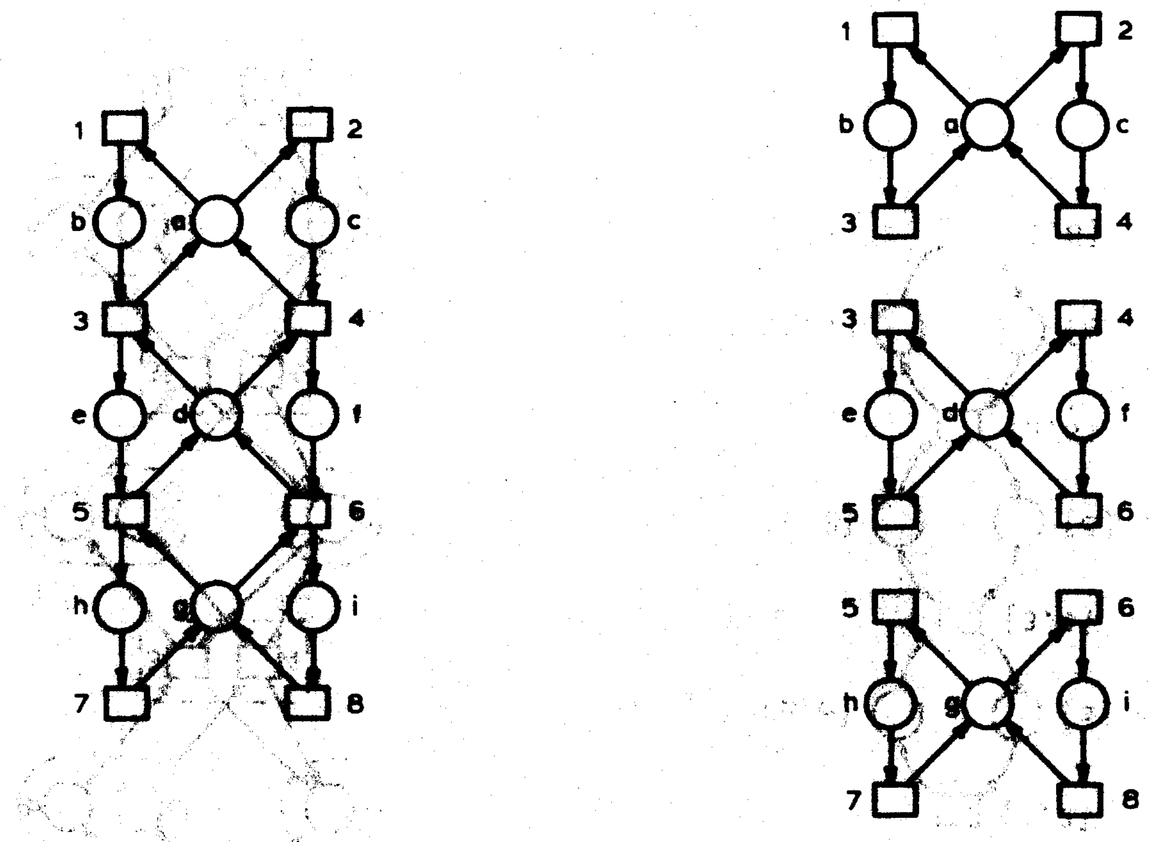
X^* denotes $S^* \cup E^*$. We write $p * q$ to mean $\langle p,q \rangle \in F^*$. (Recall that $x \cdot y$ means $\langle x,y \rangle \in F$.) For $p \in X^*$,

$$p^* = \{q \mid p * q\}$$

$$^*p = \{q \mid q * p\}$$

Property 3.5 N^* is a net. (N^* will be interpreted as a Petri net.)

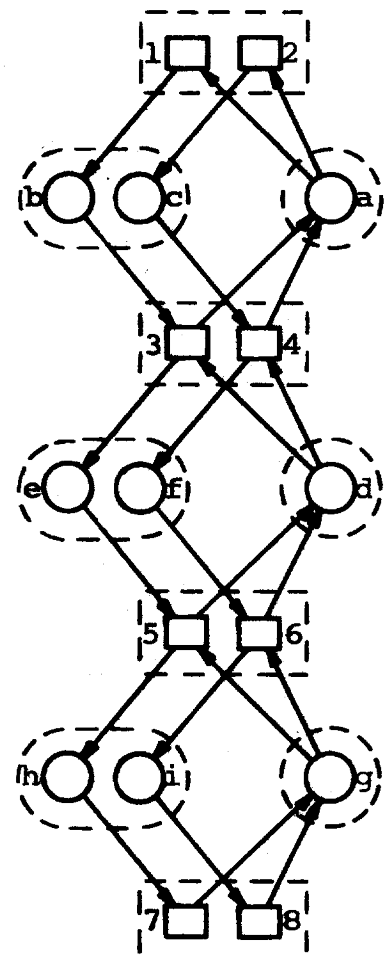
The steps involved in generating a control structure are illustrated in Figure 3.6. Notice that the control structure may be viewed as an interconnection of the elementary circuits generated by the parts. From this we have the following.



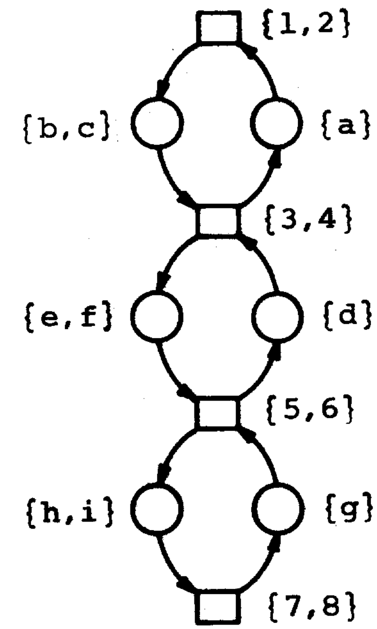
(a) The System Net (N)

(b) The Parts

Figure 3.6 Generating a Control Structure



(c) **The Alternative Classes**



(d) The Control Structure (N^*)

Figure 3.6 (Continued)

Property 3.6 N* is strongly connected.

And there is something else that we can say about N*.

Theorem 3.3: N* is an event graph.

Proof: Each link *l* belongs to at least one part. Let P be such a part. Then *l* is contained in the elementary circuit generated by P. Within that circuit, *l* has a unique input meeting and a unique output meeting. Because P is a state component, those two meetings contain all the events that are adjacent to the states in *l*. Therefore, there can be no other meetings connected to *l*.

Since it is our custom not to explicitly show the states in an event graph, the links in a control structure will be drawn as . . . 'links'. Thus, the control structure in Figure 3.6(d) will be depicted as in Figure 3.7.

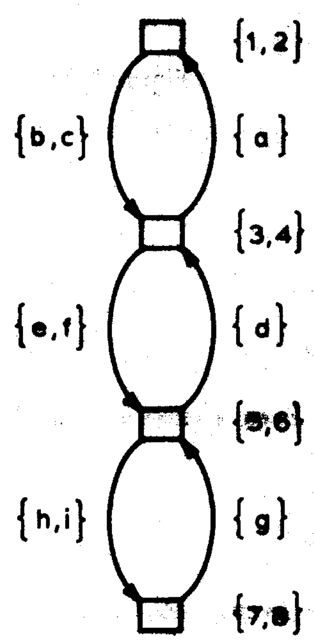


Figure 3.7 A Control Structure (Abbreviated Representation)

Between the system net N and the control structure. N^*, there exists an important structural relationship, one that will be used in relating system behavior to control behavior. The relationship is illustrated in Figure 3.8 and is expressed by the following theorem.

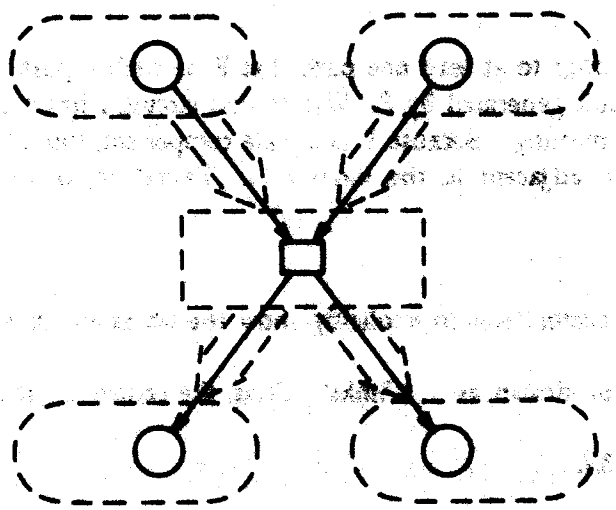


Figure 3.8

Theorem 3.4 $\forall e \in E$: $\forall l \in S^*$:

$$l \blacklozenge [e]_{oc} \Leftrightarrow |l \cap {}^{\circ}e| = 1 \quad \text{(a)}$$

$$l \blacklozenge [e]_{oc} \Leftrightarrow |l \cap {}^{\circ}e| = 0 \quad \text{(b)}$$

'The preconditions of Event e select one state from each input link of $[e]_{oc}$.'

$$[e]_{oc} \blacklozenge l \Leftrightarrow |l \cap e^{\circ}| = 1 \quad \text{(c)}$$

$$[e]_{oc} \blacklozenge l \Leftrightarrow |l \cap e^{\circ}| = 0 \quad \text{(d)}$$

'The postconditions of Event e select one state from each output link of $[e]_{oc}$.'

Proof: The theorem follows from these observations,

(1) A meeting m and a link l are connected in N^* iff there is a part P such that m and l are connected in P^*.

(2) m and l are connected in P^* iff each event in m is connected to exactly one state in l.

(3) If m and l are not connected in N^*, then no state in l is connected to an event in m. □

3.5. The Modes:

The modes are event components of the system net N. The event components of N have an interesting property.

Lemma 3.1: If M is an event component of N, then,

$$\forall q,r \in X^*: \; |X_M \cap q| = |X_M \cap r|$$

'An event component intersects all alternative classes the same number of times.'

Proof: An event component selects all arcs into and out of an event and one arc into and one arc out of a state. From Theorem 3.4, we then have,

$$\forall p_1,p_2 \in X^*: \; p_1 {}^{*} p_2 \Rightarrow |p_1 \cap X_M| = |p_2 \cap X_M|$$

This fact, together with the connectivity of N^*, produces the desired result. □

Modes are the structures that correspond to steady-state patterns of behavior. We shall take the term 'steady-state' to mean that a mode intersects an alternative class in no more than one element.

Axiom 5: $\forall M \in \mathfrak{M}: \quad \forall \gamma \in X^{*}: \quad |X_M \cap \gamma| \leq 1$

'A mode and an alternative class intersect in no more than one element.'

In figure 3.6 we presented a system net together with a set of parts. When these are combined with the modes in Figure 3.9, we get a complete system. The reader may verify that Axiom 5 is satisfied.

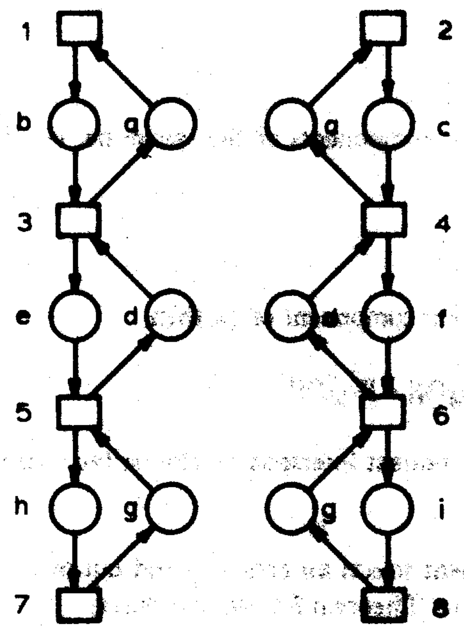


Figure 3.9 Modes

From Lemma 3.1 and Axiom 5 it follows that a mode either does not intersect any alternative class at all, or intersects each alternative class exactly once. But the first case cannot be, since it would imply a mode with no elements.

Theorem 3.5: $\forall M \in \mathfrak{M}$: $\forall \varphi \in X^*$: $|X_M \cap \varphi| = 1$

'A mode intersects each alternative class exactly once.'

Corollary 3.1: $\forall \varphi \in X^*$: $|\varphi| \leq |\mathfrak{M}|$

'The size of an alternative class cannot be greater than the number of modes.'

Theorem 3.5 together with the next theorem establish an important relationship between each mode and the control structure.

Theorem 3.6: $\forall M \in \mathfrak{M}$: $\forall x,y \in X_M$:

$$<x,y> \in F_M \Leftrightarrow [x]_\alpha \diamond [y]_\alpha$$

'There is an arc in M connecting x to y iff there is an arc in N^* connecting $[x]_\alpha$ to $[y]_\alpha$.'

Proof: $\Rightarrow$ Definition of N^* as a quotient net.

$\Leftarrow$ Assume $[x]_\alpha \diamond [y]_\alpha$. Either x or y (but not both) is an event. Assume it's x. By Theorem 3.4 and the definition of a mode as an event component, there exists an element z in $X_M \cap [y]_\alpha$ such that $<x,z> \in F_M$. But by Theorem 3.5 we know that there is only one element in $X_M \cap [y]_\alpha$. Therefore, $y = z$ and $<x,y> \in F_M$. □

Corollary 3.2: $\forall M \in \mathfrak{M}$: M is isomorphic to N^*

'Each mode is isomorphic to the control structure.'

We now have a nice visual interpretation for the class of nets produced by Axioms 1 through 5. Each net in this class can be viewed as an interconnection of isomorphic event graphs. If we imagine the elements in each alternative class to be vertically in line, then each mode will be roughly horizontal (see Figure 3.10). In the top view, alternatives are indistinguishable. So too are the modes. Their projection forms the control structure. In the side view, we can distinguish between the alternatives in an alternative class, and we can identify the individual modes.

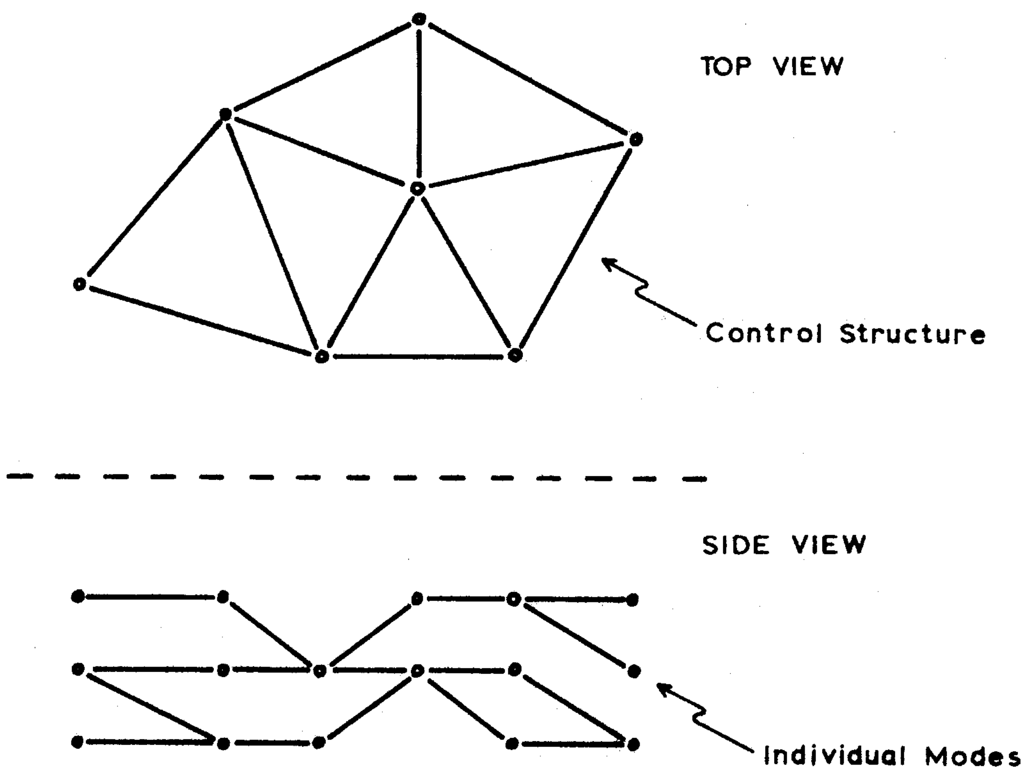


Figure 3.10 Views of a System Net

From the structure of the modes, we can say something about the structure of the parts.

Theorem 3.7: $\forall P \in \mathcal{P}$: $\forall M \in \mathfrak{M}$: $P \cap M$ is an elementary circuit of length $\gamma(P)$.

Proof: P^* is an elementary circuit of length $\gamma(P)$. Since M is isomorphic to N^*, M contains an elementary circuit of length $\gamma(P)$ whose image is P^*. This circuit is also contained in P. □

Corollary 3.3: $\forall P \in \mathcal{P}$: P is covered by elementary circuits of length $\gamma(P)$.

3.6. <u>An Initialized System</u>:

Over the last four sections, we've established the structural properties of the system $\mathcal{S}$. We're now ready to consider the behavioral properties of the <u>initialized system</u> $\mathcal{L}$.

Definition: $\mathcal{L} = \langle Z, D_{\mathcal{P}}, D_{\mathfrak{M}} \rangle$ where

$Z = \langle S, E, F, I \rangle$
$I \subseteq S$
$\forall A \in D_{\mathcal{P}}$: $|A \cap I| = 1$

I is the set of <u>initial conditions</u> of $\mathcal{L}$.
Z is the <u>initialized system net</u>.

The third requirement says that I assigns exactly one initial condition to each part.

If we think of $\mathcal{S}$ as the system 'hardware', then the set of initial conditions may be viewed as the system 'software'. With this interpretation, a piece of software (i.e. a program) has no meaning outside the context of a system. This is exactly as it should be.

Since Z is an initialized Petri net, the Simulation Rule can be applied.

Definition: The simulations of Z are called <u>system simulations</u>.

Because Z is covered by 1-token state components, namely the parts, the results of Section 2.3 are applicable:

Property 3.7: Within a system simulation, all instances of the same element are totally ordered.

Property 3.8: If <x,m> and <x,n> are instances within a system simulation, then <x,n> is the next instance of x following <x,m> iff n=m+1.

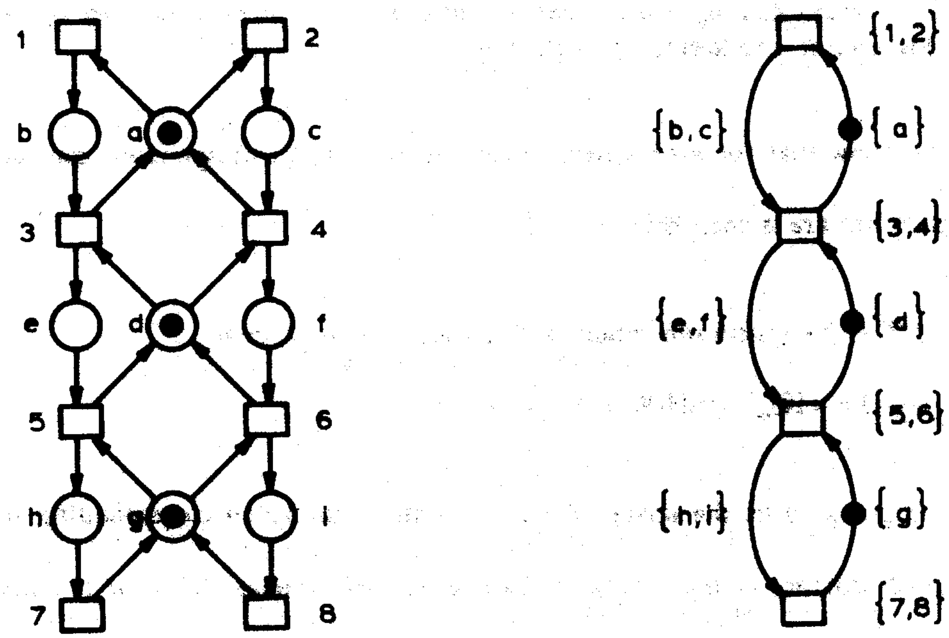
We've introduced the initialized system net, and now we introduce the 'initialized control structure'. We use the initial conditions of the system net to generate a corresponding initialization of the control structure.

Definition: $Z^* = \langle N^*, I^* \rangle$ where

$I^* = \{[s]_\alpha \mid s \in I\}$

Z^* is the <u>initialized control structure</u>

In Figure 3.11(a), we show an initialization of the system net from Figure 3.6(a). In Figure 3.11(b), we show the corresponding initialization of the control strucure from Figure 3.6(d).



(a) Z (b) Z^*

Figure 3.11 An Initialized System Net and Corresponding Initialized Control Structure

Definition: The simulations of Z^* are called <u>control simulations</u>.

In an event graph, each elementary circuit is a state component, and vice versa. The control structure has a special covering of elementary circuits generated by the parts. Because each part is assigned one token by I, the corresponding elementary circuit is assigned one token by I^*.

Theorem 3.8: Z^* is covered by 1-token state components.

Corollary 3.4: Within a control simulation, all instances of the same alternative class are totally ordered.

Corollary 3.5: If $<q,m>$ and $<q,n>$ are instances within a control simulation, then $<q,n>$ is the next instance of q following $<q,m>$ iff $n=m+1$.

We now show that for each system simulation, there is a corresponding control simulation, and two simulations are isomorphic.

Definition: If T is the system simulation <H,O,C> and $q \in H \cup O$, then

$$\Theta_T(q) = < [\hat{q}]_\alpha, |\{r \in H \cup O \mid r \leq q \wedge \hat{r} \alpha \hat{q}\}| >$$

$\Theta_T(q)$ is going to be the image of q in the control simulation corresponding to T. Notice that $\Theta_T(q)$ is an instance of the alternative class to which $\hat{q}$ belongs. Thus, if two instances in T are associated with alternatives, then those two instances will map into the same type of instance in the control simulation. Because of this, the instance number assigned to $\Theta_T(q)$ is not necessarily the instance number of q. We must count the number of instances in T that precede ($\leq$) q and are associated with the same alternative class as q. The instance number of $\Theta_T(q)$ will never be less than the instance number of q.

Definition: If T is the system simulation <H,O,C>, then,

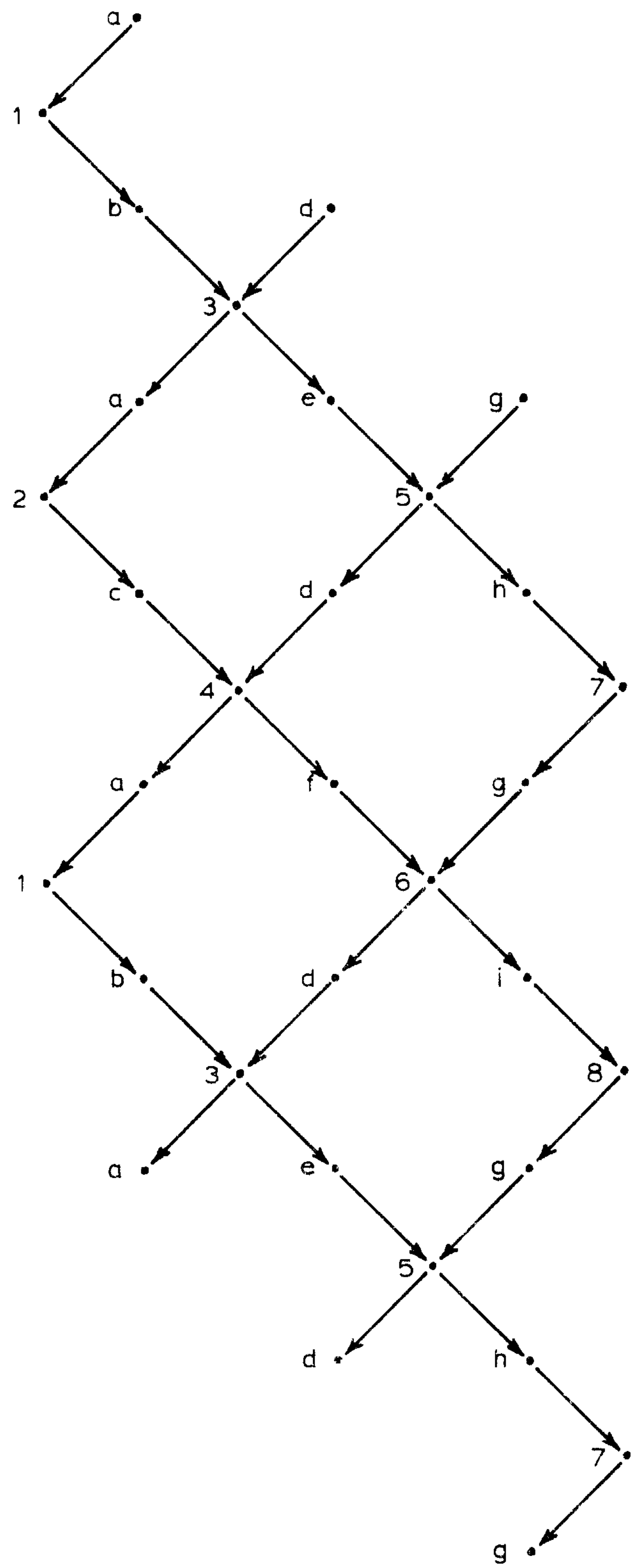
$$T^* = <\Theta_T(H), \Theta_T(O), \Theta_T(C)>^\dagger$$

In Figure 3.12(a) is a simulation of the initialized system net in Figure 3.11(a). In Figure 3.12(b) is the corresponding simulation generated by Θ_T. Notice that the <u>second</u> holding of State b in T corresponds to the <u>third</u> holding of Link {b,c} in T^*.

The next two theorems establish the relationship between T and T^* and the relationship between Z^* and T^*.

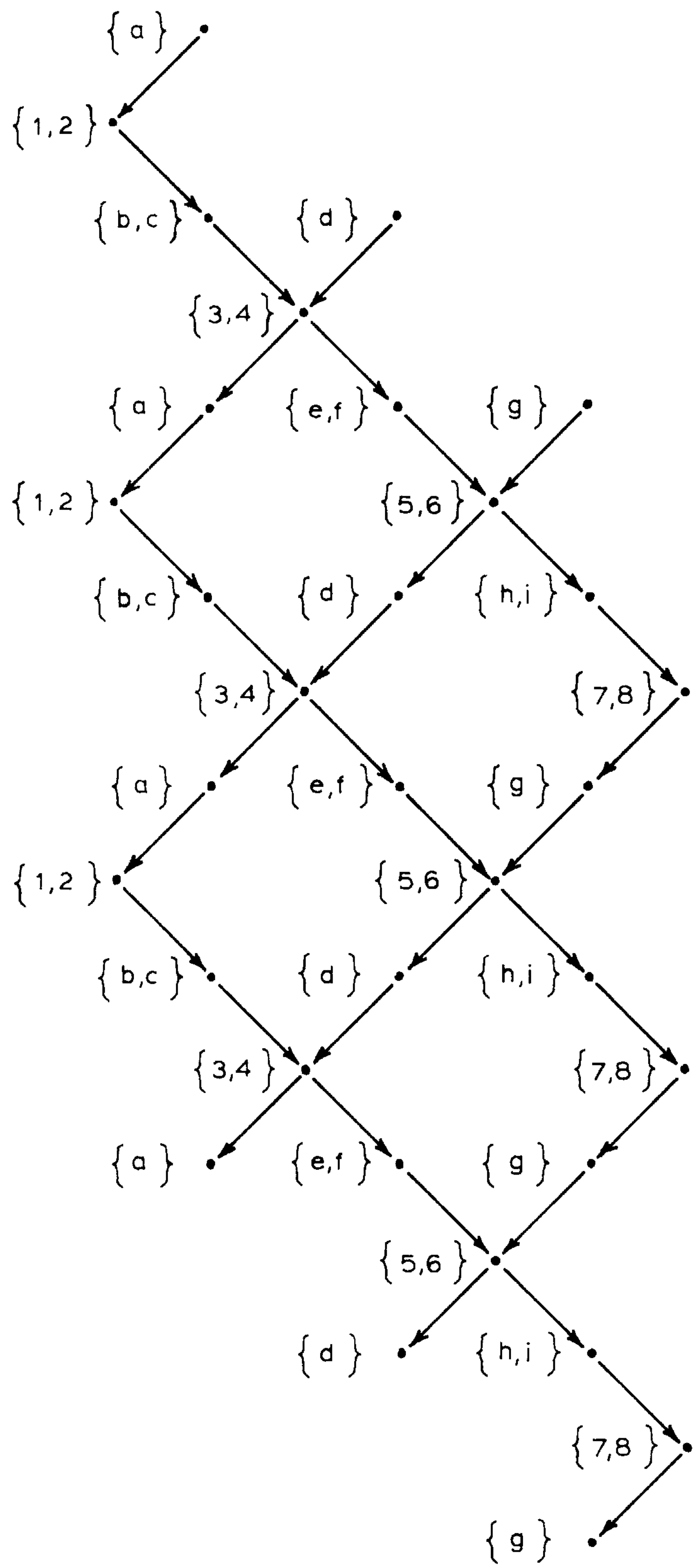
---

† $\Theta_T(C) = \{<\Theta_T(q), \Theta_T(r)> \mid <q,r> \in C\}$



(a) T

Figure 3.12 Corresponding Simulations



(b) T^*

Figure 3.12 (Continued)

Theorem 3.9: If T is a system simulation, then Θ_T is an isomorphism from T to T*.

Proof: Θ_T is clearly onto. Now suppose that $\Theta_T(q_1) = \Theta_T(q_2)$. Then

$$[\hat{q}_1]_\alpha = [\hat{q}_2]_\alpha \tag{1}$$

and,

$$|\{r \mid r \leq q_1 \wedge \hat{r} \alpha \hat{q}_1\}| = |\{r \mid r \leq q_2 \wedge \hat{r} \alpha \hat{q}_2\}| \tag{2}$$

Let $c = [\hat{q}_1]_\alpha = [\hat{q}_2]_\alpha$. Since c is an alternative class, there exists a part containing all the elements of c. Consequently, the instances of elements belonging to c are totally ordered. Line (2) says that q_1 and q_2 appear at the same point in that total ordering. Hence, $q_1 = q_2$ and Θ_T is 1-1. If C is the causality relation for T and C* the causality relation for T*, then it follows immediately from the definition of T* that,

$$\langle q, r\rangle \in C \iff \langle \Theta_T(q), \Theta_T(r)\rangle \in C^*$$

□

Theorem 3.10: If T is a system simulation, then T* is a control simulation.

Proof: If T is the initial simulation of Z, then $T = \langle I\times\{1\}, \phi, \phi\rangle$ and $T^* = \langle\{[s]_\alpha | s\in I\}\times\{1\}, \phi, \phi\rangle$. But $\{[s]_\alpha | s\in I\} = I^*$, and therefore, T* is the initial simulation of Z*.

Suppose now that T_1 satisfies the requirements of the theorem, and that T_2 is derived from T_1 through a single application of Step 2 of the Simulation Rule. Let,

$T_1 = \langle H_1, O_1, C_1\rangle$ $\quad$ $T_2 = \langle H_2, O_2, C_2\rangle$

$T_1^* = \langle H_1^*, O_1^*, C_1^*\rangle$ $\quad$ $T_2^* = \langle H_2^*, O_2^*, C_2^*\rangle$

Now there must exist a set of holdings A in the front boundary of T_1 consisting of one holding for each precondition of an event e and such that,

$$H_2 = H_1 \cup \eta(e^\bullet, H_1)$$
$$O_2 = O_1 \cup \eta(e, O_1)$$
$$C_2 = C_1 \cup A\times\eta(e,O_1) \cup \eta(e,O_1)\times\eta(e^\bullet,H_1)$$

We must show that a similar relationship exists between T_1^* and T_2^*. We note that $[e]_\alpha \in E^*$. Let $A^* = \Theta_{T_1}(A)$. Because A is contained in the front boundary of T_1, and Θ_{T_1} is an isomorphism, A* must be contained in the front boundary of T_1^*. By Theorem 14(a

& b), A^* consists of one holding for each precondition of $[e]_\alpha$. For the three components of T_2^* we have,

$$H_2^* = \Theta_{T_2}(H_2) = \Theta_{T_2}(H_1) \cup \Theta_{T_2}(\eta(e^*, H_1)) \tag{1}$$

$$O_2^* = \Theta_{T_2}(O_2) = \Theta_{T_2}(O_1) \cup \Theta_{T_2}(\eta(e, O_1)) \tag{2}$$

$$C_2^* = \{<\Theta_{T_2}(q), \Theta_{T_2}(r)> \mid <q,r> \in C_2\} = \{<\Theta_{T_2}(q), \Theta_{T_2}(r)> \mid <q,r> \in C_1\} \cup \{<\Theta_{T_2}(q), \Theta_{T_2}(r)> \mid <q,r> \in A \times \eta(e, O_1)\} \cup \{<\Theta_{T_2}(q), \Theta_{T_2}(r)> \mid <q,r> \in \eta(e, O_1) \times \eta(e^*, H_1)\} \tag{3}$$

Since T_1 is, in effect, a 'prefix' of T_2, we have $\Theta_{T_2}(q) = \Theta_{T_1}(q)$ for all $q \in H_1 \cup O_1$. Thus,

$$\Theta_{T_2}(A) = \Theta_{T_1}(A) = A^*$$

$$\Theta_{T_2}(H_1) = \Theta_{T_1}(H_1) = H_1^* \tag{4}$$

$$\Theta_{T_2}(O_1) = \Theta_{T_1}(O_1) = O_1^* \tag{5}$$

$$\{<\Theta_{T_2}(q), \Theta_{T_2}(r)> \mid <q,r> \in C_1\} = \{<\Theta_{T_1}(q), \Theta_{T_1}(r)> \mid <q,r> \in C_1\} = C_1^* \tag{6}$$

Let $<s,n>$ be a holding in $\eta(e^*, H_1)$. Because $[s]_\alpha$ is an alternative class, there exists a part containing all the states of $[s]_\alpha$. Therefore, the holdings of states belonging to $[s]_\alpha$ are totally ordered in both T_1 and T_2. Furthermore, $<s, n>$ is the last holding in the total ordering of T_2. From all this, we get,

$\Theta_{T_2}(\eta(e^*, H_1)) = \{<[s]_\alpha, n> \mid s \in e^*$ and n is the number of holdings in H_2 of states belonging to $[s]_\alpha\}$

$= \{<[s]_\alpha, n> \mid s \in e^*$ and n is the number of holdings in H_1 of states belonging to $[s]_\alpha$ - plus 1$\}$

But by Theorem 14(c & d), $\{[s]_\alpha \mid s \in e^*\} = \{[s]_\alpha \mid [s]_\alpha \in [e]_\alpha^\diamond\}$. This, together with the fact that Θ_{T_1} is a bijection, gives us,

$\Theta_{T_2}(\eta(e^*, H_1)) = \{<[s]_\alpha, n> \mid [s]_\alpha \in [e]_\alpha^\diamond$ and n is the number of instances of $[s]_\alpha$ in H_1^*- plus 1$\}$

$$= \eta([e]_\alpha^\diamond, H_1^* \tag{7}$$

Similarly,

$$\Theta_{T_2}(\eta(e, O_1)) = \eta([e]_\alpha, O_1^*) \tag{8}$$

From Lines (7) and (8) and the fact that $\Theta_{T_2}(A) = A^*$,

$$\{<\Theta_{T_2}(q), \Theta_{T_2}(r)> \mid <q,r> \in A \times \psi(s,O_1)\} = A^* \times \psi([s]_{\alpha}, O_1^*) \tag{9}$$

$$\{<\Theta_{T_2}(q), \Theta_{T_2}(r)> \mid <q,r> \in \psi(s,O_1) \times \psi(s^{\diamond}, H_1)\} = \psi([s]_{\alpha}, O_1^*) \times \psi([s]_{\alpha}^{\diamond}, H_1^*) \tag{10}$$

Finally, we get,

$H_2^* = H_1^* \cup \psi([s]_{\alpha}^{\diamond}, H_1^*)$ Lines 1, 4, & 7

$O_2^* = O_1^* \cup \psi([s]_{\alpha}, O_1^*)$ Lines 2, 5, & 8

$C_2^* = C_1^* \cup A^* \times \psi([s]_{\alpha}, O_1^*) \cup \psi([s]_{\alpha}, O_1^*) \times \psi([s]_{\alpha}^{\diamond}, O_1^*)$ Lines 3, 6, 9, & 10

Since, by hypothesis, T_1^* is a simulation of Z^*, we must conclude that T_2^* is also a simulation of Z^*. □

Corollary 3.6: For each system simulation, there is a corresponding control simulation, and the two simulations are isomorphic.

Note that the correspondence between system simulations and control simulations is not, in general, one-to-one. In most cases, there will be several system simulations mapping into a single control simulation.

3.7. Examples of Initialized Systems:

In this section, we present three examples of initialized systems. For each one we provide an interpretation. The formal techniques developed in the following chapters will confirm these interpretations.

The initialized system depicted in Figure 3.13 represents a three-stage bit pipeline. The three parts comprise the three stages. Each mode corresponds to the shifting of constant 'information'. 'Bits' enter at the topmost stage and are passed from stage to stage until they are lost at the bottommost stage.

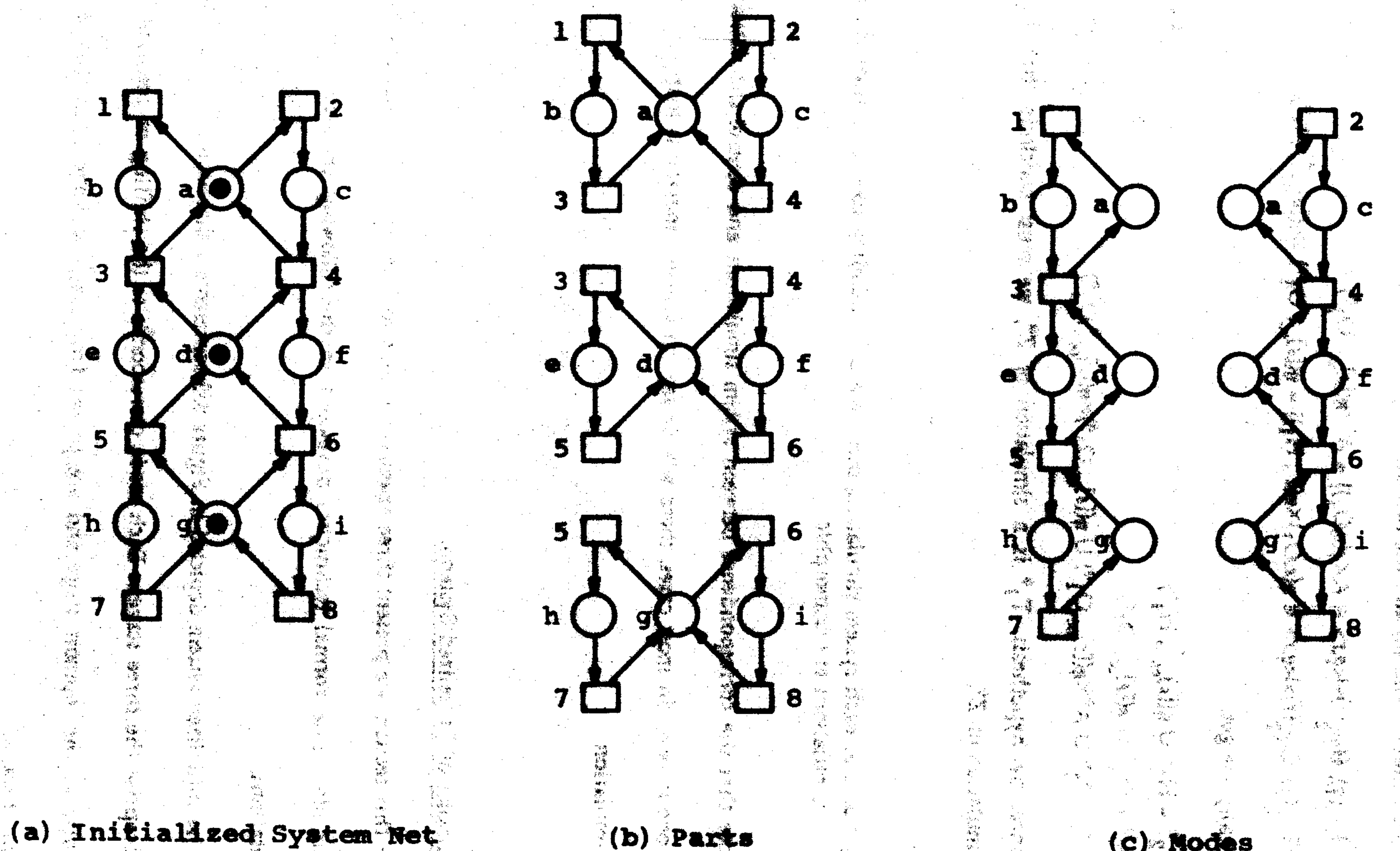
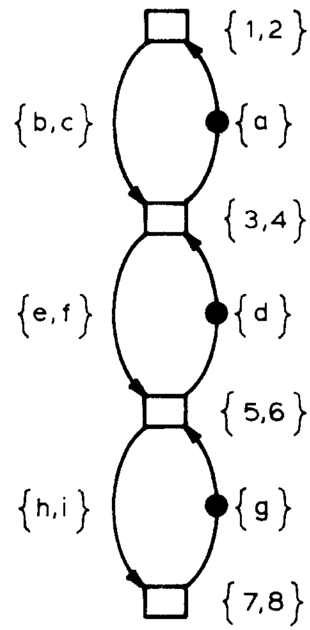


Figure 3.13 Three-Stage Bit Pipeline

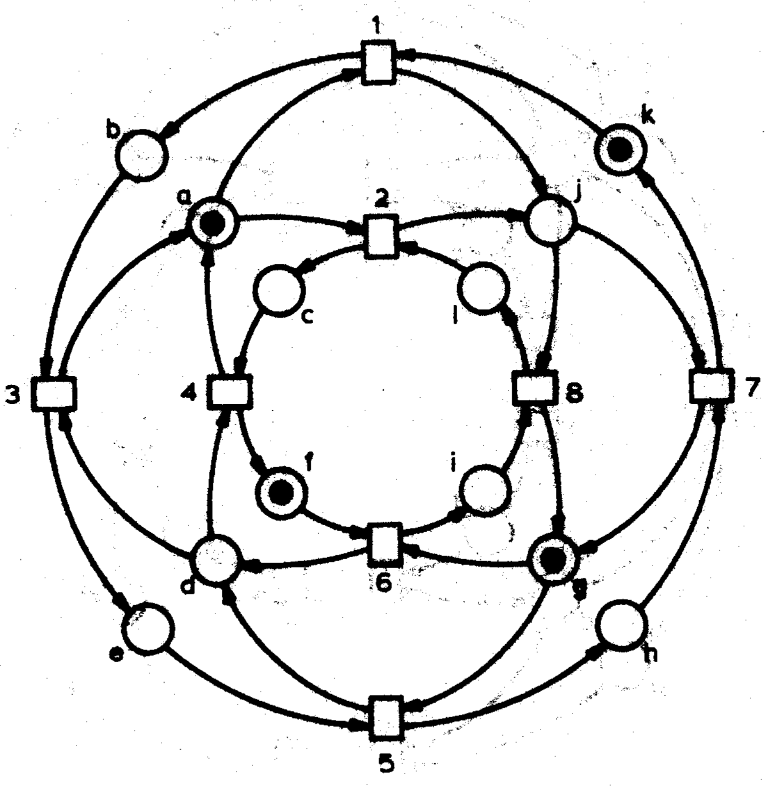


(d) Initialized Control Structure

Figure 3.13 (Continued)

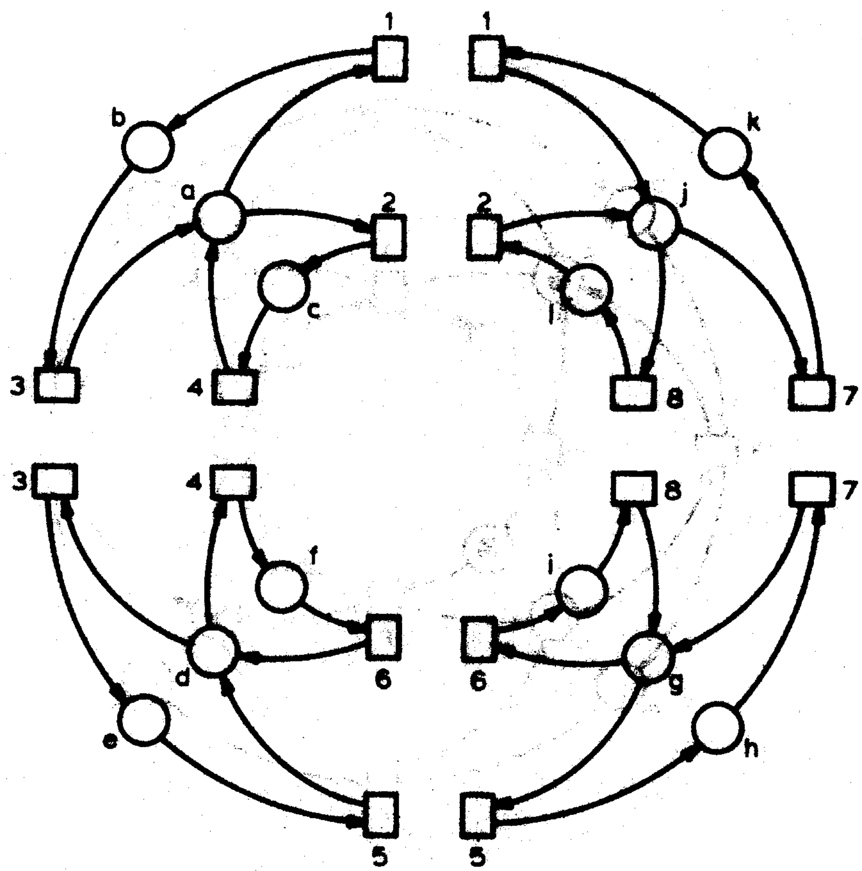
The initialized system shown in Figure 3.14 is formed by taking a four-stage bit pipeline and connecting the first and last stages. The result is a circulating bit pipeline. Notice that in this case, 'bits' are conserved.

The initialized system of Figure 3.15 describes a half adder. The choices associated with States *a* and *b* represent the two binary inputs. The reverse choices associated with States *k* and *l* represent, respectively, the sum and carry outputs. There are four modes and they correspond to the four possible operations. Notice that the system net is covered by four 1-token state components. We've chosen two of these as our parts. Although the choice is arbitrary, it has no effect on the resulting control structure. (There are, however, situations in which this is not the case. The net in Figure 3.16 is an example. Depending on which covering of state components is selected, either of two control structures will be generated. The significance of this is not yet understood.)



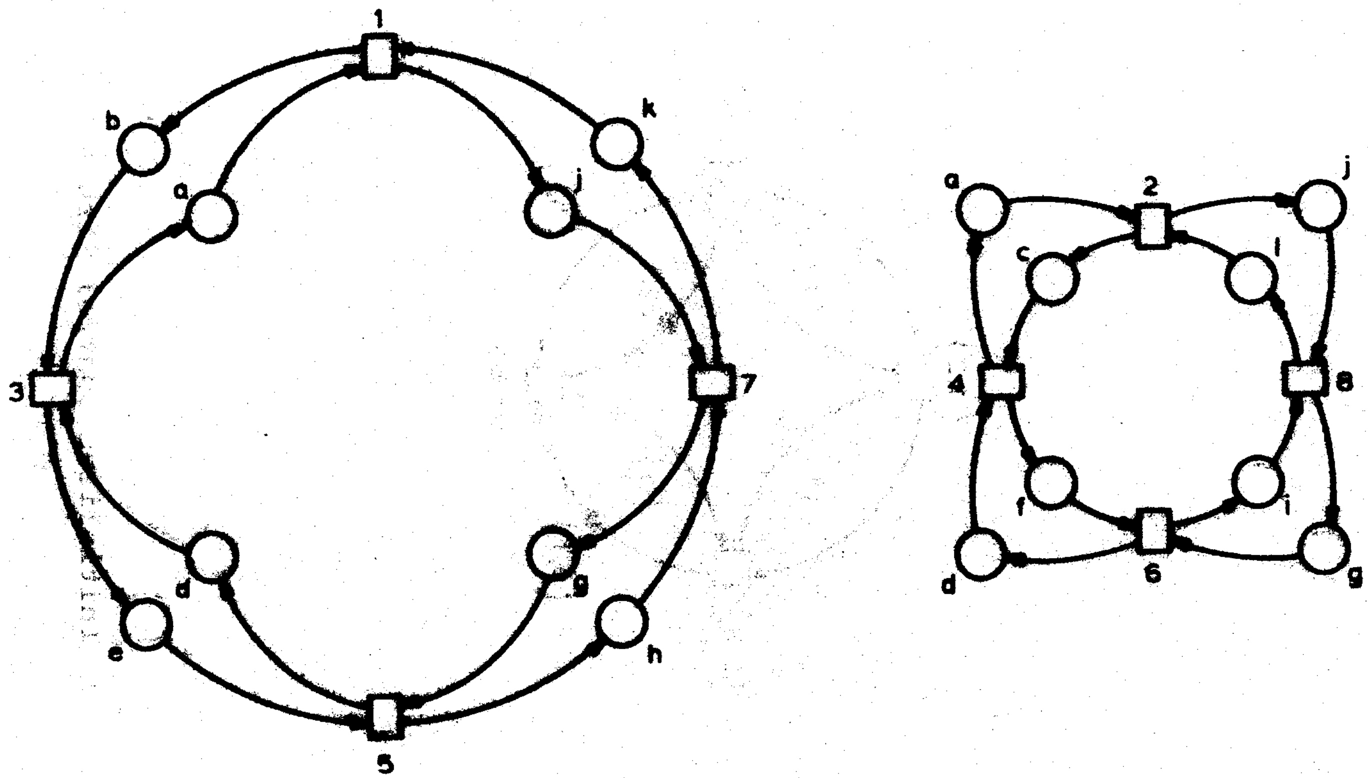
(a) Initialized System Net

Figure 3.14 Circulating Bit Pipeline



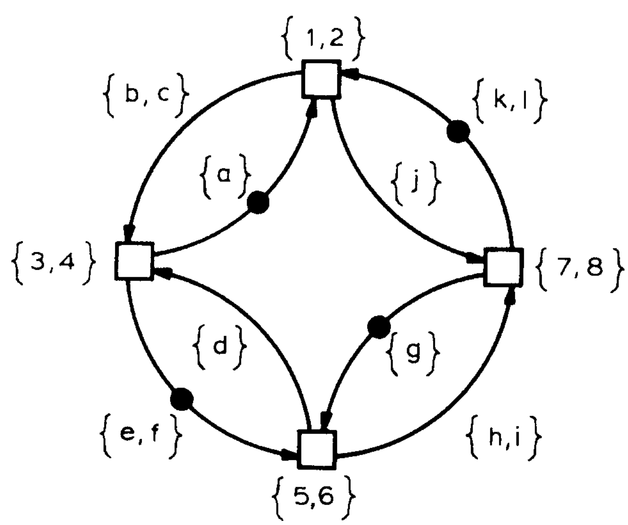
(b) Parts

Figure 3.14 (Continued)



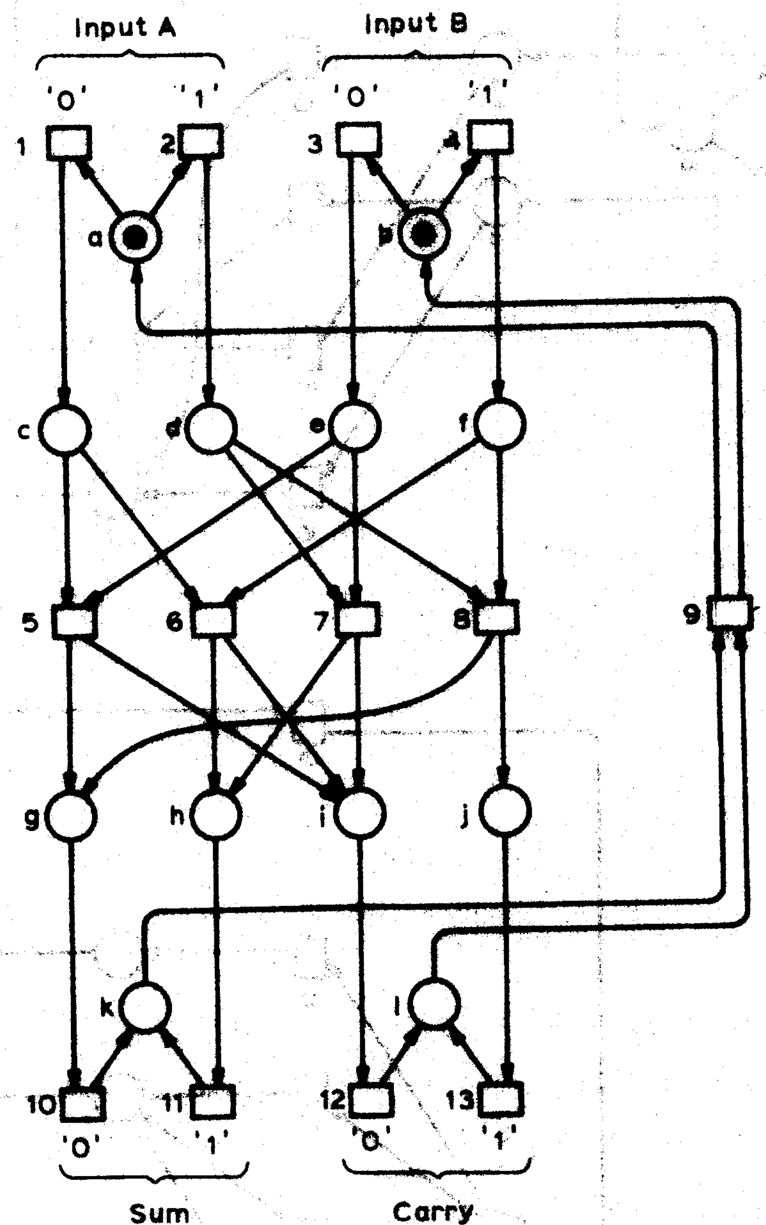
(c) Modes

Figure 3.14 (Continued)



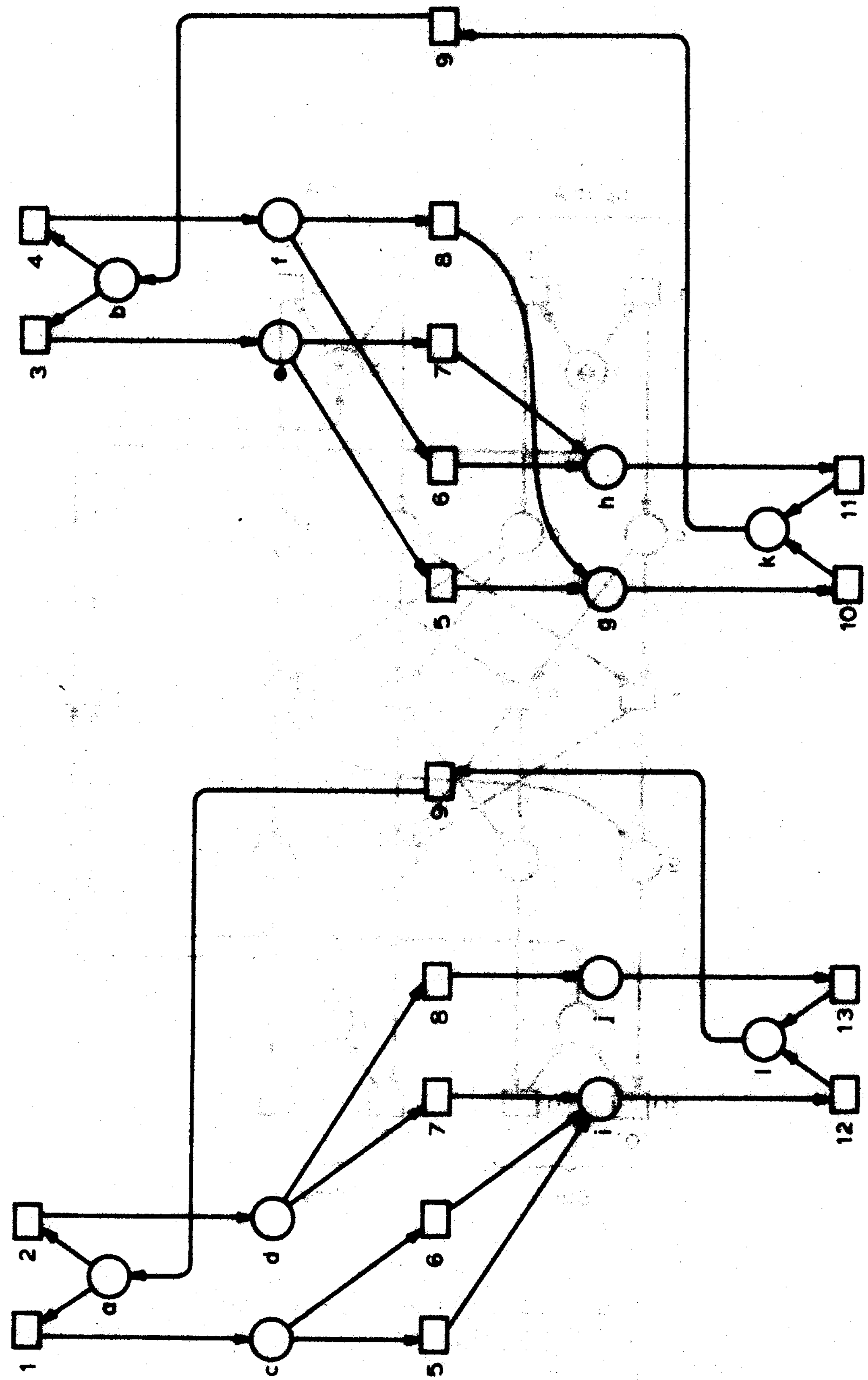
(d) Initialized Control Structure

Figure 3.14 (Continued)



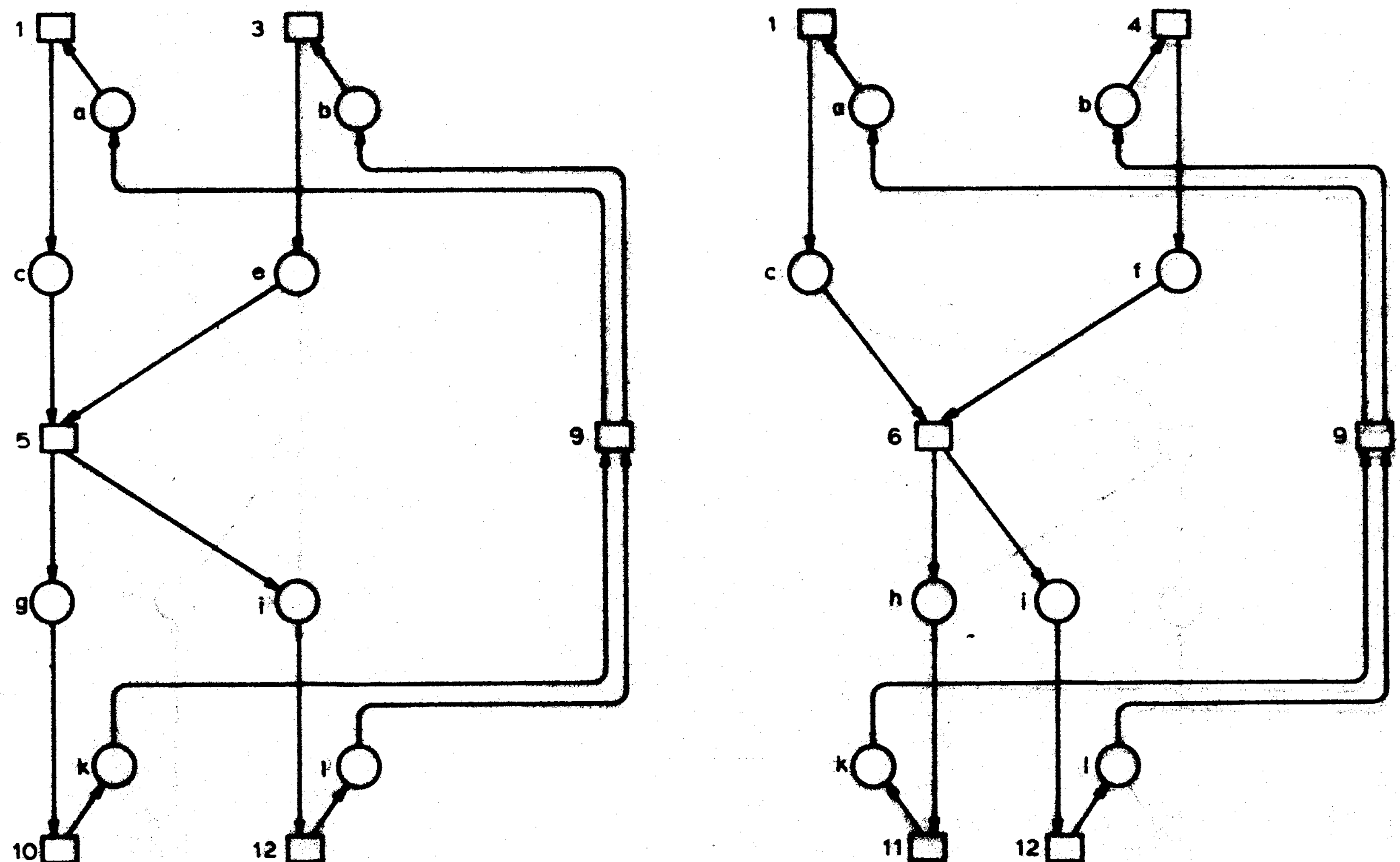
(a) Initialized System Net

Figure 3.15 Half Adder



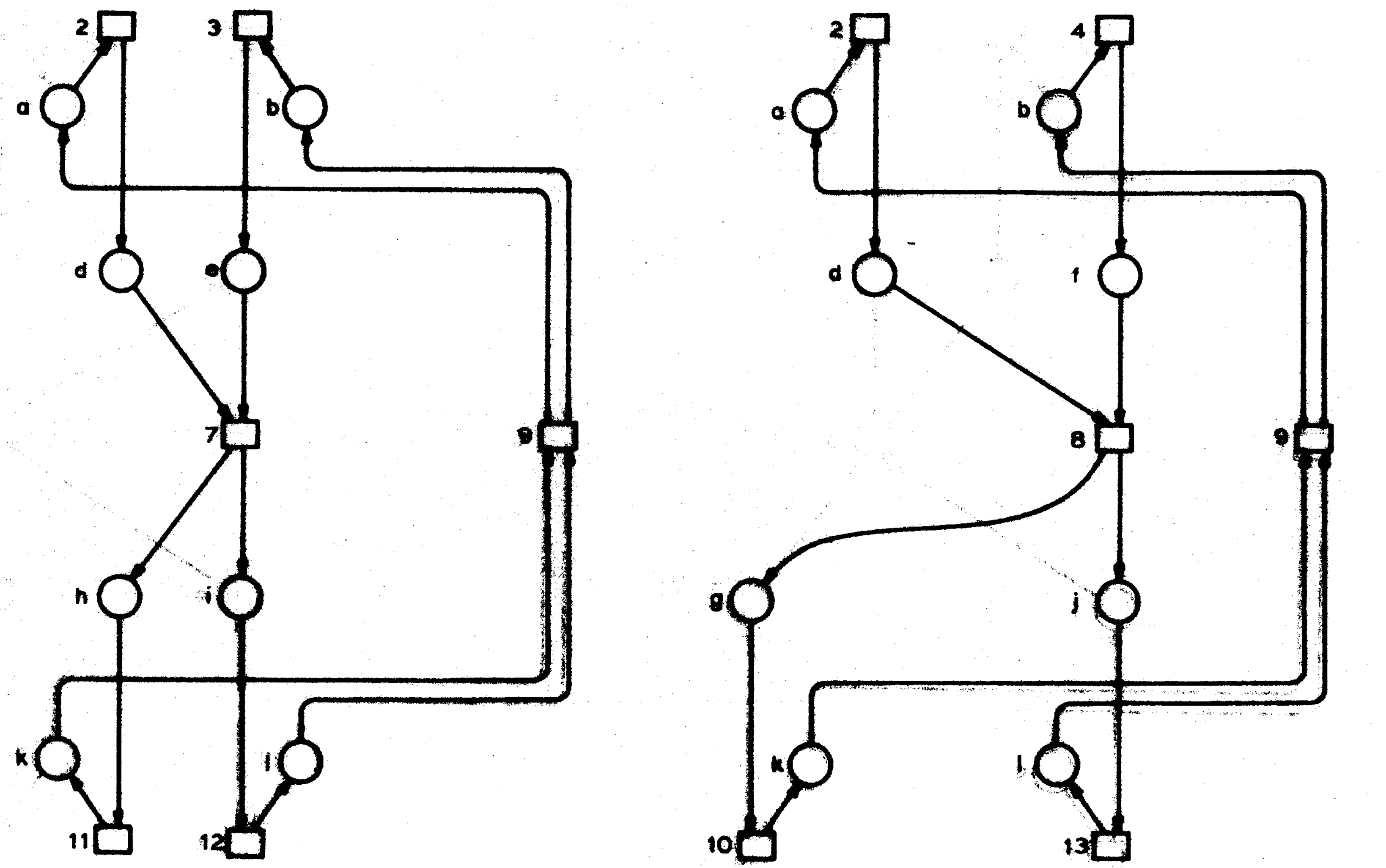
(b) Parts

Figure 3.15 (Continued)



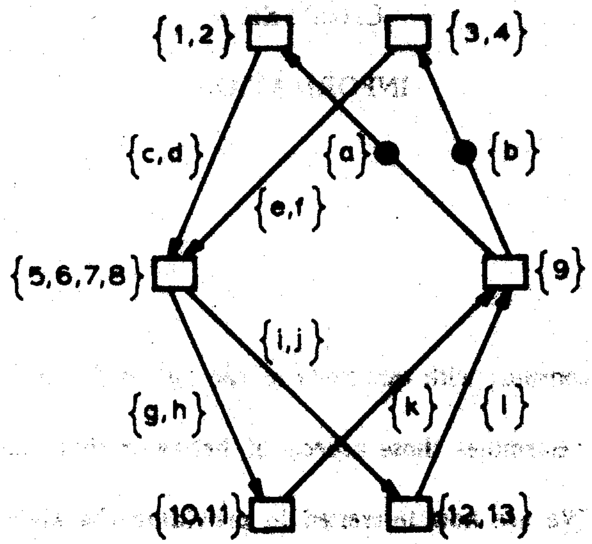
(c) Modes

Figure 3.15 (Continued)



(c) Modes

Figure 3.15 (Continued)



(d) Initialized Control Structure

Figure 3.15 (Continued)

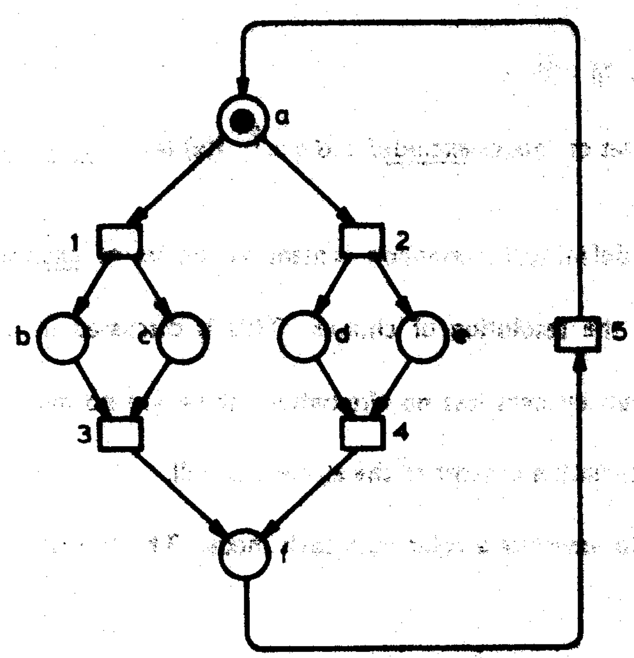


Figure 3.16 System Net with Two Control Structures

CHAPTER 4

INFORMATION

4.1. Information Content:

In this chapter we continue with our study of the system $\mathcal{S}$. In Chapter 3, we showed that the control structure N* determines those aspects of behavior that result when alternatives are made indistinguishable. We are now interested in providing the ability to distinguish between alternatives. We introduce the notion of 'information content' for that purpose.

Definition: The set of parts containing Element x is denoted by $\mathcal{P}(x)$. The set of modes containing Element x is denoted by $\mathfrak{M}(x)$.

Definition: For $x \in X$,

$$I(x) = \mathfrak{M} - \mathfrak{M}(x)$$

'I(x) is the set of modes <u>excluded</u> from x.'[†] I(x) is the <u>information content</u> of x.

The reason for defining information content as the set of <u>excluded</u> rather than <u>included</u> modes has to do with the resolution of choices. This is discussed in Sections 4.3 and 4.4. We might note that when an element has no alternatives, there are no modes excluded (Theorem 3.5) and, therefore, the information content of the element is null.

It is convenient to associate a color with each mode. The information content of an element

[†] This resembles a definition by Holt and Commoner [13]. In the context of a strongly connected state graph, they defined the 'information set' of a state to be the set of excluded elementary circuits.

can then be viewed as the set of colors associated with the excluded modes. In Figures 4.1 through 4.3 are the system nets for the systems described in Figure 3.13 through 3.15. We've associated a color with the events defining each mode. Next to each system element is its information content expressed as a set of colors.

We now show that information content does indeed distinguish between alternatives.

Theorem 4.1: $\forall x_1, x_2 \in X$:

$$[x_1]_\alpha = [x_2]_\alpha \wedge I(x_1) = I(x_2) \Leftrightarrow x_1 = x_2$$

'If two elements belong to the same alternative class and have the same information content, then they must be the same element.'

Proof: $\Leftarrow$ Obvious

$\Rightarrow$ This part of the theorem follows from these observations,

(a) $I(x_1) = I(x_2) \Rightarrow \mathfrak{M}(x_1) = \mathfrak{M}(x_2)$
(b) Each element is contained in at least one mode.
(c) A mode intersects an alternative class in exactly one element. □

So now a system element can be uniquely identified by specifying two things: (1) the <u>alternative class</u> to which it belongs, and (2) its <u>information content</u>.

4.2. Information Flow:

Suppose that q_1 and q_2 are instances in a system simulation, and that there is an elementary causal connection leading from q_1 to q_2.

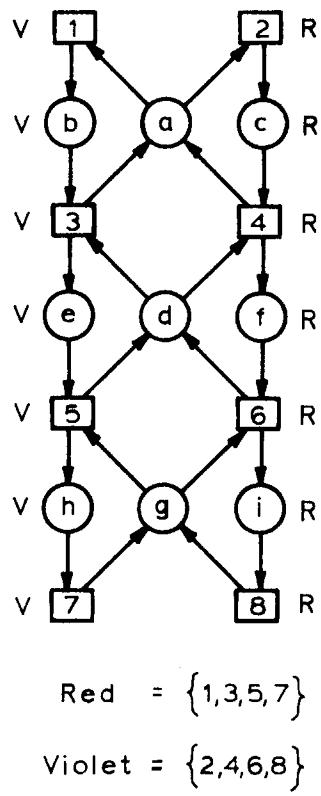
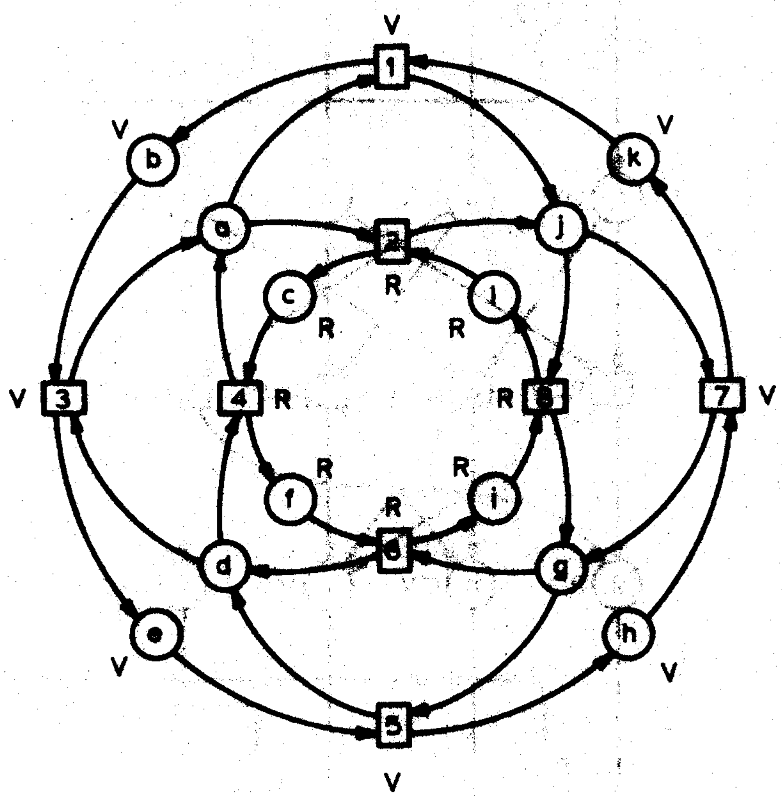


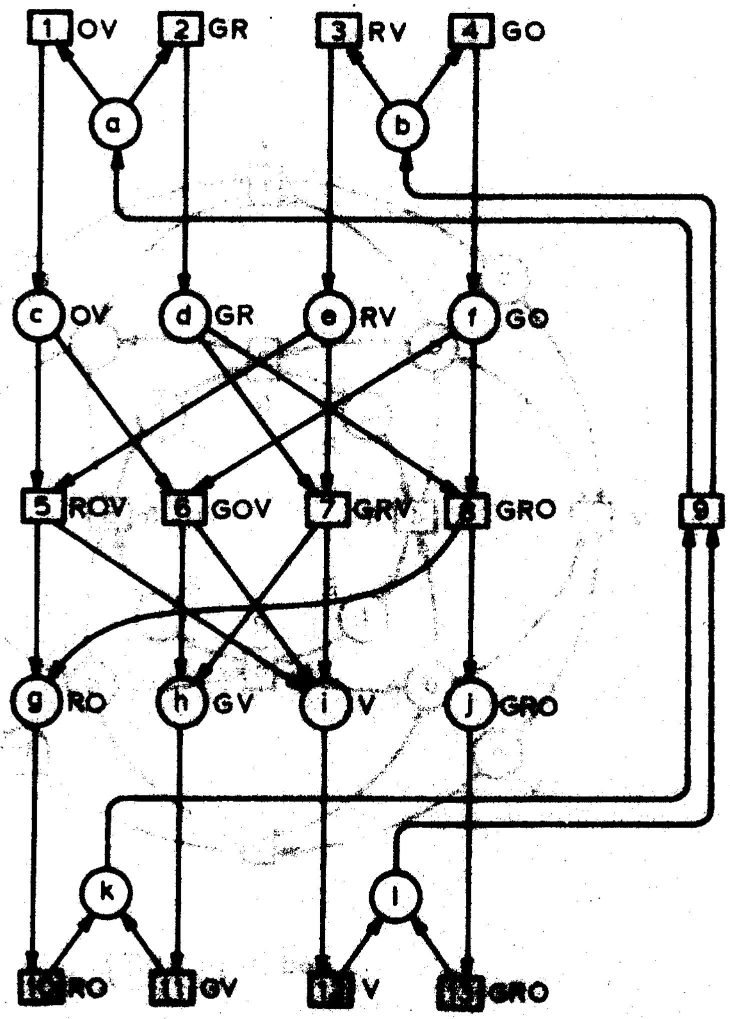
Figure 4.1 Bit Pipeline - Information Contents of the System Elements



Red = {1,3,5,7}

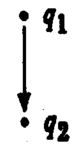
Violet = {2,4,6,8}

Figure 4.2 Circulating Bit Pipeline - Information Contents of the System Elements



Green	=	{1,3,5,9,10,12}
Red	=	{1,4,6,9,11,12}
Orange	=	{2,3,7,9,11,12}
Violet	=	{2,4,8,9,10,13}

Figure 4.3 Half Adder - Information Contents of the System Elements



Let q_1 be an instance of x_1, and q_2 an instance of x_2. Associated with q_1 is the information content of x_1, and associated with q_2 is the information content of x_2. We shall interpret the information that is common to both x_1 and x_2 as 'flowing' from q_1 to q_2.

Our convention of associating modes with colors permits a graphic representation of information flow. The arcs of a system simulation are colored according to the following algorithm: An arc connecting instance $\langle x_1,n_1 \rangle$ with instance $\langle x_2,n_2 \rangle$ is assigned a particular color iff the mode represented by that color is contained in $I(x_1) \cap I(x_2)$. In Figures 4.4 through 4.6 are some simulations for the systems described in Figures 3.13 through 3.15. Using the correspondence between colors and modes given in Figures 4.1 through 4.3, we've indicated the colors assigned to each arc. The reader is encouraged to do the actual coloring. Note that some arcs may be assigned several colors, while other arcs may be assigned no colors at all.

This formalization of information flow corresponds remarkably well with intuition. In Figure 4.4, we can see quite clearly the flow of 'bits'† down the bit pipeline. The two colors correspond to the two different bits. At Events 1 and 2, bits enter the pipeline. At Events 3 and 4, the bits are transferred from the first to the second stage. At Events 5 and 6, the bits are transferred from the second to the third stage. And finally, at Events 7 and 8, the bits are lost.

As expected, in the circulating bit pipeline, bits are conserved. As shown in Figure 4.5, the same two bits are present at the beginning of the simulation and the end of the simulation.

† The notion of a 'bit' is very restrictive and is used here only in an informal manner. Formally, information is expressed in terms of excluded modes, not in terms of bits.

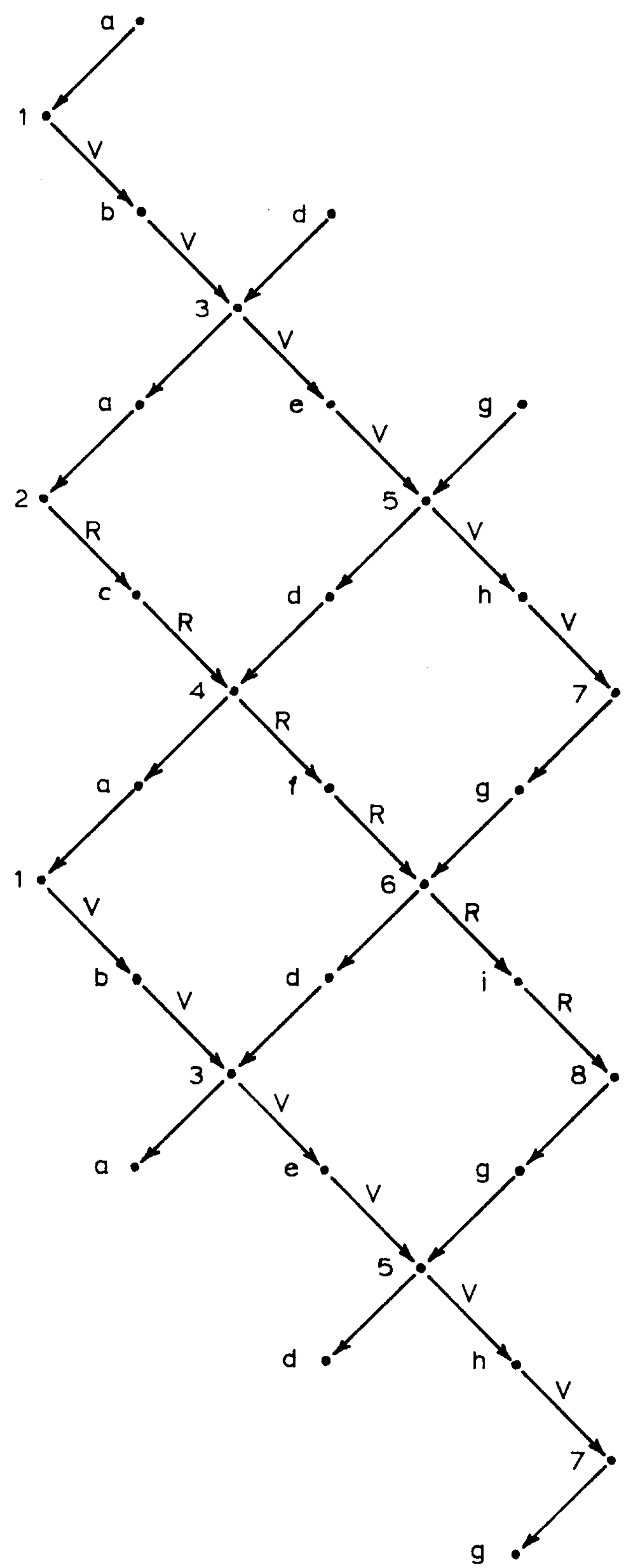


Figure 4.4 Bit Pipeline - Information Flow

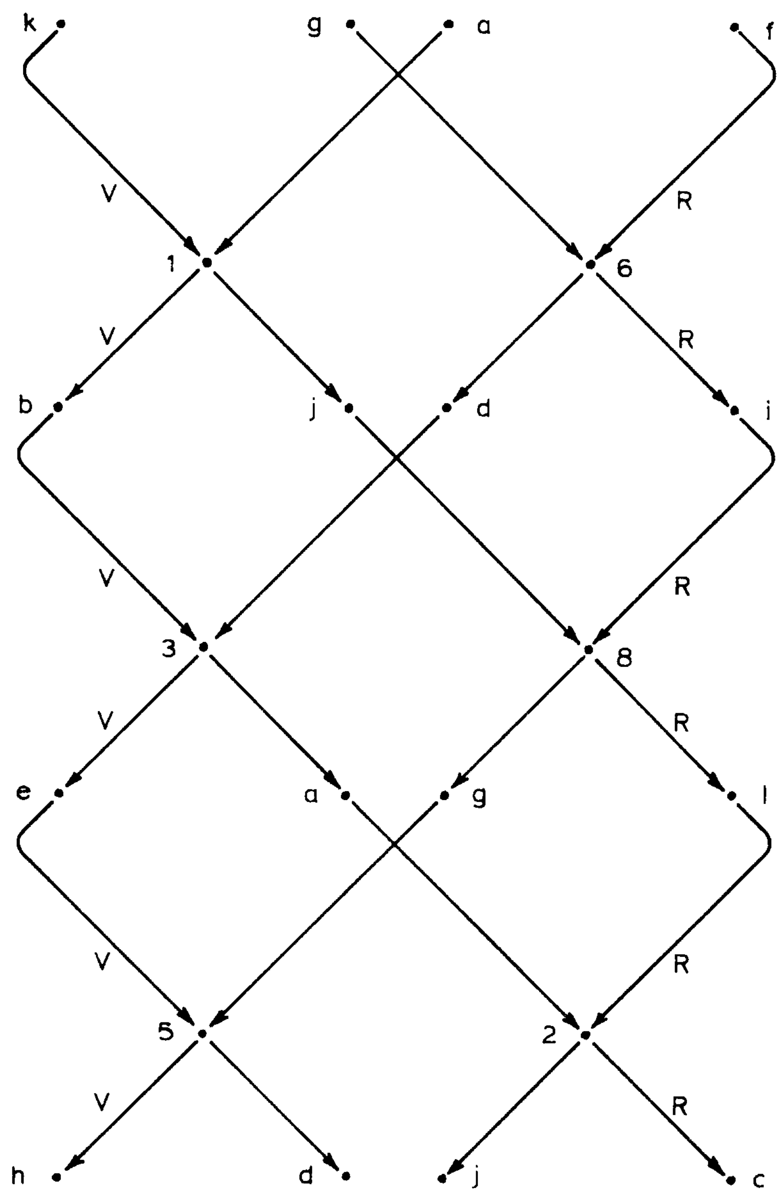


Figure 4.5 Circulating Bit Pipeline - Information Flow

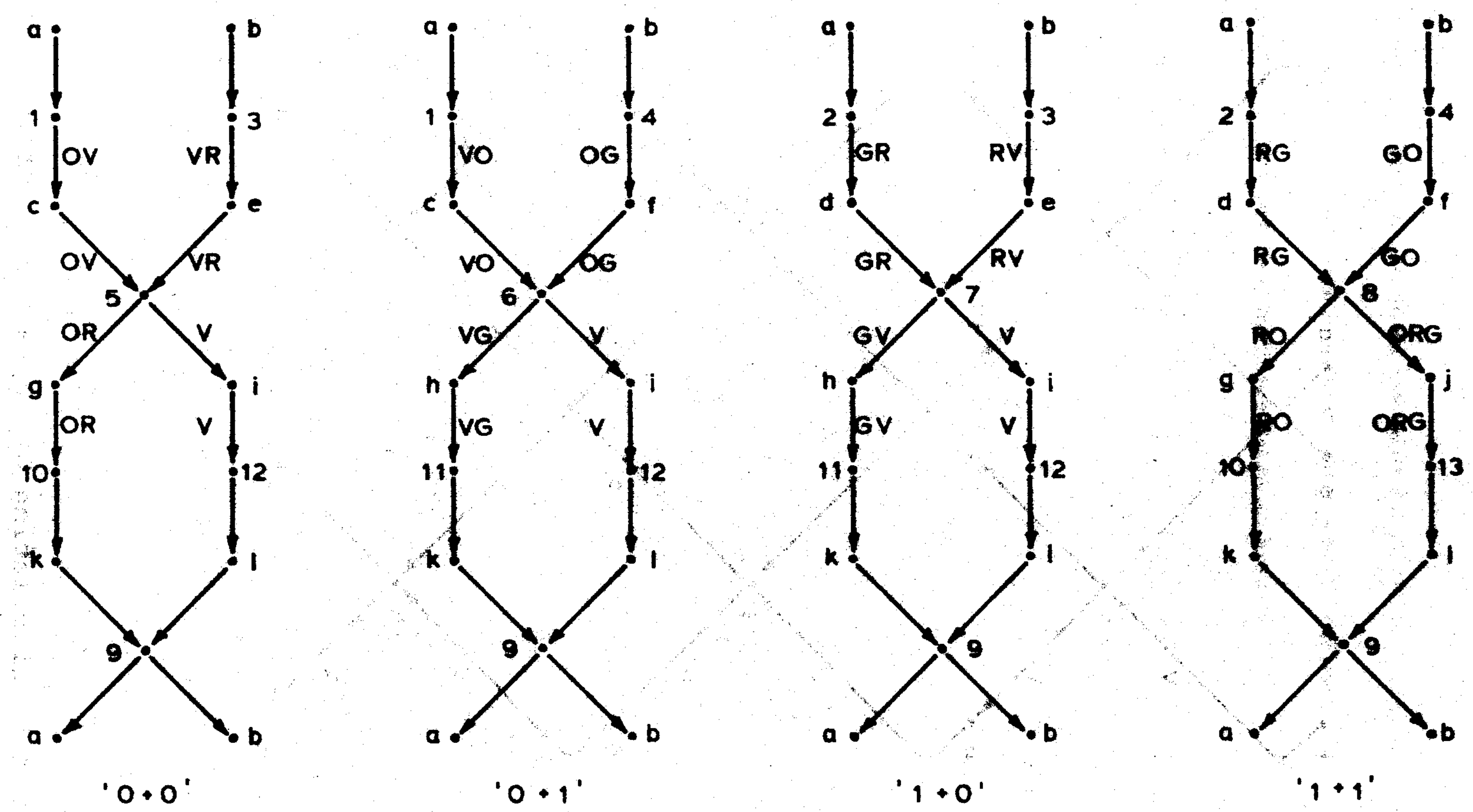


Figure 4.6 Half Adder - Information Flow

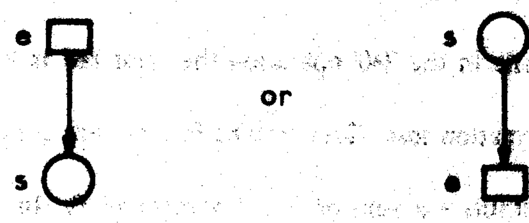
With the half adder, the situation becomes more complicated. It is no longer possible to interpret information flow in terms of bits. But the flow of information still corresponds to our intuition. As shown in Figure 4.6, information enters at the designated inputs - Events 1, 2, 3, and 4 - and is lost at the designated outputs - Events 10, 11, 12, and 13. Notice that in each of the two middle simulations, information is also lost at an interior event. In the '0+1' operation, the color orange is lost at Event 6, while in the '1+0' operation the color red is lost at Event 7. At Events 5 and 8, there is no such information loss. The reasons for this are simple. In the case of both 0+1 and 1+0, we get the same outputs - a sum of 1 and a carry of 0. In these two situations we are unable to reconstruct the inputs from the outputs. The information lost at Events 6 and 7 is what allows us to distinguish between 0+1 and 1+0. In the cases of 0+0 and 1+1, the conservation of information at Events 5 and 8 corresponds to the fact that, in both cases, the inputs can be reconstructed from the outputs. This short discussion is a preview of the ideas contained in the next two sections and in Chapter 6.

We mention now an interpretation for information flow that the reader might find helpful. We've shown that the control structure determines those aspects of behavior that result when the alternatives in an alternative class are lumped together. We've also shown that information content provides a way of distinguishing between alternatives. Our practice of associating modes with colors then permits us to think of information as colors assigned to the 'tokens' on the control structure. The colors assigned to each token determine a unique system element. By defining appropriate color transformations for the meetings in the control structure, we can duplicate the behavior of the original system.

4.3: Resolving Choice

We've said nothing so far about the relationship between information content and the resolution of choices. Such a relationship exists, and it is a very natural one.

Suppose that a state and an event in the system net are connected:



We might ask how the information contents of e and s are related. The following theorem answers that question.

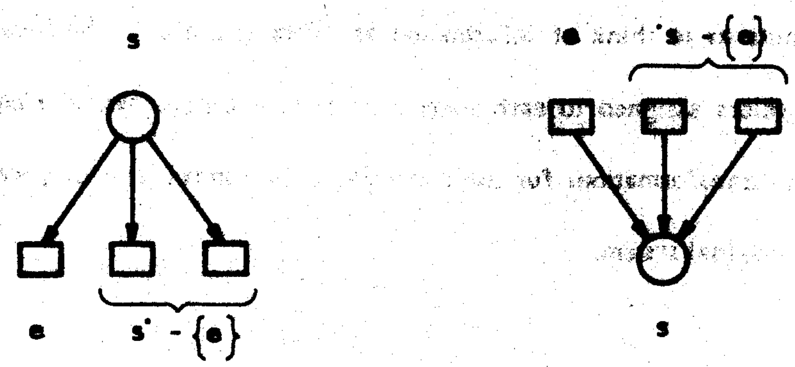
Theorem 4.2: $\forall e \in E$: $\forall s \in S$:

$$s \cdot e \Rightarrow I(e) = I(s) \cup \mathfrak{M}(s^{\cdot} - \{e\}) \quad \text{(a)}$$

$$e \cdot s \Rightarrow I(e) = I(s) \cup \mathfrak{M}({}^{\cdot}s - \{e\}) \quad \text{(b)}$$

'The information content of e equals the information content of s plus the set of modes covering all output (input) events of s except e.'

These are the two situations:



(a) (b)

Proof: We prove just Part (a).

Because an event component selects all arcs into an event,

$$\mathfrak{M}(s) = \mathfrak{M}(s^{\bullet})$$

Since $e \in s^{\bullet}$,

$$\mathfrak{M}(s^{\bullet}) = \mathfrak{M}(e) \cup \mathfrak{M}(s^{\bullet}-\{e\})$$

Thus,

$$\mathfrak{M}(s) = \mathfrak{M}(e) \cup \mathfrak{M}(s^{\bullet}-\{e\})$$

Because $\mathfrak{M}(e) \cap \mathfrak{M}(s^{\bullet}-\{e\}) = \phi$ (Theorem 3.5),

$$\mathfrak{M}(e) = \mathfrak{M}(s) - \mathfrak{M}(s^{\bullet}-\{e\})$$

And,

$$\mathfrak{M}-\mathfrak{M}(e) = \mathfrak{M}-(\mathfrak{M}(s)-\mathfrak{M}(s^{\bullet}-\{e\})) = (\mathfrak{M}-\mathfrak{M}(s)) \cup \mathfrak{M}(s^{\bullet}-\{e\})$$

From the definition of information content we get,

$$I(e) = I(s) \cup \mathfrak{M}(s^{\bullet}-\{e\})$$ □

Corollary 4.1: $\forall s \in S$: $\forall e \in E$:

$$(s \bullet e \vee e \bullet s) \Rightarrow I(s) \subseteq I(e)$$

'The information content of an event contains the information content of each precondition and each postcondition of the event.'

To illustrate Theorem 4.2, we note that in Figure 4.1, I(3)={V}, I(d)=ϕ, and $\mathfrak{M}$(4)={V}. So we have I(3)=I(d)∪$\mathfrak{M}$(4) as predicted. Corollary 4.1 means that, with our scheme for coloring the arcs of a system simulation, the colors entering and leaving a holding are the same. In other words, colors appear and disappear only at occurrences.

What is the significance of Theorem 4.2? In the case of $s{\bullet}e$, e is one of the alternatives in the forwards choice associated with s. When that choice is encountered in the course of a system simulation, it will have to be resolved. (We are not concerned at the moment whether this is a free choice or a constrained choice, or possibly a combination of the two.) Resolving the choice is equivalent to selecting one of the alternatives in $s^{\bullet}$. But selecting an event in $s^{\bullet}$ is equivalent to specifying the modes that cover the remaining events in $s^{\bullet}$. (Given $\mathfrak{M}(s^{\bullet}-\{e\})$, we can determine $\mathfrak{M}(e)$ and thus e.) Therefore, the information gained in going from s to one of the events in $s^{\bullet}$ resolves the forwards choice associated with s. Notice that when e is the only 'alternative' in $s^{\bullet}$, there is no choice to be resolved and there is no information gained.

For the case where $e{\bullet}s$, everything is reversed. We are now dealing with the backwards choice associated with s. Instead of talking about the information gained in going from s to e, we talk about the information lost in going from e to s. That information resolves the backwards choice associated with s. It is what we would need to 'back up' from State s to one of the events in ${}^{\bullet}s$.

4.4. Resolving Conflict

In the preceding section, we looked at the relationship between the information content of an event and the information content of a single precondition (postcondition) of the event. We now look at the relationship between the information content of an event and the combined information of all the event's preconditions (postconditions).

From Corollary 4.1, we have the following

Property 4.1: $\forall e \in E$:

$I(^{\bullet}e) \subseteq I(e)$ and $I(e^{\bullet}) \subseteq I(e)$

'The information content of an event contains the combined information content of the event's preconditions (postconditions).'

The concepts of 'information gain' and 'information loss' at an event follow naturally.

Definition: For $e \in E$,

$I^{+}(e) = I(e) - I(^{\bullet}e)$ (a)

$I^{-}(e) = I(e) - I(e^{\bullet})$ (b)

$I^{+}(e)$ is the information gain at Event e.

$I^{-}(e)$ is the information loss at Event e.

'The information gain (loss) at Event e is the information content of e minus the combined information content of e's preconditions (postconditions).'

In Tables 4.1-4.3, we list the information gains and losses for the events in Figures 4.1-4.3. Note that these tables are entirely consistent with our remarks in Section 4.2

We can think of information gain as information that enters a system from the system environment, and we can think of information loss as information passed from the system to the system environment. The significance of information gain and information loss lies in their relationship to conflict.

The term 'conflict' has been applied to the situation in which two events are concurrently enabled and have a common precondition. Such a situation is shown in Figure 4.7. But for the class of structures we're dealing with, (forwards) conflict can arise only when two events have the same set of preconditions. (This is called free choice.) The reasons for this are as follows. If two

	$I^+(e)$	$I^-(e)$
1	{V}	∅
2	{R}	∅
3	∅	∅
4	∅	∅
5	∅	∅
6	∅	∅
7	∅	{V}
8	∅	{R}

Table 4.1
Bit Pipeline - Information Gains and Losses

	$I^+(e)$	$I^-(e)$
1	∅	∅
2	∅	∅
3	∅	∅
4	∅	∅
5	∅	∅
6	∅	∅
7	∅	∅
8	∅	∅

Table 4.2
Circulating Bit Pipeline - Information Gains and Losses

e	$I^+(e)$	$I^-(e)$
1	{O,V}	∅
2	{G,R}	∅
3	{R,V}	∅
4	{G,O}	∅
5	∅	∅
6	∅	{O}
7	∅	{R}
8	∅	∅
9	∅	∅
10	∅	{R,O}
11	∅	{G,V}
12	∅	{V}
13	∅	{G,R,O}

Table 4.3
Half Adder -
Information Gains
and Losses

events have a common precondition, then they must belong to the same meeting. By theorem 3.4, each event selects exactly one state from each input link of that meeting. Because each link is contained within a 1-token state component (a part), no two states in the same link can hold concurrently. It follows that if the two events are to be enabled concurrently, then they must have the same preconditions. As a result, the situation depicted in Figure 4.7 cannot arise. However, the situation in Figure 4.8 can. It should be noted that everything we've said applies not only to forwards conflict, but to backwards conflict, as well.

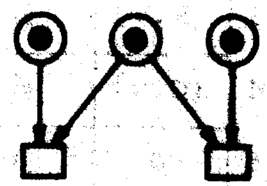


Figure 4.7

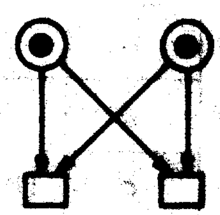


Figure 4.8

Since in our theory forwards and backwards conflict coincides with certain structural configurations, we might as well define 'conflict' in structural terms.

Definition: For $e_1, e_2 \in E$,

$$e_1 \chi^+ e_2 \Leftrightarrow {}^\bullet e_1 = {}^\bullet e_2$$

$$e_1 \chi^- e_2 \Leftrightarrow e_1{}^\bullet = e_2{}^\bullet$$

We say that e_1 and e_2 are in forwards conflict iff $e_1\chi^+e_2$ and $e_1 \neq e_2$. We say that e_1 and e_2 are in backwards conflict iff $e_1\chi^-e_2$ and $e_1 \neq e_2$. (We do not say that an event is in conflict with itself.)

Property 4.2: χ^+ and χ^- are equivalence relations on E.

Definition: The equivalence classes of χ^+ are called forwards conflict classes. The equivalence classes of χ^- are called backwards conflict classes.

Property 4.3: $\forall e \in E$,

$$[e]_{\chi^+} \subseteq [e]_\alpha \qquad \text{(a)}$$

$$[e]_{\chi^-} \subseteq [e]_\alpha \qquad \text{(b)}$$

'The forwards (backwards) conflict class associated with e is contained within the alternative class associated with e.'

In Tables 4.4-4.6, we list the forwards and backwards conflict classes for the system nets in Figures 4.1-4.3.

(a) Forwards	(b) Backwards
{1,2}	{1}
{3}	{2}
{4}	{3}
{5}	{4}
{6}	{5}
{7}	{6}
{8}	{7,8}

Table 4.4 Bit Pipeline - Conflict Classes

(a) Forwards	(b) Backwards
{1}	{1}
{2}	{2}
{3}	{3}
{4}	{4}
{5}	{5}
{6}	{6}
{7}	{7}
{8}	{8}

Table 4.5 Circulating Bit Pipeline - Conflict Classes

(a) Forwards	(b) Backwards
{1,2}	{1}
{3,4}	{2}
{5}	{3}
{6}	{4}
{7}	{5}
{8}	{6,7}
{9}	{8}
{10}	{9}
{11}	{10,11}
{12}	{12,13}
{13}	

Table 4.6 Half Adder - Conflict Classes

We now establish the relationship between information gain and forwards conflict classes, and the relationship between information loss and backwards conflict classes. A simple lemma precedes the theorem.

Lemma 4.1 $\forall e \in E$,

$$[e]_{\chi^+} = \bigcap_{s \in {}^\bullet e} s^\bullet \qquad \text{(a)}$$

'$[e]_{\chi^+}$ consists of those events whose preconditions contain ${}^\bullet e$.'

$$[e]_{\chi^-} = \bigcap_{s \in e^\bullet} {}^\bullet s \qquad \text{(b)}$$

'$[e]_{\chi^-}$ consists of those events whose postconditions contain $e^\bullet$.'

Proof: We prove Part (a).

$$\bigcap_{s \in {}^\bullet e} s^\bullet = \{a \in E \mid \forall s \in {}^\bullet e:\ s \in {}^\bullet a\}$$

$$= \{a \in E \mid \forall s \in S:\ s \in {}^\bullet e \Rightarrow s \in {}^\bullet a\}$$

$$= \{a \in E \mid {}^\bullet e \subseteq {}^\bullet a\}$$

$$= \{a \in E \mid {}^\bullet e = {}^\bullet a\} \qquad \text{Theorem 3.4(a)\&(b)}$$

$$= [e]_{\chi^+} \qquad \text{definition}$$

□

Theorem 4.3: $\forall e \in E$:

$$I^{+}(e) = \mathfrak{M}([e]_{\chi}+ - \{e\}) \tag{a}$$

$$I^{-}(e) = \mathfrak{M}([e]_{\chi}- - \{e\}) \tag{b}$$

'The information gained (lost) at Event e is equal to the set of modes covering those events in forwards (backwards) conflict with e.'

Proof: We prove Part (a).

$I^{+}(e)$	$= I(e) - I({}^{\bullet}e)$	definition
	$= I(e) - \bigcup_{s \in {}^{\bullet}e} I(s)$	definition
	$= \bigcap_{s \in {}^{\bullet}e} (I(e) - I(s))$	DeMorgans's Law
	$= \bigcap_{s \in {}^{\bullet}e} \mathfrak{M}(s^{\bullet} - \{e\})$	Theorem 4.2(a)
	$= \mathfrak{M}(\bigcap_{s \in {}^{\bullet}e} (s^{\bullet} - \{e\}))$	Theorem 3.5
	$= \mathfrak{M}((\bigcap_{s \in {}^{\bullet}e} s^{\bullet}) - \{e\})$	Boolean Algebra
	$= \mathfrak{M}([e]_{\chi}+ - \{e\})$	Lemma 4.1(a)

□

The reader may verify this theorem by comparing the information gains and losses in Tables 4.1-4.3 with the forwards and backwards conflict classes of Tables 4.4-4.6. For example, in Table 4.1, we see that the information gain of Event 1 in the bit pipeline is {V}. In Table 4.4, we see that Event 1 is in forwards conflict with Event 2. The set of modes covering Event 2 is {V}. It checks.

We note that when an event is not in forwards (backwards) conflict with any other event, its information (loss) is null. Thus, information is gained (lost) at precisely those points where there is forwards (backwards) conflict. Furthermore, the information gained or lost in a conflict situation specifies how the conflict is resolved. This is because selecting an event in a conflict class is equivalent to specifying the modes covering the remaining events - from either one we can derive the other.

CHAPTER 5

CONTROL

5.1. Event Graphs:

In Chapter 3, we showed that each system simulation is isomorphic to a simulation of the control structure. Since the control structure is an event graph, any results we obtain for event-graph simulations can be applied to system simulations. This is fortunate because event-graph simulations have some very nice properties. Those properties are the subject of this chapter.

Before getting to the results, we must introduce some notation and terminology.

Definition: For a directed Graph G, $\Pi(G)$ denotes the paths of G.[†]

Definition: For a path μ in a directed graph,

$^{\bullet}\mu$ denotes the initial endpoint (tail) of μ
$\mu^{\bullet}$ denotes the terminal endpoint (head) of μ

Definition: If μ is a path in the directed graph G, and x is a vertex of G, then,

$x \in \mu$ iff x appears in μ

Definition: For paths μ_1 and μ_2 in a directed graph,

$\mu_1 \subseteq \mu_2$ iff μ_1 is a subpath of μ_2

[†] We're repeating the definition of $\Pi(G)$ given in Chapter 2.

Definition: If μ is a path in the directed graph G, and A is a set of vertices of G, then,

$|\mu|_A$ denotes the number of times an element of A appears in μ.

If <N,I> is an initialized event graph, and μ is a path in N, then $|\mu|_I$ is called the token loading on μ.

Definition: A circuit is a path whose two endpoints are the same. An elementary circuit is a circuit in which no vertex is encountered more than once. For a directed graph G,

$\Omega(G)$ denotes the circuits of G

$\Omega^e(g)$ denotes the elementary circuits of G

Definition: In an initialized event graph, a blank circuit is a circuit containing no initial conditions (no 'tokens'). A basic circuit is an elementary circuit containing exactly one initial condition (exactly one 'token').

5.2. Paths:

Many of the ideas in this chapter relate to paths, the paths in event graphs and the paths in event-graph simulations. We begin with basic circuits.

Property 5.1: In an initialized event graph, the basic circuits and the I-token state components coincide.

This can be seen from the following.

(a) In an event graph, each state has exactly one incident arc and one emergent arc.

(b) In a state component, each event has exactly one incident arc and one emergent arc.

(c) A state component is connected.

When an initialized event graph is covered by basic circuits, we know from Corollary 2.3 that in each simulation of the event graph all instances of the same element are totally ordered. We also have the following lemma.

Lemma 5.1: If Z is an initialized event graph covered by basic circuits,
I is the set of initial conditions of Z,
C is the causality relation for a simulation of Z,

then for $<<x_1,n_1>,<x_2,n_2>> \in C$,

$$n_2=n_1 \Leftrightarrow x_2 \notin I$$
$$n_2=n_1+1 \Leftrightarrow x_2 \in I$$

'If there is an elementary causal connection leading from instance $<x_1,n_1>$ to instance $<x_2,n_2>$, then $n_2=n_1$ for $x_2 \notin I$ and $n_2=n_1+1$ for $x_2 \in I$.'

Proof: From Theorem 2.2 we know that $<x_1,x_2>$ must be an arc in the event graph for Z, and therefore, must be contained in a basic circuit of Z. The image of that basic circuit in the simulation associated with C is a single strand (Corollary 2.1) in which $<x_1,n_1>$ and $<x_2,n_2>$ are consecutive instances. This strand traces out a path around the basic circuit beginning at the unique initial condition of the basic circuit. Since the instances of each element are numbered consecutively, the number of an instance in the strand indicates which cycle the instance appears in. The lemma follows from the fact that each cycle begins with a holding of the initial condition associated with the basic circuit. □

In Figure 5.1 is an initialized event graph covered by basic circuits. In Figure 5.2 is a simulation of that event graph in which each holding of an initial condition is indicated by a dashed circle. Notice that instance numbers are incremented by 1 at each holding of an initial condition, and that they remain the same at all other instances.

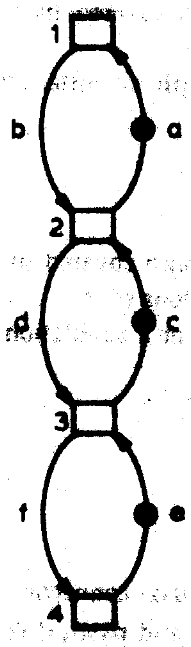


Figure 5.1 Initialized Event Graph Covered by Basic Circuits

The following theorem is a direct consequence of Lemma 5.1.

Theorem 5.1: If Z is an initialized event graph covered by basic circuits,
I is the set of initial conditions of Z,
T is a simulation of Z, then,

$$\forall \sigma_1,\sigma_2 \in \Pi(T):\ {}^\bullet\sigma_1 = {}^\bullet\sigma_2 \wedge \sigma_1^\bullet = \sigma_2^\bullet \Rightarrow |\rho_1|_I = |\rho_2|_I$$

'If σ_2 and σ_2 are paths (causal connections) in T having the same endpoints, then the token loadings on their images are the same.'

Proof: Let ${}^\bullet\sigma_1 = {}^\bullet\sigma_2 = \langle x,m\rangle$ and $\sigma_1^\bullet = \sigma_2^\bullet = \langle y,n\rangle$. By Lemma 5.1, the number of holdings of initial conditions crossed by both σ_1 and σ_2 is n-m for $x \notin I$ and n-m+1 for $x \in I$. In either case $|\rho_1|_I = |\rho_2|_I$. □

As an illustration of Theorem 5.1, consider the following two paths in Figure 5.2,

$\sigma_1 = \langle 2,1\rangle\langle a,2\rangle\langle 1,2\rangle\langle b,2\rangle\langle 2,2\rangle\langle a,3\rangle\langle 1,3\rangle\langle b,3\rangle\langle 2,3\rangle\langle d,3\rangle\langle 3,3\rangle$

$\sigma_2 = \langle 2,1\rangle\langle d,1\rangle\langle 3,1\rangle\langle f,1\rangle\langle 4,1\rangle\langle e,2\rangle\langle 3,2\rangle\langle f,2\rangle\langle 4,2\rangle\langle e,3\rangle\langle 3,3\rangle$

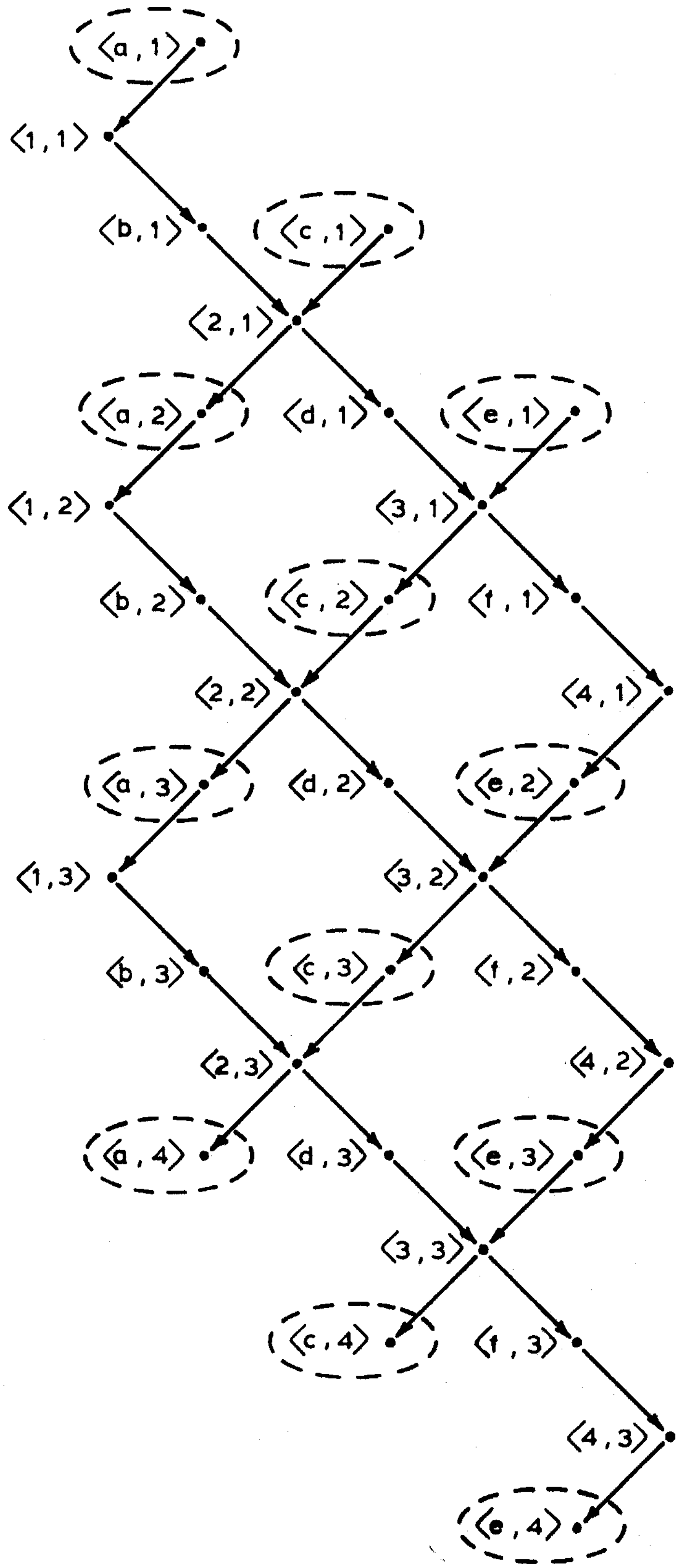


Figure 5.2 Holdings of Initial Conditions

Since σ_1 and σ_2 have the same endpoints, their images should have the same token loadings. Let's see.

$\hat{\sigma}_1$=2$\underline{a}$1b2$\underline{a}$1b2d3 and $|\hat{\sigma}_1|_I=2$

$\hat{\sigma}_2$=2d3f4$\underline{e}$3f4$\underline{e}$3 and $|\hat{\sigma}_2|_I=2$

It checks.

For an initialized event graph covered by basic circuits, the next theorem establishes a special relationship between the paths of the event graph and the paths of a corresponding simulation.

Theorem 5.2: If $Z=\langle N,I\rangle$ is an initialized event graph covered by basic circuits, and T is a simulation of Z,

then $\forall\mu\in\Pi(N)$: $\forall\sigma\in\Pi(T)$:

$$^{\bullet}\mu={}^{\bullet}\hat{\sigma} \;\wedge\; \mu^{\bullet}=\hat{\sigma}^{\bullet} \;\wedge\; |\mu|_I=|\hat{\sigma}|_I \;\Rightarrow\; E\zeta\in\Pi(T){:}\; {}^{\bullet}\zeta={}^{\bullet}\sigma \;\wedge\; \zeta^{\bullet}=\sigma^{\bullet} \;\wedge\; \hat{\zeta}=\mu.$$

'If μ is a path in N and σ is a path in T, and if the endpoints of μ and $\hat{\sigma}$ are the same and μ and $\hat{\sigma}$ have the same token loadings, then there exists a path in T having the same endpoints as σ and whose image is μ.'

Proof: Let ${}^{\bullet}\sigma=\langle x_1,n_1\rangle$ and $\sigma^{\bullet}=\langle x_2,n_2\rangle$. The required path ζ can be constructed by backtracking from $\langle x_2,n_2\rangle$. Property 2.3 guarantees that at each step there will be a way of extending the path in accordance with μ. There's one exception though, and that's when a holding in the back boundary of T is reached. By Lemma 5.1, when a holding in the back boundary is reached, the token loading on the path already generated is n_2. There are two cases to consider: (1) $\langle x_1,n_1\rangle$ in the back boundary of T and (2) $\langle x_1,n_1\rangle$ not in the back boundary of T. In the first case, $|\hat{\sigma}|_I=n_2$ by Lemma 5.1, and, therefore, the path must be complete (otherwise, we would have $|\mu|_I>n_2$ and $|\mu|_I\neq|\hat{\sigma}|_I$). In the second case, $|\hat{\sigma}|_I<n_2$ by Lemma 5.1, and, thus, this case cannot arise. So we've now got a path ζ such that $\zeta^{\bullet}=\langle x_2,n_2\rangle=\sigma^{\bullet}$ and $\hat{\zeta}=\mu$. Because ${}^{\bullet}\mu=x_1$, ${}^{\bullet}\zeta$ must be an instance of x_1, and because $|\mu|_I=|\hat{\sigma}|_I$, $|\hat{\zeta}|_I=|\hat{\sigma}|_I$. From Lemma 5.1 and the three facts (1) ${}^{\bullet}\zeta={}^{\bullet}\sigma$, (2) $|\hat{\zeta}|_I=|\hat{\sigma}|_I$, and (3) ${}^{\bullet}\hat{\zeta}={}^{\bullet}\hat{\sigma}$, it follows that ${}^{\bullet}\zeta={}^{\bullet}\sigma$. □

To illustrate Theorem 5.2, we consider the following path μ in the event graph of Figure 5.1 and the following path σ in the simulation of Figure 5.2

μ = 2a1b2a1b2d3

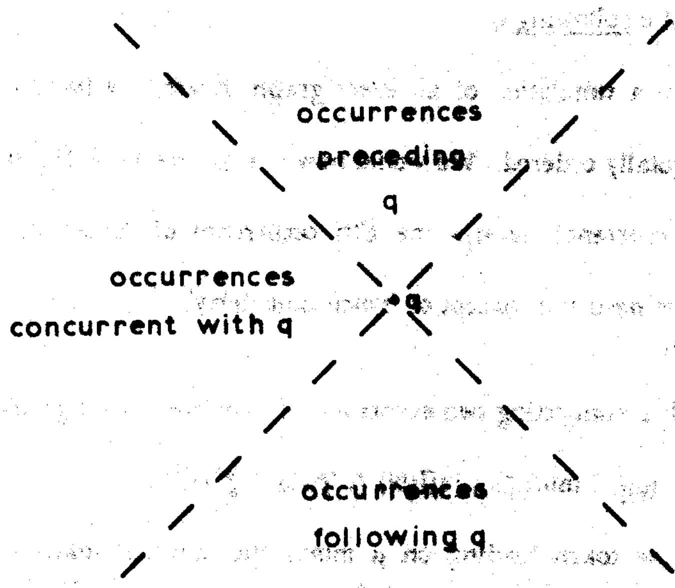
$\sigma = \langle 2,1\rangle\langle d,1\rangle\langle 3,1\rangle\langle f,1\rangle\langle 4,1\rangle\langle e,2\rangle\langle 3,2\rangle\langle f,2\rangle\langle 4,2\rangle\langle a,3\rangle\langle 3,3\rangle$

We have ${}^{\bullet}\mu = {}^{\bullet}\sigma = 2$, $\mu^{\bullet} = \sigma^{\bullet} = 3$, and $|\mu|_1 = |\sigma|_1 = 2$. Therefore, there should be a path ζ in the simulation of Figure 5.2 having the same endpoints as σ and such that $\hat{\zeta} = \mu$. There is.

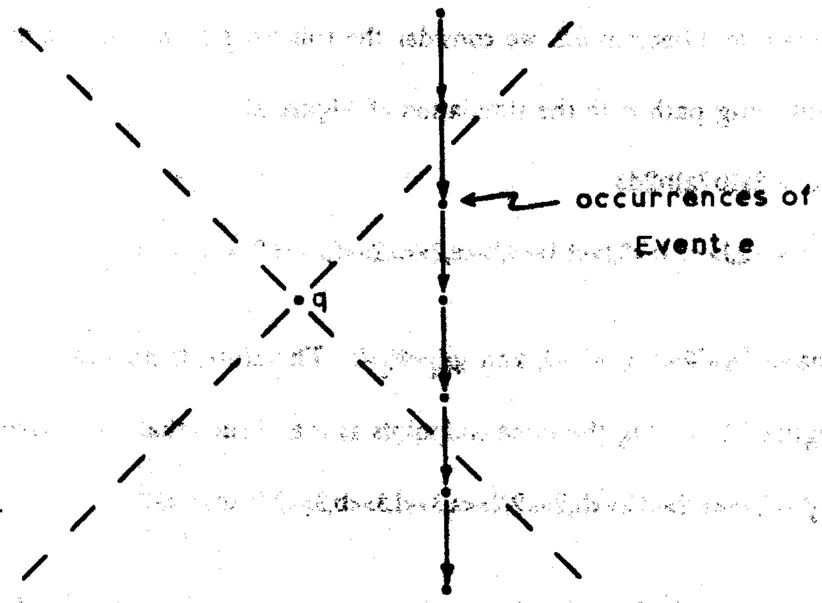
$\zeta = \langle 2,1\rangle\langle a,2\rangle\langle 1,2\rangle\langle b,2\rangle\langle 2,2\rangle\langle a,3\rangle\langle 1,3\rangle\langle b,3\rangle\langle 2,3\rangle\langle d,3\rangle\langle 3,3\rangle$

We now look at the circumstances surrounding the ordering of two occurrences in an event-graph simulation. The first part of our discussion is applicable to all Petri-net simulations.

Suppose that q is an occurrence in a simulation. Then we can separate the other occurrences into three categories: (1) those that precede q, (2) those that follow q, and (3) those that are concurrent with q.



Suppose that the occurrences of Event e are totally ordered in the simulation with q.



Now if r is an occurrence of Event e preceding q, then there exists a positive integer k such that 'r is the k'th occurrence of Event e preceding q.' For example, r might be the 'second-to-last' occurrence of Event e preceding q. In Figure 5.2, $\langle 1,1\rangle$ is the third-to-last occurrence of Event 1 preceding $\langle 3,3\rangle$, while $\langle 1,3\rangle$ is the last occurrence of Event 1 preceding $\langle 3,3\rangle$. Similar remarks apply to occurrences of Event e <u>following</u> q.

We know that in a simulation of an event graph covered by basic circuits, all instances of the same element are totally ordered. We would now like to determine for such a simulation under what conditions is Occurrence $\langle e_1,n_1\rangle$ the k'th occurrence of Event e_1 preceding Occurrence $\langle e_2,n_2\rangle$. To do that, we need the concept of 'synchronic delay'.

Definition: For a path μ connecting two events in the initialized event graph $Z=\langle N,I\rangle$,

$$\delta_Z(\mu) \;=\; |\mu|_I - \min\{|\zeta|_I \mid \zeta\in\Pi(N) \wedge {}^\bullet\zeta={}^\bullet\mu \wedge \zeta^\bullet=\mu^\bullet\}$$

'$\delta_Z(\mu)$ is the token loading on μ minus the minimal token loading on those paths having the same endpoints as μ.' $\delta_Z(\mu)$ is the <u>synchronic delay</u> of μ (with respect to Z). (When Z is understood, we shall write the synchronic delay of μ as just $\delta(\mu)$.)

We give the synchronic delays for several of the paths in the event graph of Figure 5.1.

σ_1 = 1b2d3f4	$\delta(\sigma_1)$ = 0-0 = 0
σ_2 = 4e3c2a1	$\delta(\sigma_2)$ = 3-3 = 0
σ_3 = 2d3c2a1	$\delta(\sigma_3)$ = 2-1 = 1
σ_4 = 3f4e3	$\delta(\sigma_4)$ = 1-0 = 1
σ_5 = 3c2d3f4e3	$\delta(\sigma_5)$ = 2-0 = 2

In the special case where μ is a circuit, the minimal token loading between the endpoints of μ is 0. We thus have the following property.

Property 5.2: If Z is the initialized event graph <N,I>, then,

$$\forall \omega \in \Omega(N): \quad \delta_Z(\omega) = |\omega|_I$$

'The synchronic delay of a circuit is equal to its token loading.'

The following theorem says, in effect, that the synchronic delay of a path cannot be decreased by extending the path - the synchronic delay either remains the same or increases.

Theorem 5.3: If μ_1 and μ_2 are paths connecting events in the initialized event graph Z, then,

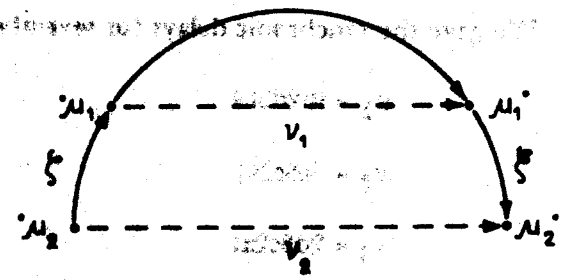
$$\mu_1 \subseteq \mu_2 \;\Rightarrow\; \delta_Z(\mu_1) \leq \delta_Z(\mu_2)$$

'If μ_1 is a subpath of μ_2, then the synchronic delay of μ_1 is less than or equal to the synchronic delay of μ_2'

Proof: Let $\mu_2 = \zeta\mu_1\xi$. Let ν_1 be a path of minimal token loading from ${}^\bullet\mu_1$ to $\mu_1{}^\bullet$. Let ν_2 be a path of minimal token loading from ${}^\bullet\mu_2$ to $\mu_2{}^\bullet$ We have,

$$\delta_Z(\mu_1) = |\mu_1|_I - |\nu_1|_I$$

and, $\delta_Z(\mu_2) = |\mu_2|_I - |\nu_2|_I$



Since $\mu_2 = \zeta\mu_1\xi$,

$$|\mu_2|_I = |\zeta|_I + |\mu_1|_I + |\xi|_I$$

Thus, $\delta_Z(\mu_2) = |\zeta|_I + |\mu_1|_I + |\xi|_I - |\nu_2|_I$ (1)

Because ν_2 is a path of minimal token loading from $^\bullet\mu_2$ to $\mu_2^\bullet$,

$$|\nu_2|_I \le |\zeta|_I + |\nu_1|_I + |\xi|_I \qquad (2)$$

Combining lines (1) and (2), we get,

$$\delta_Z(\mu_2) \ge |\mu_1| - |\nu_1|_I = \delta_Z(\mu_1)$$ □

We're now ready for the major result of this section.

Theorem 5.4: If Z is an initialized event graph covered by basic circuits,
T is a simulation of Z,
$\langle e_1,n_1\rangle$ and $\langle e_2,n_2\rangle$ are occurrences in T,

then the following are equivalent

(a) $\langle e_1,n_1\rangle$ is the k'th occurrence of e_1 preceding $\langle e_2,n_2\rangle$

(b) $\langle e_2,n_2\rangle$ is the k'th occurrence of e_2 following $\langle e_1,n_1\rangle$

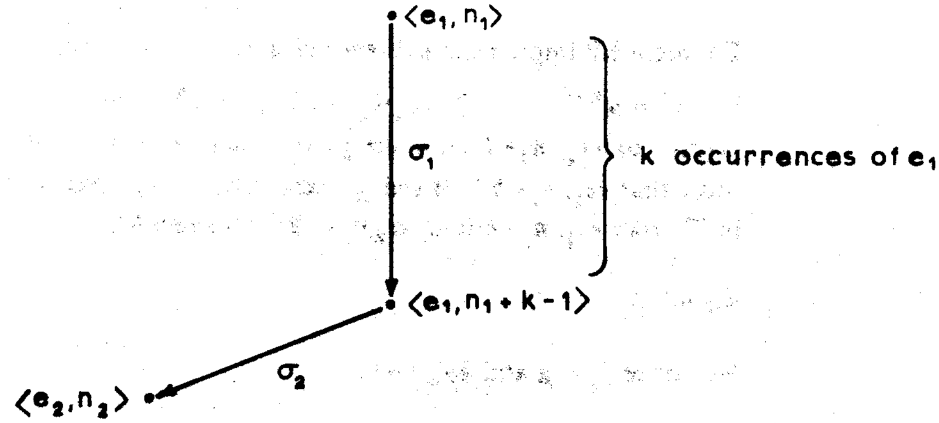
(c) $\exists\sigma\in\Pi(T)$: $^\bullet\sigma=\langle e_1,n_1\rangle \wedge \sigma^\bullet = \langle e_2,n_2\rangle \wedge \delta_Z(\sigma)=k-1$

'There exists a path from $\langle e_1,n_1\rangle$ to $\langle e_2,n_2\rangle$, and the synchrononic delay of its image is k-1.'

Proof: We prove that (a) ⇔ (c). It follows, by symmetry, that (b) ⇔ (c). Let $Z=\langle N,I\rangle$.

(a) ⇒ (c)

Since $\langle e_1,n_1\rangle$ is the k'th occurrence of e_1 preceding $\langle e_2,n_2\rangle$, $\langle e_1,n_1+k-1\rangle$ must be the last. Let σ_1 be a path from $\langle e_1,n_1\rangle$ to $\langle e_1, n_1+k-1\rangle$, and σ_2 a path from $\langle e_1, n_1+k-1\rangle$ to $\langle e_2,n_2\rangle$. (A path always exists between ordered occurrences.)



Let $\sigma = \sigma_1 \cdot \sigma_2$, and let μ be a path in N from e_1 to e_2 such that $\delta_Z(\mu)=0$. The definition of synchronic delay gives us,

$$\delta_Z(\theta) = |\theta|_T - |\mu|_T = |\theta_1|_T + |\theta_2|_T - |\mu|_T \tag{1}$$

From Lemma 5.1, we have

$$|\theta_1|_T = (n_1+k-1)-n_1 = k-1 \tag{2}$$

We now show that $|\theta_2|_T = |\mu|_T$. Since $\delta_Z(\mu)=0$, $|\mu|_T \le |\theta_2|_T$. Let $a = |\theta_2|_T - |\mu|_T$, and let ω be an elementary circuit in N beginning and ending at e_1 and having a token loading of 1. (Such a circuit exists because Z is covered by basic circuits.) Let $\mu' = \omega^a \mu$.† μ' is a path in N from e_1 to e_2.

$$|\mu'|_T = a + |\mu|_T = |\theta_2|_T$$

Since μ' and θ_2 have the same endpoints (e_1 and e_2) and $|\mu'|_T = |\theta_2|_T$, Theorem 5.2 implies the existence of a path ξ in T from $\langle e_1, n_1+k-1 \rangle$ to $\langle e_2, n_2 \rangle$ such that $\bar{\xi} = \mu'$. Because $\bar{\xi} = \omega^a \mu$ and $e_1 \in \omega$, ξ will cross a occurrences of e_1 after leaving $\langle e_1, n_1+k-1 \rangle$, and before reaching $\langle e_2, n_2 \rangle$. But $\langle e_1, n_1+k-1 \rangle$ is the <u>last</u> occurrence of e_1 preceding $\langle e_2, n_2 \rangle$. Therefore, we must conclude that $a=0$ and,

$$|\theta_2|_T = |\mu|_T \tag{3}$$

From Lines (1), (2), and (3), we have $\delta_Z(\theta) = k-1$.

(c) ⇒ (a)

Let μ and ω be as defined in the first part, and let $\mu' = \omega^{k-1}\mu$. We get,

$$|\mu'|_T = k-1+|\mu|_T = \delta(\theta)+|\mu|_T = |\theta|_T - |\mu|_T + |\mu|_T = |\theta|_T$$

---

† ω^a means that ω is repeated a times.

Theorem 5.2 implies the existence of a path ρ in T from $<e_1,n_1>$ to $<e_2,n_2>$ such that $\rho = \mu' = \omega^{k-1}\mu$. Let $\rho = \rho_1 \cdot \rho_2$ where $\rho_1 = \omega^{k-1}$ and $\rho_2 = \mu$. ρ_1 is a path in T from $<e_1,n_1>$ to $<e_1, n_1+k-1>$, while ρ_2 is a path in T from $<e_1, n_1+k-1>$ to $<e_2,n_2>$. To show that $<e_1, n_1+k-1>$ is the __last__ occurrence of e_1 preceding $<e_2,n_2>$, let ξ be any path in T from $<e_1, n_1+k-1>$ to $<e_2,n_2>$. By Theorem 5.1,

$$|\xi|_T = |\rho_2|_T$$

But since $\rho_2 = \mu$ and $\delta_Z(\mu) = 0$,

$$\delta_Z(\xi) = 0 \qquad (1)$$

Suppose that ξ contains an occurrence of e_1 after $<e_1, n_1+k-1>$. Let ξ_1 be the part of ξ preceding that occurrence. ξ_1 is a circuit in N (beginning and ending at e_1). By Lemma 5.1, $|\xi_1|_T > 0$. In other words, ξ contains a circuit with a token loading greater than 0. But this is inconsistent with Line (1). We must conclude that no path in T between $<e_1, n_1+k-1>$ and $<e_2, n_2>$ contains an occurrence of e_1 after $<e_1, n_1+k-1>$. Thus, $<e_1, n_1+k-1>$ is the last occurrence of e_1 preceding $<e_2,n_2>$, and $<e_1,n_1>$ is the k'th occurrence of e_1 preceding $<e_2,n_2>$. □

The equivalence of Statements (a) and (b) in Theorem 5.4 can be visualized as in Figure 5.3

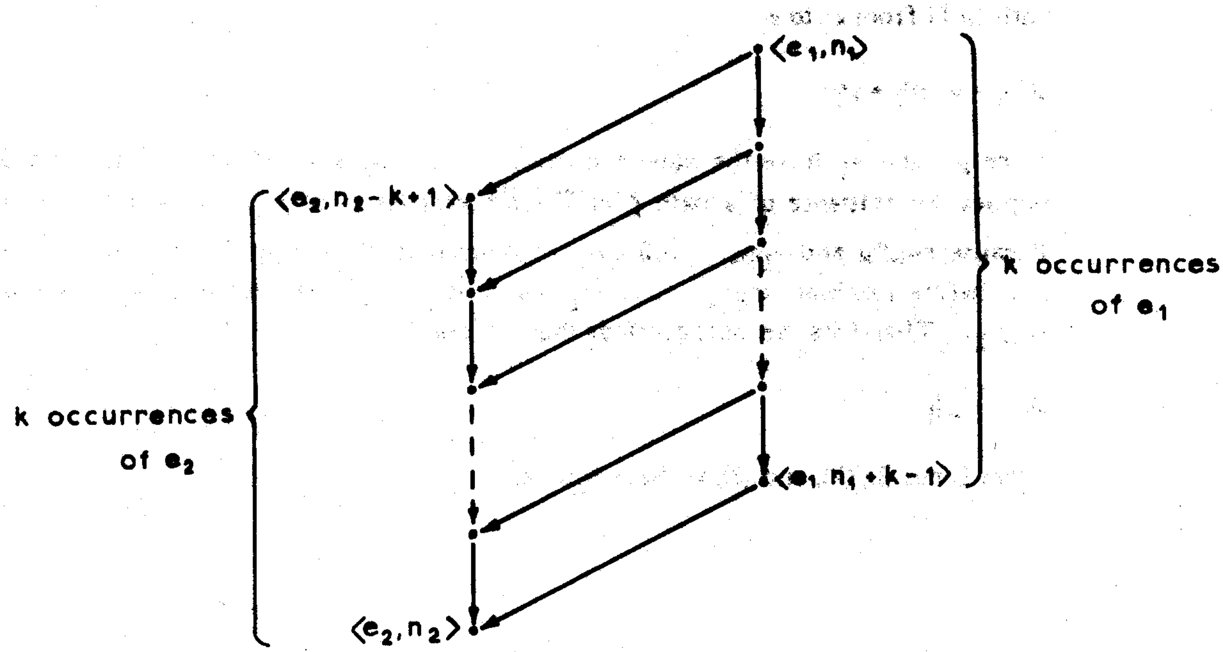


Figure 5.3 Ordered Occurrences in an Event-Graph Simulation

In the simulation of Figure 5.2, we see that <2,1> is the <u>second</u> occurrence of Event 2 <u>preceding</u> Occurrence <1,3> and that <1,3> is the <u>second</u> occurrence of Event 1 <u>following</u> <2,1>. We have the following path σ connecting <2,1> and <1,3>.

$$\sigma = \langle 2,1\rangle\langle d,1\rangle\langle 3,1\rangle\langle c,2\rangle\langle 2,2\rangle\langle a,3\rangle\langle 1,3\rangle$$

$$\hat{\sigma} = 2d3c2a1 \text{ and } \delta(\hat{\sigma})=1$$

The theorem checks out.

5.3. <u>Cones</u>:

Because synchronic delay is not a convenient concept to work with, we introduce the concepts of 'back cone' and 'front cone'.

Definition: If Z is an initialized event graph,
N is the event graph associated with Z,
S is the set of states of N,
e is an event in N, then,

$$\phi_Z^-(e) = \{s\in S \mid \nexists\mu\in\Pi(N):\ {}^\bullet\mu = s \wedge s\in\mu \wedge \mu^\bullet = e \wedge \delta(\mu)=0\}$$

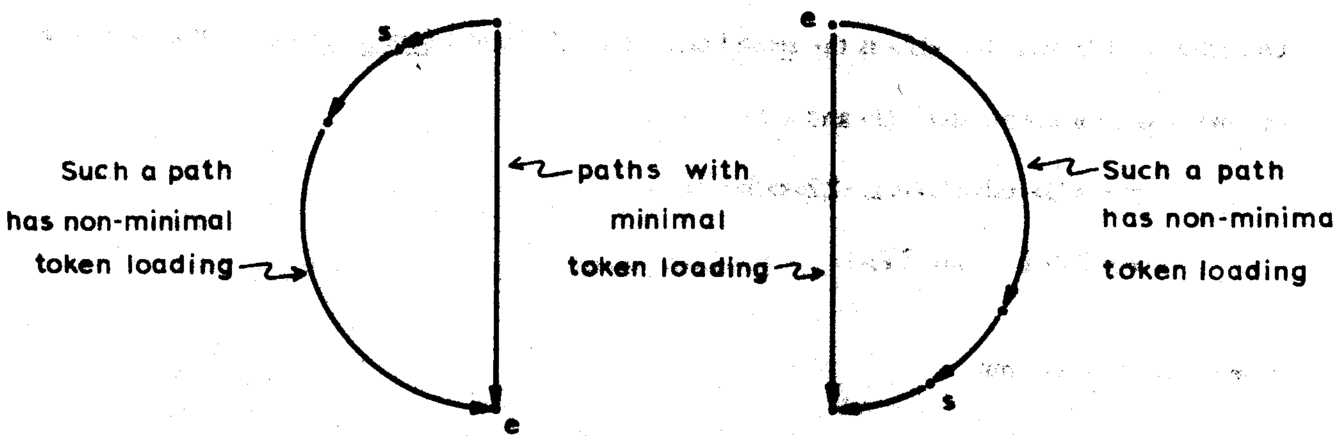
'$\phi_Z^-(e)$ is the set of states s such that there does not exist a path of delay zero beginning at the input event of s, passing through s, and terminating at e.'

$$\phi_Z^+(e) = \{s\in S \mid \nexists\mu\in\Pi(N):\ {}^\bullet\mu = e \wedge s\in\mu \wedge s^\bullet = \mu^\bullet \wedge \delta(\mu)=0\}$$

'$\phi_Z^+(e)$ is the set of states s such that there does not exist a path of delay zero beginning at e, passing through s, and terminating at the output event of s.'

$\phi_Z^-(e)$ is the <u>back cone</u> of e_1 and $\phi_Z^+(e)$ the <u>front cone</u> of e. (When Z is understood, we shall omit it as a subscript of ϕ.)

The two definitions are illustrated in Figure 5.4. In effect, what $s\in\phi_Z^-(e)$ means is that the



(a) $s \in \phi_Z^-(e)$ (b) $s \in \phi_Z^+(e)$

Figure 5.4 Paths Associated with Front and Back Cones

	a	b	c	d	e	f
$\phi^-(1)$	0	1	0	1	0	1
$\phi^-(2)$	1	0	0	1	0	1
$\phi^-(3)$	1	0	1	0	0	1
$\phi^-(4)$	1	0	1	0	1	0

(a) Back Cones

	a	b	c	d	e	f
$\phi^+(1)$	1	0	1	0	1	0
$\phi^+(2)$	0	1	1	0	1	0
$\phi^+(3)$	0	1	0	1	1	0
$\phi^+(4)$	0	1	0	1	0	1

(b) Front Cones

Table 5.1 Characteristic Functions for Front and Back Cones

'quickest' way from the input event of s to e is <u>not</u> through s. Similarly, what $s \in \phi_Z^+(e)$ means is that the 'quickest' way from e to the output event of s is <u>not</u> through s. In Tables 5.1(a) and 5.1(b) we give the characteristic functions for the front and back cones of the events in Figure 5.1. Note that in our example, $\phi_Z^-(e)$ is the complement of $\phi_Z^+(e)$ for each event e. This is not generally the case.

The significance of front and back cones is best understood in terms of simulations. (The first part of our discussion applies to all simulations, not just event-graph simulations). Suppose that q is an occurrence in a simulation. The occurrences in that simulation can be separated into two categories: (1) those that precede or are equal to q and (2) all others. With respect to q, these two sets form, respectively, the past and the 'not past'. Now between the two sets of occurrences there is a boundary, and this boundary is associated with a set of holdings. These holdings have the property of not preceding q but of being initiated by occurrences that do. This is illustrated in Figure 5.5. If we imagine the simulation to be three-dimensional, then the boundary resembles the surface of a cone. Similar remarks apply to the boundary between the future and the 'not future' with respect to q. In this case, the holdings making up the boundary have the property of not following q but of being terminated by occurrences that do. This is illustrated in Figure 5.6. In Figures 5.7 and 5.8 is a simulation of the event graph in Figure 5.1. We've indicated the 'back cones' and 'front cones' for occurrences of Event 3. Notice that the simulation is 'sliced up' by the 'cones' of each type.

The reader has undoubtedly noticed that we've used the term 'cone' to describe both sets of states and sets of holdings. The correspondence between the two views is straightforward: If q is an occurrence of Event e, then the holdings in the 'back cone' of q are holdings of states in $\phi_Z^-(e)$

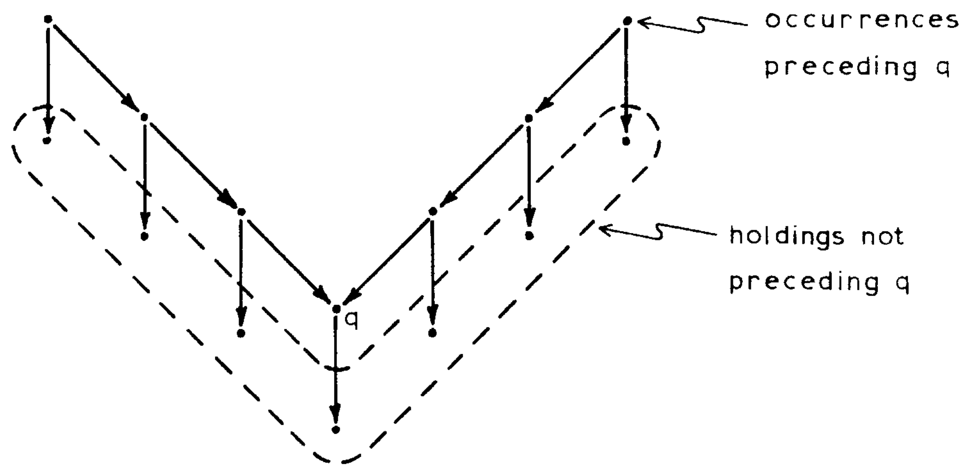


Figure 5.5 Boundary between Past and 'Not Past'

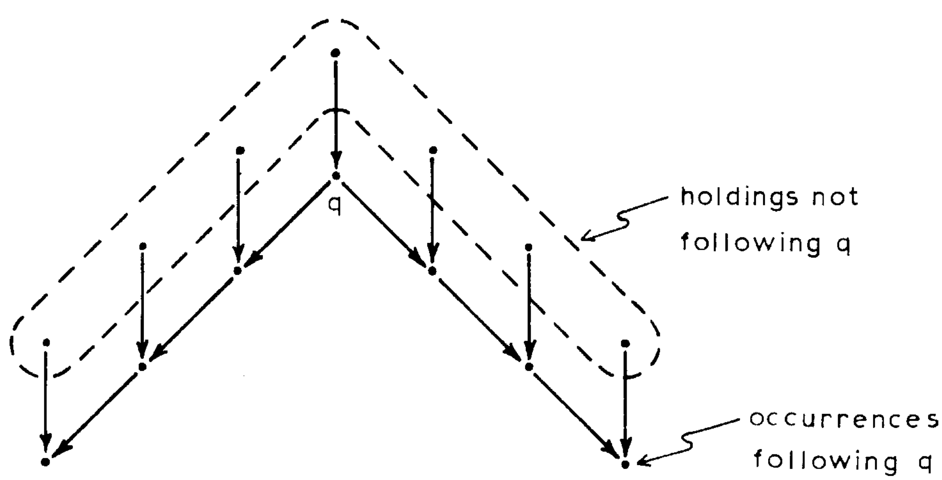


Figure 5.6 Boundary between Future and 'Not Future'

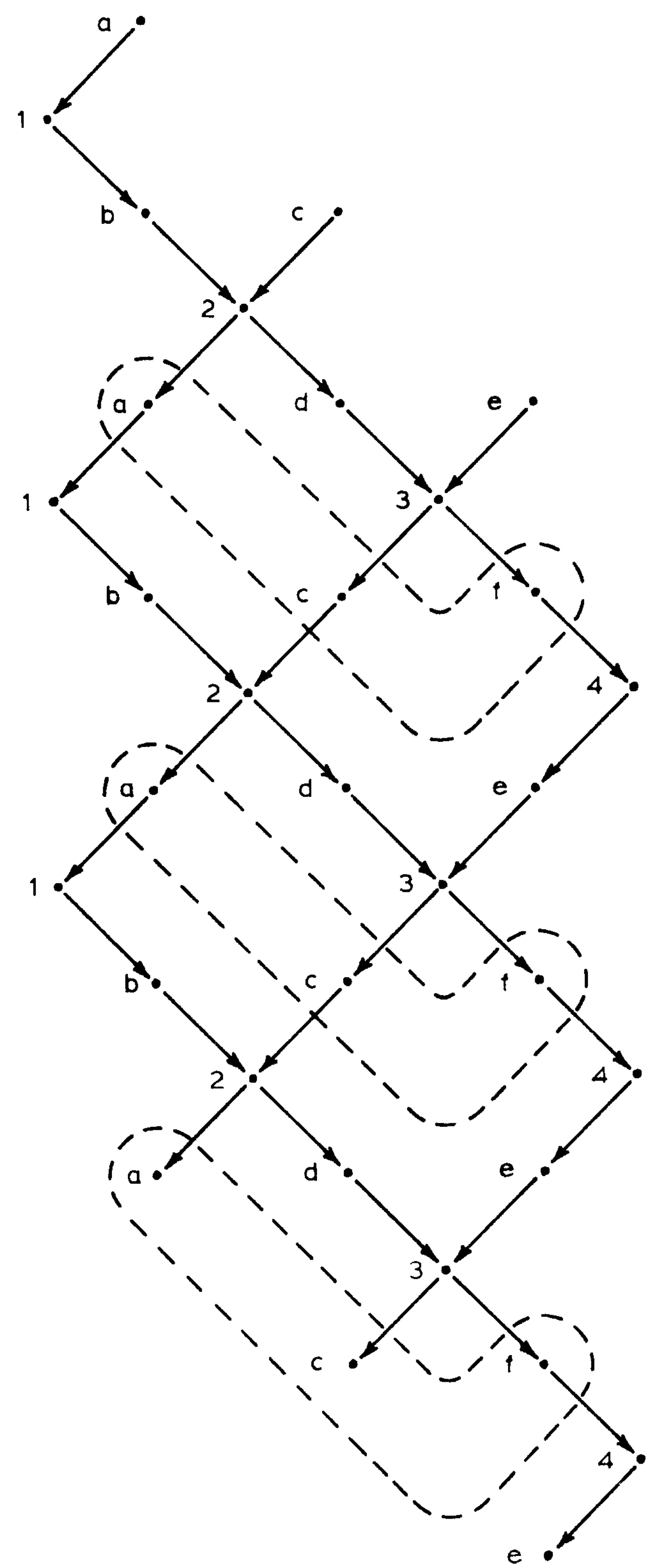


Figure 5.7 'Back Cones' for Occurrences of Event 3

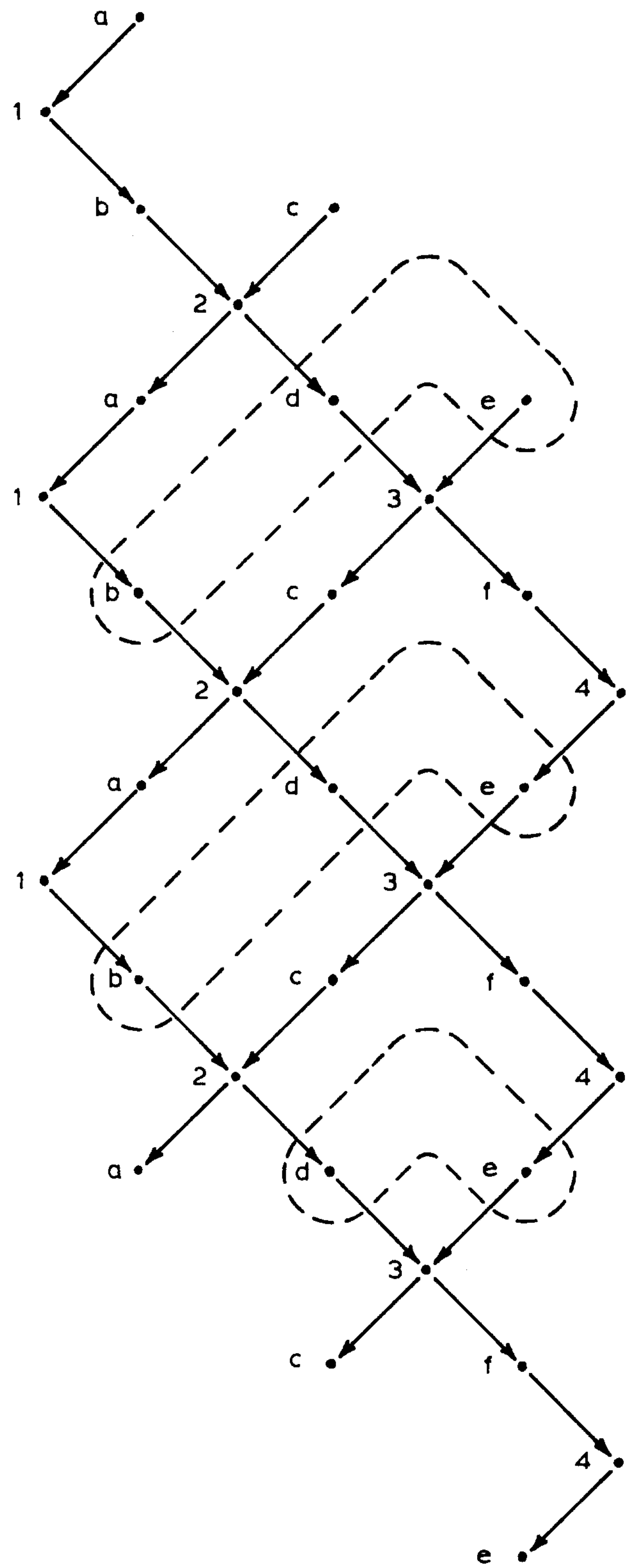


Figure 5.8 'Front Cones' for Occurrences of Event 3

and the holdings in the 'front cone' of q are holdings of states in $\phi_Z^+(e)$. Thus, we see that in Figure 5.7 each holding in the 'back cone' of an occurrence of Event 3 is a holding of a state in $\phi^-(3)=\{a,c,f\}$. Similarly, we see that in Figure 5.8 each holding in the 'front cone' of an occurrence of Event 3 is a holding of a state in $\phi^+(3)=\{b,d,e\}$. These ideas are expressed in the following theorem.

Theorem 5.5: If Z is an initialized event graph covered by basic circuits, and <H,O,C> is a simulation of Z, then for $\langle s,m\rangle \in H$ and $\langle e,n\rangle \in O$,

$$\langle s,m\rangle \not\leq \langle e,n\rangle \wedge (\exists q \in O:\ q \cdot \langle s,m\rangle \wedge q \leq \langle e,n\rangle) \Rightarrow s \in \phi_Z^-(e) \quad \text{(a)}$$

$$\langle e,n\rangle \not\leq \langle s,m< \wedge (\exists q \in O:\ \langle s,m\rangle \cdot q \wedge \langle e,n\rangle \leq q) \Rightarrow s \in \phi_Z^+(e) \quad \text{(b)}$$

'If <s,m> doesn't precede (follow) <e,n> but is initiated (terminated) by an occurrence that does, then s must be in the back (front) cone of e.

Proof: We prove just the first half, the second half being symmetrical.

Let $q = \langle e',n'\rangle$. Then q must necessarily be the last occurrence of e' preceding $\langle e,n\rangle$, because any later occurrence of e' would have to follow $\langle s,m\rangle$. ($\langle s,m\rangle$ must be terminated before another holding of s is initiated.) By Theorem 5.4, there exists a path σ from q to $\langle e,n\rangle$ such that $\delta_Z(\theta)=0$. Now, if there existed a path μ in the event graph such that ${}^\bullet\mu=e' \wedge$ $s \in \mu \wedge \mu^\bullet = e \wedge \delta(\mu)=0$, then by Theorem 5.2 there would have to be a path in the simulation beginning at q, containing a holding of s, and ending at $\langle e,n\rangle$. But this contradicts the premise that $\langle s,m\rangle \not\leq \langle e,n\rangle$. Therefore, neither the path in the simulation nor the path in the event graph exists. An so $s \in \phi_Z^-(e)$. □

We now relate the concept of cones to the ideas in the preceding section. Consider the following observation about the simulation in Figure 5.7: If q is an occurrence of Event 3, and if σ is a path originating at an occurrence $\langle e,n\rangle$ and terminating at q, then $\langle e,n\rangle$ is the k'th occurrence of e preceding q iff σ crosses k-1 'back cones'. For example, <1,1> is the fourth-to-last occurrence of Event 1 preceding <3,4>, and each path between the two occurrences crosses exactly 3 'back cones'.

A similar observation applies to the 'front cones' in Figure 5.8. If q is an occurrence of Event 3, and if σ is a path originating at q and terminating at an occurrence $<e,n>$, then $<e,n>$ is the k'th occurrence of e following q iff σ crosses k-1 'front cones'. These ideas are reflected in the following theorem and corollary.

Theorem 5.6: If Z is an initialized event graph covered by basic circuits and μ is a path in Z, then,

$$\delta_Z(\mu) = |\mu|_{\phi_Z^-(\mu^\circ)} \qquad (a)$$

$$\delta_Z(\mu) = |\mu|_{\phi_Z^+({}^\circ\mu)} \qquad (b)$$

'The synchronic delay of μ is equal to the number of times the back (front) cone of μ's head (tail) is crossed.'

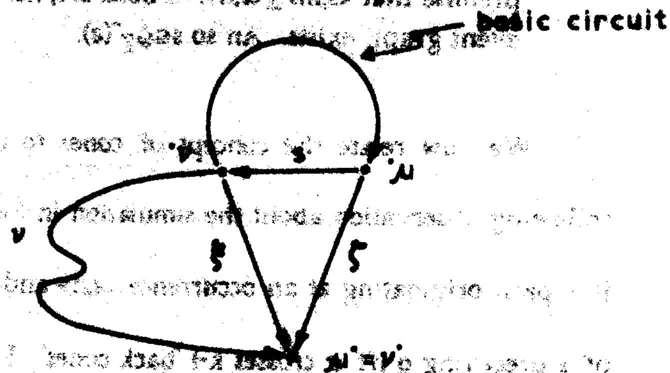
Proof: We'll prove Part (a) by induction on the length of μ.

For $|\mu|=0$, we have $\delta(\mu)=0$ and $|\mu|_{\phi_Z^-(\mu^\circ)}=0$.

For the induction step, it is sufficient to show that when μ is formed by appending State s to the tail of Path ν, then,

$$\delta(\mu) = \begin{cases} \delta(\nu)+1 & \text{if } s\in\phi_Z^-(\mu^\circ) \\ \delta(\nu) & \text{if } s\notin\phi_Z^-(\mu^\circ) \end{cases}$$

Let ξ be a path from ${}^\circ\nu$ to ν° such that $\delta(\xi)=0$, and ζ a path from ${}^\circ\mu$ to μ° such that $\delta(\zeta)=0$. We have,



$\delta(\nu) = |\nu|_\Gamma - |\xi|_\Gamma$

$\delta(\mu) = |\mu|_\Gamma - |\zeta|_\Gamma = |s\nu|_\Gamma - |\zeta|_\Gamma = |s|_\Gamma + |\nu|_\Gamma - |\zeta|_\Gamma$

Combining, we get,

$$\delta(\mu) = \delta(\nu)+|s|_\Gamma+|\xi|_\Gamma-|\zeta|_\Gamma \qquad (1)$$

$s\in\phi_Z^-(\mu^\circ)$: For this case, there cannot exist a path beginning at ${}^\circ\mu$, containing s, ending at μ°, and whose delay is zero. Because $s\xi$ has the first three properties, it must be that $\delta(s\xi)\neq 0$. This implies that $|\zeta|_\Gamma < |s\xi|_\Gamma$, or equivalently,

$|s|_I + |\xi|_I - |\zeta|_I \geq 1$ (2)

By hypothesis, there exists a basic circuit containing State s. This means there's a path from $^\circ\nu$ to $^\circ\mu$ with a token loading of $|s|_I$. When this path is appended to the tail of ζ, we get a path from $^\circ\nu$ to ν° with a token loading of $1 - |s|_I + |\zeta|_I$. Since ξ is a path from $^\circ\nu$ to ν° and $\delta(\xi) = 0$,

$|\xi|_I \leq 1 - |s|_I + |\zeta|_I$ (3)

From Lines (2) and (3) we have

$|s|_I + |\xi|_I - |\zeta|_I = 1$

and from Line (1),

$\delta(\mu) = \delta(\nu) + 1$

$s \notin \phi_Z^-(\mu)$: For this case, there exists a path beginning at $^\circ\mu$, containing s, ending at μ°, and whose delay is zero. Since $s^\circ = {}^\circ\xi$, $\xi^\circ = \mu^\circ$, and $\delta(\xi) = 0$, $s\xi$ must be such a path. That means that $s\xi$ and ζ have the same endpoints and the same delay (zero). Thus, $|s\xi|_I = |\zeta|_I$, or, equivalently,

$|s|_I + |\xi|_I - |\zeta|_I = 0$

From Line (1), we get,

$\delta(\mu) = \delta(\nu)$ □

As an illustration of Theorem 5.6, consider the path μ = [illegible] in Figure 5.1. We have,

$\delta(\mu) = 2$

$^\circ\mu = 3$ and $\mu^\circ = 4$

$\phi^-(\mu^\circ) = \{a,c,e\}$ and $\phi^+(^\circ\mu) = \{b,d,e\}$

$|\mu|_{\phi^-(\mu^\circ)} = 2$ and $|\mu|_{\phi^+(^\circ\mu)} = 2$

Combining Theorem 5.4 and Theorem 5.6, we get the following corollary.

Corollary 5.1: If Z is an initialized event graph covered by basic circuits,
T is a simulation of Z,
$<e_1, n_1>$ and $<e_2, n_2>$ are occurrences in T,
then the following are equivalent.

(a) $<e_1, n_1>$ is the k'th occurrence of e_1 preceding $<e_2, n_2>$

(b) $<e_2, n_2>$ is the k'th occurrence of e_2 following $<e_1, n_1>$

(c) $\exists \sigma \in \Pi(T)$: ${}^\bullet\sigma = <e_1,n_1> \wedge \sigma^\bullet = <e_2,n_2> \wedge \delta_Z(\theta) = k-1$

'There is a path from $<e_1,n_1>$ to $<e_2,n_2>$ and the synchronic delay of its image is k-1.'

(d) $\exists \sigma \in \Pi(T)$: ${}^\bullet\sigma = <e_1,n_1> \wedge \sigma^\bullet = <e_2,n_2> \wedge |\theta|_{\phi_Z^-(e_2)} = k-1$

'There is a path from $<e_1,n_1>$ to $<e_2,n_2>$, and its image crosses the back cone of e_2 k-1 times.'

(e) $\exists \sigma \in \Pi(T)$: ${}^\bullet\sigma = <e_1,n_1> \wedge \sigma^\bullet = <e_2,n_2> \wedge |\theta|_{\phi_Z^+(e_1)} = k-1$

'There is a path from $<e_1,n_1>$ to $<e_2,n_2>$, and its image crosses the front cone of e_1 k-1 times.

The next two theorems provide us with some useful properties of cones.

Theorem 5.7: If $Z=<N,I>$ is an initialized event graph covered by basic circuits, N is connected, and E is the set of events of N, then,

$\forall e \in E$: $\forall \omega \in \Omega(N)$: $|\omega|_I = |\omega|_{\phi_Z^-(e)} = |\omega|_{\phi_Z^+(e)}$

'For any event e, the token loading on a circuit is equal to the number of times e's back (front) cone is crossed.'

Proof: We'll prove just the first equality

Since N is connected and covered by circuits, it is strongly connected. Let μ be a path from ${}^\bullet\omega$ and $\omega^\bullet$ to e. Thus, $\delta(\mu) = 0$ and $\mu^\bullet = e$. Consider the path $\omega\mu$.

$\delta_Z(\omega\mu) = |\omega\mu|_I - |\mu|_I = |\omega|_I$ (1)

In addition, we have by Theorem 5.6(a),

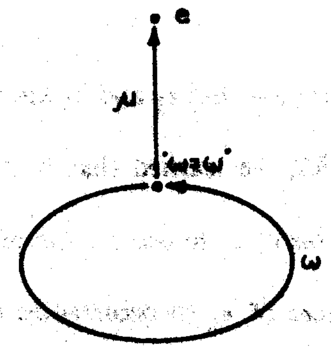
$\delta_Z(\omega\mu) = |\omega\mu|_{\phi_Z^-(e)} = |\omega|_{\phi_Z^-(e)} + |\mu|_{\phi_Z^-(e)}$

But $|\mu|_{\phi_Z^-(e)} = \partial_Z(\mu) = 0$. Thus,

$\partial_Z(\omega\mu) = |\omega|_{\phi_Z^-(e)}$ (2)

From Lines (1) and (2) we get,

$|\omega|_I = |\omega|_{\phi_Z^-(e)}$



□

Theorem 5.8: If Z=<N,I> is an initialized event graph covered by basic circuits, N is connected, and E is the set of events of N, then,

$\forall e \in E$: $\forall \mu_1,\mu_2 \in \Pi(N)$:

$^\bullet\mu_1 = {}^\bullet\mu_2 \wedge \mu_1^\bullet = \mu_2^\bullet \Rightarrow |\mu_1|_I - |\mu_2|_I = |\mu_1|_{\phi^-(e)} - |\mu_2|_{\phi^-(e)} = |\mu_1|_{\phi^+(e)} - |\mu_2|_{\phi^+(e)}$

'If two paths have the same endpoints, then the difference in their token loadings is equal to the difference in the number of times they cross e's back (front) cone.'

Proof: We prove the theorem for just the first equality.

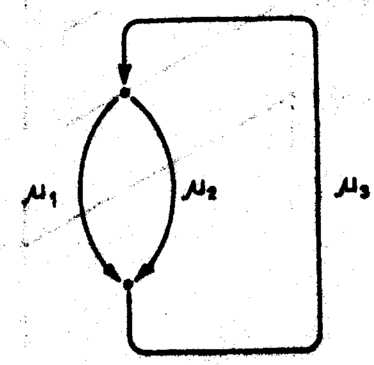
Since N is connected and covered by circuits, it is strongly connected. Let μ_3 be a path from $\mu_1^\bullet$ and $\mu_2^\bullet$ to $^\bullet\mu_1$ and $^\bullet\mu_2$. By Theorem 5.7,

$|\mu_1|_I + |\mu_3|_I = |\mu_1|_{\phi^-(e)} + |\mu_3|_{\phi^-(e)}$

$|\mu_2|_I + |\mu_3|_I = |\mu_2|_{\phi^-(e)} + |\mu_3|_{\phi^-(e)}$

Combining, we get,

$|\mu_1|_I - |\mu_2|_I = |\mu_1|_{\phi^-(e)} - |\mu_2|_{\phi^-(e)}$



□

5.4. System Space:

No theory of systems can be considered complete without notions of space and time. We introduce in this section a notion of 'system space' and in the next section a notion of 'system time'.

Suppose that e_1 and e_2 are events in an initialized event graph covered by basic circuits. In Section 5.2, we learned that in each simulation of the event graph the orderings leading from occurrences of e_1 to occurrences of e_2 are as shown in Figure 5.9(a), and the orderings leading from occurrences of e_2 to occurrences of e_1 are as shown in Figure 5.9(b). The question now is this: How do we combine the two figures to get the complete ordering relationship between occurrences of e_1 and e_2? That question is answered by the concept of synchronic distance.

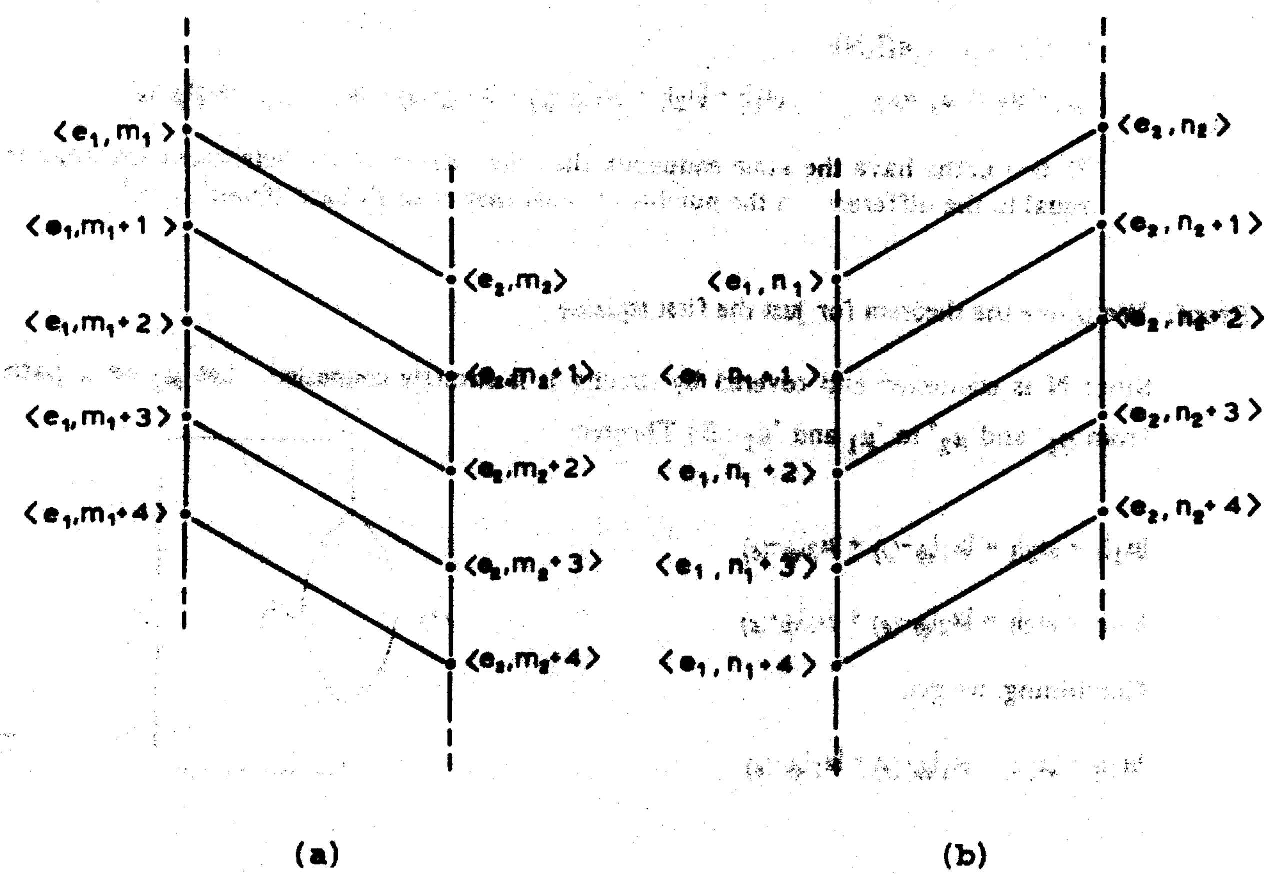


Figure 5.9 Orderings between Occurrences

Synchronic distance is a measure of the 'slack' between two events in an event graph. It determines 'how far ahead' one event can get with respect to another.

Definition: For Events e_1 and e_2 in the strongly-connected event graph $Z=\langle N,I\rangle$,[†]

$$\rho_Z(e_1,e_2) = \begin{cases} 0 & \text{if } e_1=e_2 \\ \min\{|\omega|_I \mid \omega\in\Omega(N) \wedge e_1,e_2\in\omega\} & \text{if } e_1\neq e_2 \end{cases}$$

'$\rho_Z(e_1,e_2)$ is the minimal token loading on those circuits containing both e_1 and e_2.'

$\rho_Z(e_1,e_2)$ is the <u>synchronic distance</u> between e_1 and e_2 (with respect to Z). (When Z is understood, we usually omit it as a subscript of ρ.)

In Table 5.2 we give the synchronic distances between the events in Figure 5.1.

e_2

e_1	1	2	3	4
1	0	1	2	3
2	1	0	1	2
3	2	1	0	1
4	3	2	1	0

$\rho(e_1,e_2)$

Table 5.2: Synchronic Distances

The following theorem provides the connection between synchronic distance and the ordering relationship between two events.

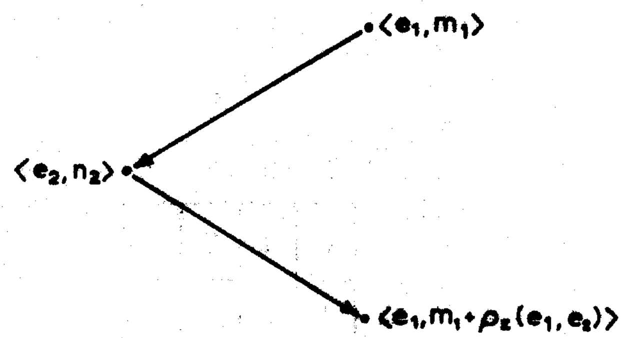
[†] This is equivalent to the notion of distance used by Commoner [13]. (See pp. 112-116)

Theorem 5.9: If Z is an initialized event graph covered by basic circuits and strongly connected,
T is a simulation of Z,
$\langle e_1,m_1\rangle$, $\langle e_2,n_2\rangle$, and $\langle e_1,n_1\rangle$ are occurrences in T,

then,

$$\left.\begin{array}{c}\langle e_1,m_1\rangle \text{ is the last occurrence of } e_1 \text{ preceding } \langle e_2,n_2\rangle \\ \text{and} \\ \langle e_1,n_1\rangle \text{ is the first occurrence of } e_1 \text{ following } \langle e_2,n_2\rangle\end{array}\right\} \Rightarrow n_1 = m_1 + \rho(e_1,e_2)$$

The theorem is illustrated by the following figure.



Proof: By Theorem 5.4 there exists a path σ_1 from $\langle e_1,m_1\rangle$ to $\langle e_2,n_2\rangle$ and a path from σ_2 from $\langle e_2,n_2\rangle$ to $\langle e_1,n_1\rangle$ such that $\delta(\sigma_1) = 0$ and $\delta(\sigma_2) = 0$. Let Z = $\langle N,I\rangle$ and $\sigma = \sigma_1 \cdot \sigma_2$. Because σ_1 is a path from e_1 to e_2 of minimal token loading and σ_2 is a path from e_2 to e_1 of minimal token loading, σ must have minimal token loading with respect to those circuits containing both x and y. In other words, $|\sigma|_I = \rho_Z(e_1, e_2)$. Since σ is a path from $\langle e_1,m_1\rangle$ to $\langle e_1,n_1\rangle$, we have, by Lemma 5.1, that $n_1 - m_1 = |\sigma|_I$. The desired result follows. □

We see in Figure 5.2 that <1,1> is the last occurrence of Event 1 preceding <3,1>, and <1,3> is the next occurrence of Event 1 following <3,1>. The difference in occurrence numbers of the two occurrences of Event 1 is 2. This is the synchronic distance Events 1 and 3.

With Theorem 5.9, if we have an initialized event graph satisfying the necessary requirements, then we can determine the ordering relationship between occurrences of two events. All we need know is the synchronic distance between the two events. In Figure 5.10, we show the ordering relationships for several values of $\rho(e_1,e_2)$.

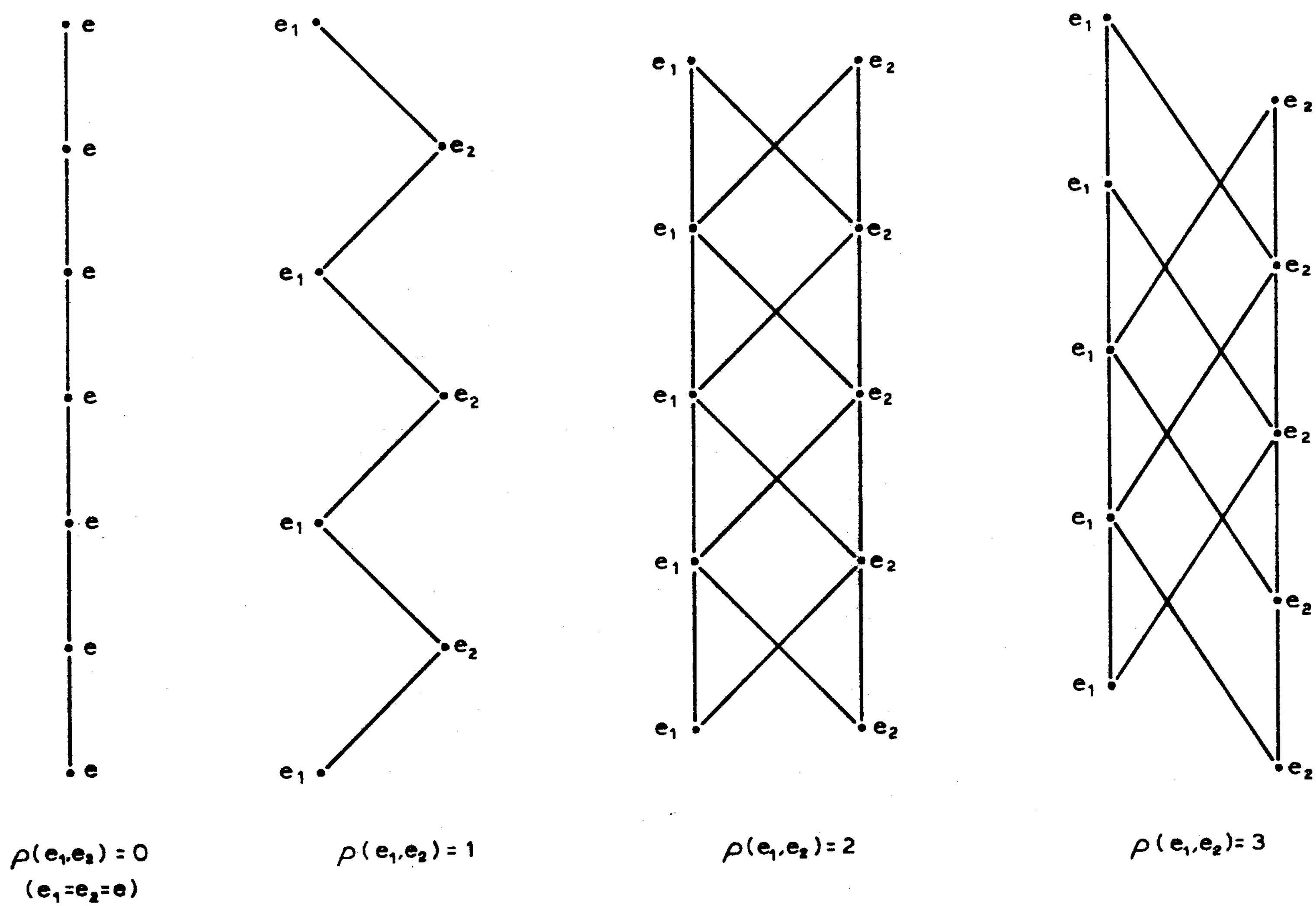


Figure 5.10 Ordering Relationships between Occurrences of Two Events

The next theorem states that ρ is a metric when the event graph satisfied two elementary properties.

Theorem 5.10: If Z is an initialized event graph that is (1) strongly connected and (2) free of blank circuits, then ρ_Z is a metric on the set of events. Specifically, for events e_1, e_2, and e_3,

(a) $\rho_Z(e_1,e_2) = 0 \Leftrightarrow e_1=e_2$

(b) $\rho_Z(e_1,e_2) = \rho_Z(e_2,e_1)$

(c) $\rho_Z(e_1,e_3) \leq \rho_Z(e_1,e_2)+\rho_Z(e_2,e_3)$

Proof: First of all, strong connectivity guarantees that ρ_Z is well defined. Property (a) follows from the fact that Z is free of blank circuits. Properties (b) and (c) follow directly from the definition of synchronic distance.

Strong connectivity and absence of blank circuits, together with coverability by basic circuits, might be regarded as criteria for 'well-formedness' in event graphs. Earlier work [4] has shown that absence of blank circuits and coverability by basic circuits are necessary and sufficient conditions for 'liveness' and 'safeness' in event graphs.†

We now relate the ideas of this section to the theory in Chapter 3. Theorems 3.3 and 3.8 and Property 3.6 state that the initialized control structure is a strongly connected event graph covered by basic circuits. Thus, Theorem 5.9 applies. Furthermore, we have the following corollary to Theorem 5.10.

Corollary 5.2: If the initialized control structure Z^* is free of blank circuits, then $\langle E^*,\rho_{Z^*}\rangle$ is a metric space.

† Liveness means that simulations of the event graph can be extended arbitrarily far. Safety means that instances of the same element are totally ordered.

Definition: If Z^* is free of blank circuits, then $\langle E^*, \rho_{Z^*} \rangle$ is called the system space.

5.5. System Time:

In most theories of system behavior, 'time' is introduced as a primitive concept. Our approach is novel in that the concept of time is derived from the logical structure of a system. We only require that the initialized control structure satisfy three simple properties. There is no need to augment the definition of a system, and there is no need to modify the simulation rule.

Definition: An initialized event graph $\langle N,I \rangle$ is said to be synchronous iff it (1) is connected, (2) is covered by basic circuits, and (3) satisfies the synchrony property:

$$\exists k \in N: \quad \forall \omega \in \Omega^e(N): \quad |\omega| = k|\omega|_I$$

'The length of each elementary circuit is proportional to its token loading.'

An initialized event graph that is not synchronous is called nonsynchronous. A synchronous system is one in which the initialized control structure is synchronous. A system that is not synchronous is called nonsynchronous.[†]

For examples of synchronous systems, the reader may refer back to Figures 3.13-3.15. In Figure 3.13(d), the proportionality constant between the length of an elementary circuit and its token loading is 2[‡]. In Figure 3.14(d), it is also 2. In Figure 3.15(d), it is 4. An example of a nonsynchronous event graph is shown in Figure 5.11.

[†] The question of what an 'asynchronous' system is is outside the scope of this discussion.

[‡] In determining the length of a circuit or a path in an event graph, we count the arcs in the abbreviated representation of the event graph. This reduces the length by a factor of two.

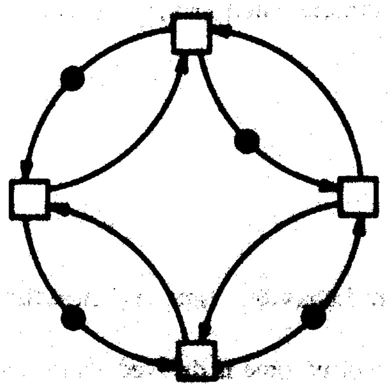


Figure 5.11 A Nonsynchronous Event Graph

We consider now some basic properties of synchronous event graphs. Because a synchronous event graph is connected and covered by basic circuits, we have the following:

Property 5.3: A synchronous event graph is strongly connected.

From the synchrony property, we get the following.

Property 5.4: A synchronous event graph is free of blank circuits.

Also from the synchrony property, we know that the length of each elementary circuit is a multiple of k, and that the length of each basic circuit is equal to k. It follows that k is equal to the gcd of the lengths of the elementary circuits. Now, because any circuit, elementary or otherwise, can be viewed as the superposition of elementary circuits, the synchrony property applies to all circuits. These results are reflected in the next property.

Property 5.5: If <N,I> is a synchronous event graph, then,

$\forall \omega \in \Omega(N)$: $|\omega| = \gamma(N) \cdot |\omega|_I$

To help us in understanding the notion of time presented below, we introduce the concept of the 'phase relation'. The phase relation is generated from an event graph in exactly the same way that the alternativeness relation is generated from a part. The same 'collapsing' procedure is used.

Definition: The <u>phase relation</u> for the event graph N=<S,E,F> is the minimal relation $\beta_N \subseteq (S \cup E)^2$ such that

$\forall x \in (S \cup E)$: $x \beta_N x$

$\forall x_1,x_2,x_3,x_4 \in (S \cup E)$: $x_1 \beta_N x_2 \wedge ((x_1 \cdot x_3 \wedge x_2 \cdot x_4) \vee (x_3 \cdot x_1 \wedge x_4 \cdot x_2)) \Rightarrow x_3 \beta_N x_4$

Elements x_1 and x_2 are said to be <u>in phase</u> iff $x_1 \beta_N x_2$.

The concepts and results established in Section 3.3 for the alternativeness relation carry over the the phase relation:

Property 5.6: If N is the event graph <S,E,F>, then β_N is an equivalence relation on S∪E, and,

$\forall A \in (S \cup E)/\beta_N$: $A \subseteq S \vee A \subseteq E$

'Each equivalence class induced by β_N contains either exclusively states or exclusively events.'

Definition: If N is an event graph, then those equivalence classes induced by β_N consisting of states are called <u>phases</u> and those consisting of events are called <u>phase transitions</u>.

Definition: If N is the strongly-connected event graph $<S,E,F>$, then the <u>fundamental circuit</u> of N is the quotient net $\tilde{N}=<\tilde{S},\tilde{E},\tilde{F}>$, where,

$$\tilde{S} = \{[s]_{\beta_N} \mid s \in S\}$$
$$\tilde{E} = \{[e]_{\beta_N} \mid e \in E\}$$
$$\tilde{F} = \{<[x]_{\beta_N},[y]_{\beta_N}> \mid <x,y> \in F\}$$

Property 5.7: $\tilde{N}$ is a net.

Property 5.8: $\tilde{N}$ is an elementary circuit of length $\gamma(N)$.†

In Figure 5.12, we show how the fundamental circuit for the event graph in Figure 5.1 is generated.

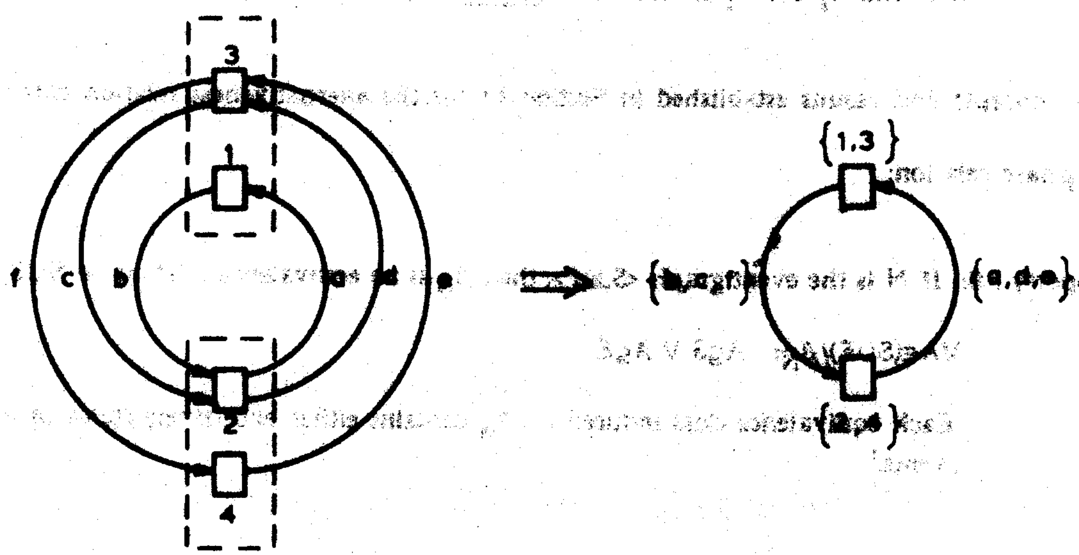


Figure 5.12 Generating a Fundamental Circuit

The fundamental circuit may be thought of as a 'clock'. Now with this interpretation, it might appear that to make an event graph 'synchronous', it will be necessary to 'fire' the events in

† This property corresponds to Theorem 3.2

a phase transition 'in unison'. However, this is not the case. In fact, it won't even be necessary for the initial conditions to be in phase. All that's required is that the initial conditions belong to a 'marking class' in which it is possible for just those states in a given phase to hold. As it turns out, there is exactly one such marking class. It has been shown[†] that in a strongly connected event graph free of blank circuits, two 'markings' belong to the same marking class iff they induce the same token loading on each circuit. Now in the situation where just those states in a given phase hold, the token loading on each circuit is known:

$$\forall \omega \in \Omega(N): \qquad |\omega|_I = \frac{|\omega|}{\gamma(N)}$$

But this is just Property 5.5, which is equivalent to the synchrony property. Therefore, a set of initial conditions can be brought 'into phase' iff the synchrony property is satisfied.

With the preceding discussion as background, we're now ready to develop the major results pertaining to synchronous event graphs.

Theorem 5.11: If <N,I> is a synchronous event graph, then,

$$\forall \mu_1, \mu_2 \in \Pi(N): \; {}^{\bullet}\mu_1 = {}^{\bullet}\mu_2 \wedge \mu_1^{\bullet} = \mu_2^{\bullet} \Rightarrow |\mu_1| - \gamma(N) \cdot |\mu_1|_I = |\mu_2| - \gamma(N) \cdot |\mu_2|_I$$

'If μ_1 and μ_2 are paths in N having the same endpoints, then the length of μ_1 minus $\gamma(N)\cdot$(token loading on μ_1) equals the length of μ_2 minus $\gamma(N)\cdot$(token loading on μ_2).'

Rroof: Because a synchronous event graph is strongly connected, there exists a path μ_3 from $\mu_1^{\bullet}$ and $\mu_2^{\bullet}$ to ${}^{\bullet}\mu_1$ and ${}^{\bullet}\mu_2$. From Property 5.5 we have,

$$|\mu_1| + |\mu_3| = \gamma(N) \cdot (|\mu_1|_I + |\mu_3|_I)$$

$$|\mu_2| + |\mu_3| = \gamma(N) \cdot (|\mu_1|_I + |\mu_3|_I)$$

[†] Theorem 11 in [4]

Combining, we get,

$|\mu_1|-\gamma(N)\cdot|\mu_1|_I = |\mu_2|-\gamma(N)\cdot|\mu_2|_I$ □

Theorem 5.12: If <N,I> is a synchronous event graph, then,

$\forall \mu_1,\mu_2 \in \Pi(N): {}^\bullet\mu_1 = \mu_2^\bullet \wedge \mu_1^\bullet = {}^\bullet\mu_2 \Rightarrow |\mu_1|-\gamma(N)\cdot|\mu_1|_I = \gamma(N)\cdot|\mu_2|_I-|\mu_2|$

'If $\mu_1\cdot\mu_2$ forms a circuit then ...'

Proof: Because $\mu_1\mu_2$ forms a circuit, we have by Property 5.5,

$|\mu_1|+|\mu_2| = \gamma(N)(|\mu_1|_I+|\mu_2|_I)$

The theorem follows.

Theorem 5.13: If T=<H,O,C> is a simulation of a synchronous event graph, then,

$\forall \sigma_1,\sigma_2 \in \Pi(T): {}^\bullet\sigma_1 = {}^\bullet\sigma_2 \wedge \sigma_1^\bullet = \sigma_2^\bullet \Rightarrow |\sigma_1| = |\sigma_2|$

'In a simulation of a synchronous event graph, if two paths have the same endpoints, then they must be the same length.'

Proof: From theorem 5.1, we know that $|\theta_1|_I = |\theta_2|_I$. Since θ_1 and θ_2 are both paths in the synchronous event graph generating T, we have $|\theta_1| = |\theta_2|$ by Theorem 5.11. It follows that $|\sigma_1| = |\sigma_2|$. □

Before we can define the 'time interval' between two holdings or two occurrences in a 'synchronous simulation', we must first have a notion of 'how far out of phase' the first instances of two states or two events are.

Definition: If Z is the synchronous event graph <N,I>, then,
for Events e and e_2,

$$\partial_Z(e_1,e_2) = n \quad \text{iff} \quad \exists \mu \in \Pi(N)\colon {}^{\bullet}\mu = e_1 \wedge \mu^{\bullet} = e_2 \wedge n = |\mu| - \gamma(N)|\mu|_I$$

'$\partial_Z(e_1,e_2) = n$ iff there exists a path μ from e_1 to e_2 such that $n = |\mu| - \gamma(N)|\mu|_I$.'

for States s_1 and s_2,

$\partial_Z(s_1, s_2) = \partial_Z(e_1, e_2)$ where e_1 is the unique output event of s_1 and e_2 is the unique output event of s_2.

(Theorem 5.11 guarantees that ∂_Z is well defined.)

$\partial_Z(x_1,x_2)$ tells us 'how far ahead' the first instance of x_2 is going to be with respect to the first instance of x_1. In Table 5.3, we give the values of $\partial(e_1,e_2)$ and $\partial(s_1,s_2)$ for the synchronous event graph of Figure 5.1.

(a) $\partial(e_1,e_2)$

e_1 \ e_2	1	2	3	4
1	0	1	2	3
2	-1	0	1	2
3	-2	-1	0	1
4	-3	-2	-1	0

(b) $\partial(s_1,s_2)$

s_1 \ s_2	a	b	c	d	e	f
a	0	1	1	2	2	3
b	-1	0	0	1	1	2
c	-1	0	0	1	1	2
d	-2	-1	-1	0	0	1
e	-2	-1	-1	0	0	1
f	-3	-2	-2	-1	-1	0

Table 5.3

Using ∂_Z, we now define the 'time interval' between two holdings or two occurrences in a synchronous simulation.

Definition: If $\langle x_1,n_1\rangle$ and $\langle x_2,n_2\rangle$ are either two holdings or two occurrences in a simulation of the synchronous event graph $Z=\langle N,I\rangle$, then,

$$\Delta_Z(\langle x_1,n_1\rangle,\langle x_2,n_2\rangle) = (n_2-n_1)\gamma(N) + \partial_Z(x_1,x_2)$$

$\Delta_Z(q_1,q_2)$ is the <u>time interval</u> from q_1 to q_2.

We give some sample time intervals for the simulation in Figure 5.2. In this case, $\gamma(N)=2$.

$$\Delta(\langle d,1\rangle,\langle 4,3\rangle) = 2x2+3 = 7$$

$$\Delta(4,2\rangle,\langle d,3\rangle) = 1x2-3 = -1$$

$$\Delta(b,2\rangle,\langle f,1\rangle) = (-1)x2+2 = 0$$

$$\Delta(\langle a,2\rangle,\langle a,2\rangle) = 0x2+2 = 2$$

The next three theorems show that Δ_Z satisfies those properties that one would naturally expect of a metric for time.

Theorem 5.14: If q_1, q_2, and q_3 are either three holdings or three occurrences in a simulation of the synchronous event graph Z, then,

(a) $\Delta_Z(q_1,q_1) = 0$

(b) $\Delta_Z(q_1,q_2) = -\Delta_Z(q_2,q_1)$

(c) $\Delta_Z(q_1,q_3) = \Delta_Z(q_1,q_2) + \Delta_Z(q_2,q_3)$

Proof: (a) Follows directly from the definitions of Δ_Z and ∂_Z.

(b) $\partial_Z(x_1,x_2) = -\partial_Z(x_2,x_1)$ by Theorem 5.12. It follows that $\Delta_Z(q_1,q_2) = -\Delta_Z(q_1,q_2)$.

(c) Let $q_1=\langle x_1,n_1\rangle$, $q_2=\langle x_2,n_2\rangle$, $q_3=\langle x_3,n_3\rangle$, and let $Z = \langle N, I\rangle$. We have,

$$\Delta(q_1,q_3) = (n_3-n_1)\,\gamma(N) + \partial(x_1,x_3)$$

and $\Delta(q_1,q_2) + \Delta(q_2,q_3) = (n_2-n_1)\gamma(N) + \partial(x_1,x_2) + (n_3-n_2)\gamma(N) + \partial(x_2,x_3)$

For occurrences (and similarly for holdings),

$$\partial(x_1,x_2) + \partial(x_2,x_3) = n \Leftrightarrow \exists \mu_1,\mu_2 \in \Pi(N): {}^{\bullet}\mu_1 = x_1 \wedge \mu_1{}^{\bullet} = x_2 \wedge {}^{\bullet}\mu_2 = x_2 \wedge \mu_2{}^{\bullet} = x_3$$

$$\wedge\ n = |\mu_1| + |\mu_2| - \gamma(N)(|\mu_1|_I + |\mu_2|_I)$$

$$\Leftrightarrow \exists \mu \in \Pi(N): {}^{\bullet}\mu = x_1 \wedge \mu^{\bullet} = x_3 \wedge n = |\mu| - \gamma(N)|\mu|_I$$

$$\Leftrightarrow \partial(x_1,x_3) = n$$

Thus, $\partial(x_1, x_2) + \partial(x_2, x_3) = \partial(x_1, x_3)$. It follows that,

$$\Delta(q_1, q_2) + \Delta(q_2, q_3) = (n_3-n_1)\gamma(N) + \partial(x_1, x_3) = \Delta(q_1, q_3)$$ □

Theorem 5.15: If T=<H,O,C> is a simulation of the synchronous event graph Z, then,

$\forall \sigma \in \Pi(T)$:

$${}^{\bullet}\sigma, \sigma^{\bullet} \in O \Rightarrow \Delta_Z({}^{\bullet}\sigma, \sigma^{\bullet}) = |\sigma|_H \quad \text{(a)}$$

$${}^{\bullet}\sigma, \sigma^{\bullet} \in H \Rightarrow \Delta_Z({}^{\bullet}\sigma, \sigma^{\bullet}) = |\sigma|_O \quad \text{(b)}$$

'If σ is a path in T connecting two occurrences (holdings), then the time interval between ${}^{\bullet}\sigma$ and $\sigma^{\bullet}$ is equal to the number of holdings (occurrences) crossed by σ.'

Proof: (a) Let ${}^{\bullet}\sigma = \langle e_1,n_1\rangle$, $\sigma^{\bullet} = \langle e_2,n_2\rangle$, and Z = <N, I>. Thus,

$$\Delta_Z({}^{\bullet}\sigma,\sigma^{\bullet}) = (n_2-n_1)\ \gamma(N) + \partial_Z(e_1,e_2)$$

Since $\hat{\sigma}$ is a path from e_1 to e_2, the definition of ∂ gives us,

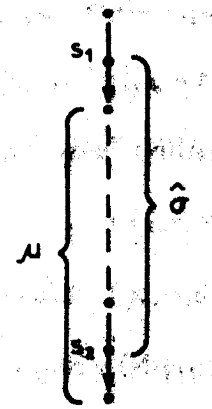
$$\partial_Z(e_1,e_2) = |\hat{\sigma}| - \gamma(N)|\hat{\sigma}|_I$$

We know that $|\sigma|_H = |\hat{\sigma}|$, and, therefore,

$$\Delta({}^{\bullet}\sigma,\sigma^{\bullet}) = (n_2-n_1)\ \gamma(N) + |\sigma|_H - \gamma(N)|\hat{\sigma}|_I$$

From Lemma 5.1, we have $n_2-n_1 = |\hat{\sigma}|_I$. The desired result follows immediately.

(b) Let ${}^{\bullet}\sigma = \langle s_1,n_1\rangle$ and $\sigma^{\bullet} = \langle s_2,n_2\rangle$. Let μ be generated from $\hat{\sigma}$ according to the following diagram,



μ is a path in N, and $|\sigma|_O = |\hat{\sigma}|_E = |\mu|$. From the definition of ∂ we get,

$$\partial(s_1,s_2) = |\mu| - \gamma(N)|\mu|_I = |\sigma|_O - \gamma(N)|\mu|_I$$

Which, in turn, produces,

$$\Delta({}^{\bullet}\sigma, \sigma^{\bullet}) = (n_2-n_1)\gamma(N) + |\sigma|_O - \gamma(N)|\mu|_I$$

Because of the way in which μ is constructed, we have from Lemma 5.1, $n_2-n_1 = |\mu|_I$. The desired result follows. □

Theorem 5.16: If <H, O, C> is a simulation of the synchronous event graph Z, then,

$\forall q_1, q_2 \in O$: $\forall h_1, h_2 \in H$:

$$q_1{}^{\bullet}h_1 \wedge q_2{}^{\bullet}h_2 \Rightarrow \Delta_Z(q_1, q_2) = \Delta_Z(h_1, h_2) \qquad \text{(a)}$$

$$h_1{}^{\bullet}q_1 \wedge h_2{}^{\bullet}q_2 \Rightarrow \Delta_Z(q_1, q_2) = \Delta_Z(h_1, h_2) \qquad \text{(b)}$$

'If Occurrences q_1 and q_2 initiate (terminate) Holdings h_1 and h_2 respectively, then the time interval between q_1 and q_2 is the same as the time interval between h_1 and h_2.

Proof: We'll prove just Part (a).

Let $q_1 = \langle e_1,n_1\rangle$, $q_2 = \langle e_2,n_2\rangle$, $h_1 = \langle s_1,m_1\rangle$, and $h_2 = \langle s_2,m_2\rangle$. And let Z = <N, I>. We know that $e_1{}^{\bullet}s_1$ and $e_2{}^{\bullet}s_2$. Now let μ be a path in N from the unique output event of s_1 to e_2. Let e_3 be the unique output event of s_2. We get,

$$\Delta(q_1, q_2) = (n_2 - n_1)\,\gamma(N) + |e_1 s_1 \mu| - \gamma(N)|s_1 \mu|_I$$

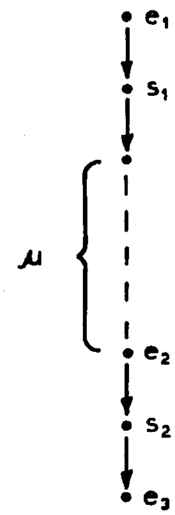
$$\Delta(h_1, h_2) = (m_2 - m_1)\,\gamma(N) + |\mu s_2 e_3| - \gamma(N)|\mu s_2|_I$$

We have immediately, $|e_1 s_1 \mu| = |\mu| + 1 = |\mu s_2 e_3|$. From Lemma 5.1, we get, $m_1 = n_1 + |s_1|_I$ and $m_2 = n_2 + |s_2|_I$. Thus,

$$\Delta(h_1, h_2) = (n_1 + |s_2|_I - n_1 - |s_1|_I)\gamma(N) + |e_1 s_1 \mu| - \gamma(N)|\mu s_2|_I$$

$$= (n_2 - n_1)\,\gamma(N) + |e_1 s_1 \mu| - \gamma(N)|s_1 \mu|_I$$

$$= \Delta(q_1, q_2)$$



□

The notion of 'simultaneity' is defined in a straightforward way.

Definition: If q_1 and q_2 are either two occurrences or two holdings in a simulation of the synchronous event graph Z, then,

$$q_1 \,\tau_Z\, q_2 \iff \Delta_Z(q_1, q_2) = 0$$

τ_Z is called the simultaneity relation. We say that instances q_1 and q_2 are simultaneous iff $q_1 \,\tau_Z\, q_2$.

Property 5.9: If T is a simulation of the synchronous event graph Z, then τ_Z defines an equivalence relation on the holdings and occurrences of T. Furthermore, each equivalence class induced by τ_Z contains either exclusively holdings or exclusively occurrences.

Definition: In a simulation of the synchronous event graph Z, the equivalence classes induced by τ_Z are called simultaneity classes. A simultaneity class of occurrences is called an instant of time, or just a time. A simultaneity class of holdings is called an (elementary) interval of time.

We have the following property from Theorem 5.15.

Property 5.10: If two instances in a 'synchronous simulation' are simultaneous, then they are either coincident or concurrent.

We have the following property from Theorem 5.16.

Property 5.11: If q_1, q_2, q_3, and q_4 are instances in a simulation of the synchronous event graph Z, then,

$$q_1{}^\bullet q_3 \wedge q_2{}^\bullet q_4 \wedge q_1 \tau_Z q_2 \Rightarrow q_3 \tau_Z q_4 \quad \text{(a)}$$

$$q_1{}^\bullet q_3 \wedge q_2{}^\bullet q_4 \wedge q_3 \tau_Z q_4 \Rightarrow q_1 \tau_Z q_2 \quad \text{(b)}$$

'If two instances are simultaneous, then their immediate successors (predecessors) are also simultaneous.'

Properties 5.10 and 5.11 mean that in a 'synchronous simulation', the simultaneity classes form a series of 'slices', with instants of time and intervals of time strictly alternating. This is illustrated in Figure 5.13. The simulation depicted is generated from the synchronous event graph in Figure 5.1.

Theorem 5.17: If q_1 and q_2 are instances in a simulation of the synchronous event graph Z, then,

$$q_1 \tau_Z q_2 \Rightarrow \hat{q}_1 \beta_Z \hat{q}_2$$

'If q_1 and q_2 are simultaneous, then $\hat{q}_1$ and $\hat{q}_2$ must be in phase.'

Proof: We give the proof for occurrences, the proof for holdings being similar. Let $Z = \langle N, I\rangle$, and let $q_1 = \langle x_1, n_1\rangle$ and $q_2 = \langle x_1, n_1\rangle$. We have,

$$q_1 \tau_Z q_2 \Leftrightarrow \Delta_Z(q_1, q_2) = 0$$

$$\Leftrightarrow (n_2 - n_1)\gamma(N) + \partial_Z(x_1, x_2) = 0$$

$$\Leftrightarrow \exists \mu \in \Pi(N):\ {}^\bullet\mu = e_1 \wedge \mu^\bullet = e_2 \wedge |\mu| - \gamma(N)|\mu|_I = (n_1 - n_2)\gamma(N)$$

$$\Leftrightarrow \exists \mu \in \Pi(N):\ {}^\bullet\mu = e_1 \wedge \mu^\bullet = e_2 \wedge |\mu| = (n_1 - n_2 + |\mu|_I)\gamma(N)$$

Thus, $q_1 \tau_Z q_2$ implies that there exists a path between x_1 and x_2 whose length is a multiple of $\gamma(N)$. It follows that x_1 and x_2 must be in phase. □

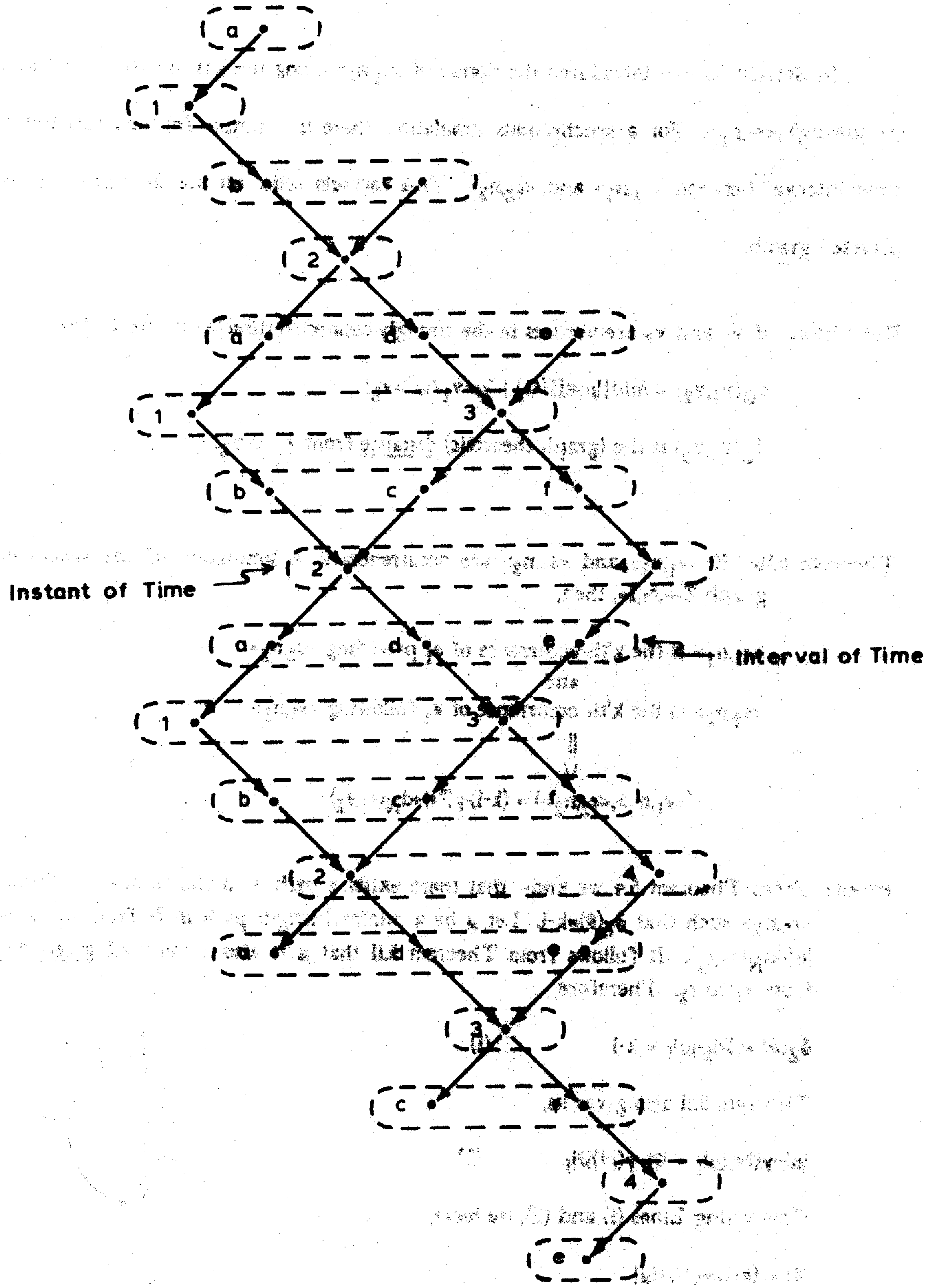


Figure 5.13 Simultaneity Classes

In Section 5.2, we introduced the notion of $<e_1,n_1>$ being the k'th occurrence of e_1 preceding (following) $<e_2,n_2>$. For a synchronous simulation, there is a simple formula relating k and the time interval between $<e_1,n_1>$ and $<e_2,n_2>$. This formula relies on the concept of 'distance' in a directed graph.

Definition: If v_1 and v_2 are vertices in the strongly connected directed graph G, then,

$$d_G(v_1,v_2) = \min\{|\mu| \mid \mu \in \Pi(G) \mid {}^{\bullet}\mu = v_1 \wedge \mu^{\bullet} = v_2\}$$

$d_G(v_1,v_2)$ is the (graph theoretic) <u>distance</u> from v_1 to v_2.

Theorem 5.18: If $<e_1,n_1>$ and $<e_2,n_2>$ are occurrences in a simulation of the synchronous event graph $Z=<N,I>$, then,

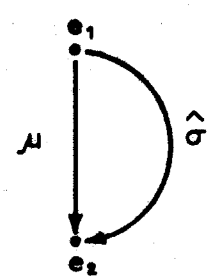
$<e_1,n_1>$ is the k'th occurrence of e_1 preceding $<e_2,n_2>$
and
$<e_2,n_2>$ is the k'th occurrence of e_2 following $<e_1,n_1>$

$$\Downarrow$$

$$\Delta_Z(<e_1,n_1>,<e_2,n_2>) = (k-1)\gamma(N)+d_N(e_1,e_2)$$

Proof: From Theorem 5.4 we know that there exists a path σ in the simulation from $<e_1,n_1>$ to $<e_2,n_2>$ such that $\delta_Z(\hat{\sigma})=k-1$. Let μ be a minimal length path in N from e_1 to e_2. That is, $|\mu|=d_N(e_1,e_2)$. It follows from Theorem 5.11 that μ is also a minimal <u>token</u> <u>loading</u> path from e_1 to e_2. Therefore,

$$\delta_Z(\hat{\sigma}) = |\hat{\sigma}|_I - |\mu|_I = k-1 \qquad (1)$$



Theorem 5.11 also gives us,

$$|\mu|-\gamma(N)|\mu|_I = |\hat{\sigma}|-\gamma(N)|\hat{\sigma}|_I \qquad (2)$$

Combining Lines (1) and (2), we have,

$$|\hat{\sigma}| = (k-1)\gamma(N)+|\mu|$$

But $|\hat{\sigma}|=|\sigma|_H$ and $|\mu|=d_N(e_1,e_2)$, and so,

$$|\sigma|_H = (k-1)\gamma(N)+d_N(e_1,e_2)$$

From Theorem 5.15(a) we have $|\sigma|_H = \Delta_Z(\langle e_1,n_1\rangle\langle e_2,n_2\rangle)$. The desired result follows. □

In the simulation of Figure 5.2, we see that <1,1> is the third occurrence of Event 1 preceding <4,3>. We have k=3, $\gamma(N)=2$, and $d_N(1,4)=3$. Thus,

$$\Delta(\langle 1,1\rangle,\langle 4,3\rangle) = (3\text{-}1)\text{x}2 + 3 = 7$$

This checks with the value of $\Delta(\langle 1,1\rangle,\langle 4,3\rangle)$ computed earlier in this section.

The final results of this section have to do with four functions defined earlier in this chapter. For general event graphs, these functions depend upon the set of initial conditions, but for synchronous event graphs, they are independent of the initial conditions.

Property 5.12: For a path μ in the synchronous event graph Z=<N,I>,

$$\delta_Z(\mu) = (|\mu|\text{-}d_N({}^{\bullet}\mu,\mu^{\bullet})) \;/\; \gamma(N)$$

Property 5.13: If e is an event in the synchronous event graph Z=<S,E,F,I>, then,

$$\phi_Z^-(e) = \{s\in S \mid \exists\mu\in\Pi(N){:}\ {}^{\bullet}\mu{\bullet}s \wedge s\in\mu \wedge \mu^{\bullet}{=}e \wedge |\mu|{=}d_N({}^{\bullet}\mu,\mu^{\bullet})\}$$

$$\phi_Z^+(e) = \{s\in S \mid \exists\mu\in\Pi(N){:}\ {}^{\bullet}\mu{=}e \wedge s\in\mu \wedge s{\bullet}\mu^{\bullet} \wedge |\mu|{=}d_N({}^{\bullet}\mu,\mu^{\bullet})\}$$

Property 5.14: If e_1 and e_2 are events in the synchronous event graph Z=<N,I>, then,

$$\rho_Z(e_1,e_2) = (d_N(e_1,e_2)+d_N(e_2,e_1)) \;/\; \gamma(N)$$

Before concluding this section, we should perhaps say a word about the distinction between 'system time' and 'observer time'. System time is strictly a system-relative concept, and is observer independent. Observer time, on the other hand, is relative to a particular observer. In the case of a clocked system, the two notions of time are, for practical purposes, the same. However, in the case of an unclocked system, the two notions of time may bear little resemblance to one another. For example, it might be possible for Instance q_1 to precede Instance q_2 in system time, and for q_1 to follow q_2 in observer time, or vice versa. The only thing we can say with certainty is that if there is a causal connection leading from q_1 to q_2, then q_1 will precede q_2 in both system time and observer time.

CHAPTER 6

PREDICTION AND POSTDICTION

6.1. Information and Control:

In Chapters 4 and 5, we examined separately the two components of system behavior: information and control. In this chapter, results from the two areas are brought together to produce a technique for predicting and postdicting system behavior.

As we showed in Chapter 3, for each system simulation there is a corresponding control simulation, and the two are isomorphic. Consider a pair of corresponding simulations. Because the control structure is an event graph, the control simulation has the regular properties described in Chapter 5. Since the system simulation is isomorphic to the control simulation, it too has these regular properties. Of course, the system simulation also has certain 'irregular' properties, but these are describable using the concepts of information flow.

It is the irregular properties of system simulations that are the focus of this chapter. However, in getting our results, we will take advantage of both the properties of information flow and the regular properties of event graph simulations.

6.2. Transactions:

Suppose that we have a system simulation and a corresponding control simulation. Within the control simulation, there is a total ordering among occurrences of the same meeting (Corollary 3.4). For each such total ordering, there is a corresponding total ordering in the system simulation among occurrences of those events belonging to the related meeting. These ideas are illustrated in Figures 6.1-6.4. In Figures 6.1 and 6.2, we've redrawn the initialized system net and the initialized control structure for the bit-pipeline example of Section 3.7. In Figures 6.3 and 6.4 we give a system simulation and the corresponding control simulation. We've indicated in the control simulation the occurrences of Meeting {5,6}, and we've indicated in the system simulation the occurrences of Events 5 and 6. Note the one-to-one correspondence between the two sets of occurrences. The same idea applies to Meetings [illegible], [illegible] and [illegible].

We adopt this terminology:

Definition: If, within a system simulation, the n'th occurrence associated with Meeting m exists and is an occurrence of Event *e*, then we say that *e* is the n'th __transaction__ at Meeting m (for that simulation).

For the system simulation in Figure 6.3, we have the following.

Event 5 is the __first__ transaction at Meeting {5,6}
Event 6 is the __second__ transaction at Meeting {5,6}
Event 5 is the __third__ transaction at Meeting {5,6}

Now suppose that *q* is either a holding or an occurrence in a system simulation. Then we can speak of the n'th transaction at Meeting m 'relative to *q*'.

Definition: If T is a system simulation,
q is an instance in T,
e is an event,
m is a meeting,
n is a nonzero integer,

then,

Within T, __*e* is the n'th transaction at Meeting m relative to *q*__

iff

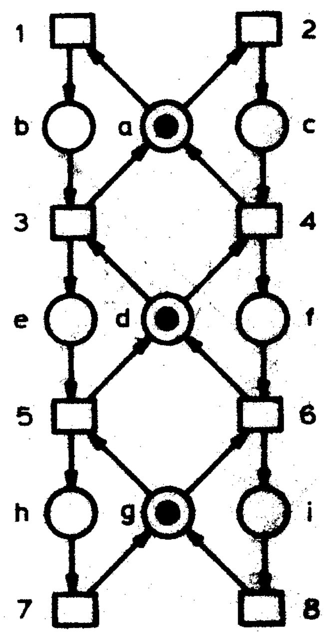


Figure 6.1 Initialized System Net

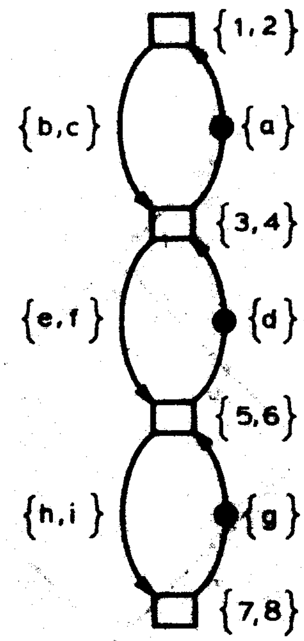


Figure 6.2 Initialized Control Structure

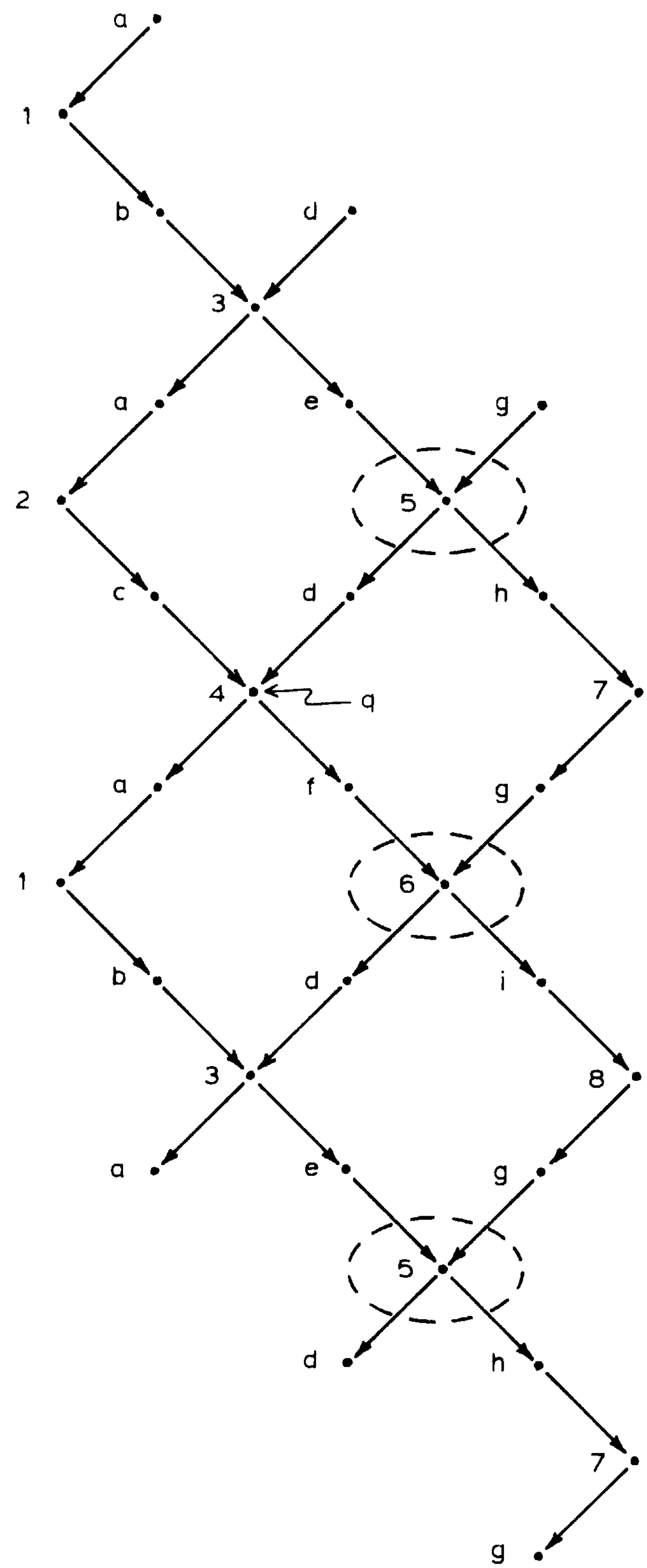


Figure 6.3 A System Simulation

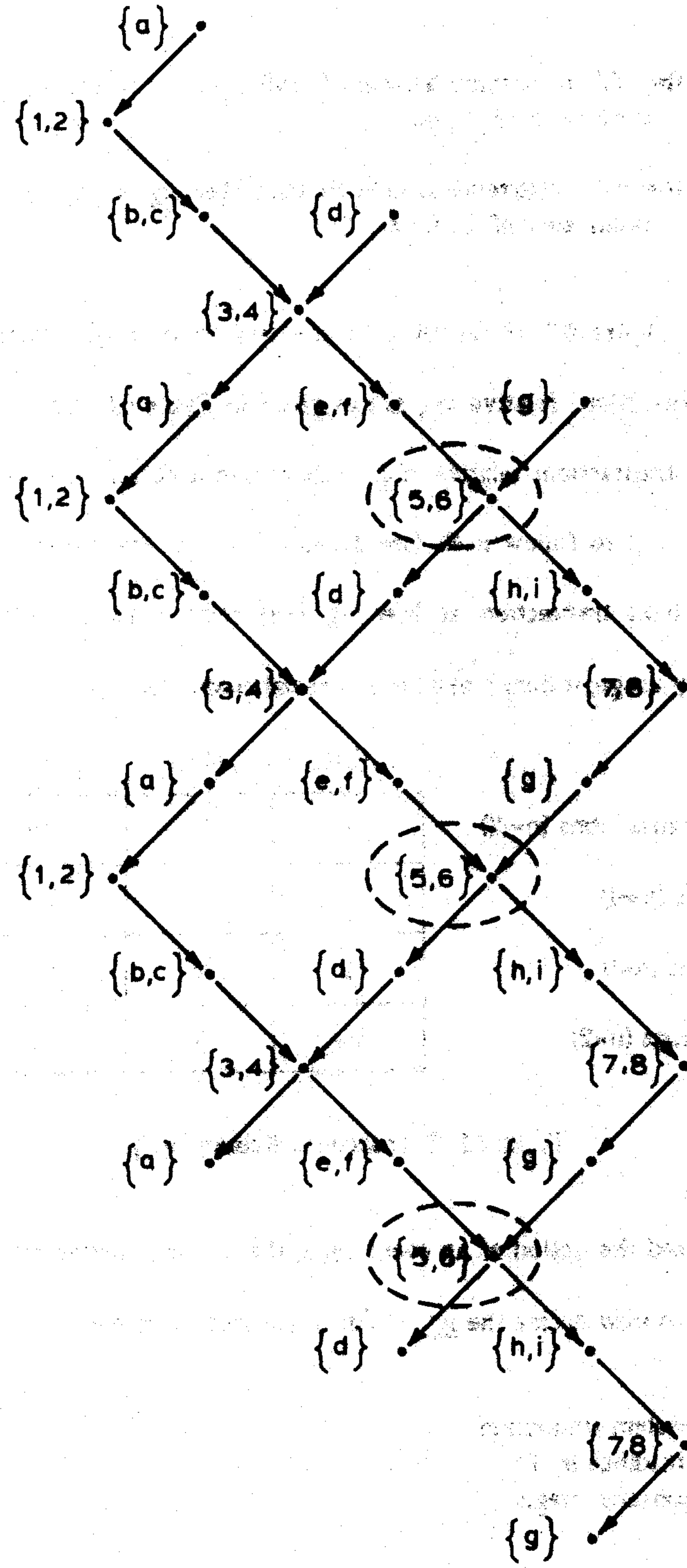


Figure 6.4 Corresponding Control Simulation

For n<o, the n'th occurrence associated with Meeting m preceding q exists and is an occurrence of Event e.

For n>o, the n'th occurrence associated with Meeting m following q exists and is an occurrence of Event e.

In the simulation of Figure 6.3, if we let q be the first (and only) occurrence of Event 4 as indicated, then the transactions relative to q are as given in Table 6.1. Note that for $n \leq -3$ and for $n \geq 3$, there are no n'th transactions relative to q. Note also that because an occurrence is considered both to precede itself and to follow itself (see definition in Section 2.3), Event 4 is both the last transaction and the next transaction at Meeting {3,4} relative to q. Although this may not correspond to ordinary usage, it does make the mathematics simpler.

Meetings	{1,2}	{3,4}	{5,6}	{7,8}
Second-to-last Transactions (n=-2)	1	3	none	none
Last Transactions (n=-1)	2	4	5	none
Next Transactions (n=1)	1	4	6	8
Second Transactions (n=2)	none	3	5	7

Table 6.1 Transactions Relative to q

Having introduced the notion of an event being the n'th transaction at some meeting relative to some instance, we can now define the set of n'th transactions relative to an instance.

Definition: If T is a system simulation,
q is an instance in T,
n is a nonzero integer,

then,

$t_n(q,T) = \{e \in E \mid$ Within T, e is the n'th transaction at $[e]_{mc}$ relative to $q\}$

If we let T be the simulation in Figure 6.3 and q be as defined above, then we have the following values for $t_n(q,T)$.

$$t_{-2}(q,T) = \{1,3\}$$

$$t_{-1}(q,T) = \{2,4,5\}$$

$$t_1(q,T) = \{1,4,6,8\}$$

$$t_2(q,T) = \{3,5,7\}$$

$$t_n(q,T) = \phi \quad \text{for } n \leq -3 \text{ and for } n \geq 3$$

The values of $t_n(q,T)$ are the things about which we're going to do our predicting and postdicting.

6.3. Extendible Simulations

The problems of predicting and postdicting system behavior are greatly simplified if the system simulations under consideration are 'extendible'.

Definition: A system simulation T is said to be forwards (backwards) <u>extendible</u> iff for each instance q in T and each positive integer k there exists a second system simulation T′ with an occurrence q' such that,

(a) $\hat{q}' = \hat{q}$

(b) for $1 \leq n \leq k$ $(-k \leq n \leq -1)$ and $\forall m \in E^*$:

$$t_n(q',T') \cap m \neq \phi$$

'T′ has an n'th transaction relative to q' for every meeting.'

(c) $t_n(q,T) \subseteq t_n(q',T')$

'The set of n'th transactions in T relative to q is contained within the set of n'th transactions in T′ relative to q'.'

Extendibility corresponds to the absence of 'deadlock' in the initialized system net.

We can state a necessary condition and some sufficient conditions for extendibility. From earlier work [4], we know that in an event graph, no event contained in a blank circuit can ever occur. Therefore, if any system simulation is to be extendible, either forwards or backwards, the initialized control structure must be free of blank circuits. Suppose that this is the case. Then the only way for the system net to 'hang up' in the forwards (backwards) direction is for there to be a pattern of holdings on the input (output) links of some meeting such that no event in the meeting is forwards (backwards) enabled by those holdings. As an example of a backwards 'hang up', consider the simulation in Figure 6.5. It is a system simulation for the half adder of Figure 3.15 with States *h* and *f* as initial conditions. Because the holdings of *h* and *f* do not backwards enable any event in Meeting {5,6,7,8}, the simulation is not backwards extendible. The following are sufficient conditions for all system simulations to be forwards extendible.

(a) Z^* has no blank circuits.

(b) $\forall m \in E^*$: If a set A consists of exactly one state from each each input link of m, then $\exists e \in m$: ${}^\bullet e = A$

The following are sufficient conditions for all system simulations to be backwards extendible.

(a) Z^* has no blank circuits.

(c) $\forall m \in E^*$: If a set A consists of exactly one state from each output link of m, then $\exists e \in m$: $e^\bullet = A$

Except for the situation noted above, the initialized systems in Sections 3.7 all satisfy these conditions.

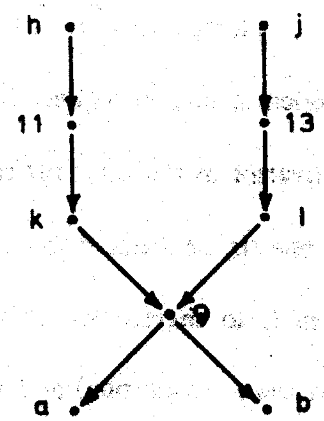


Figure 6.5 A Simulation that is Not Backwards Extendible

Before proceeding, a word about the phenomenon of 'deadlock' is in order. In the past, deadlock in a Petri net has been used to represent the corresponding notion in an actual system. This approach is not suited to the theory presented here. If we interpret occurrences of events as representing the passage of time as described in Section 5.5, then deadlock in the system net would correspond to 'time standing still'. Since this is not our intent, extendibility is a reasonable assumption. But the question arises as to how the phenomenon of deadlock is to be represented. Since deadlock is a 'mode' of behavior, there will probably be a mode (as defined formally) corresponding to any deadlock situation.

6.4. Prediction Graphs and Postdiction Graphs:

Our efforts in this chapter are concerned with the following problem.

> We know that *q* is an instance in some system simulation and that *q* is associated with the alternative class *c*. If we also knew which element in *c q* was an instance of, what would this additional knowledge tell us about the possible patterns of transactions prior to *q* and subsequent to *q*?

That additional knowledge allows us to identify an element from among its alternatives. But this is exactly the same thing that our notion of information content does. So the 'information content' of the additional knowledge is equivalent to the information content of the element identified. Anything that can be deduced from one can be deduced from the other.

Our approach to the problem is to characterize all the ways in which the information associated with q could have gotten there (postdiction) and all the ways in which it could have emanated from there (prediction). Information content consists of a set of excluded modes. Because information is 'additive', we can treat each mode in the information content of q separately and then merge the results. Associated with each exclusion, there are two subsets of the simulation containing q. One subset traces the exclusion into the past and the other traces it into the future. These two subsets determine partial histories of the transactions prior to q and subsequent to q. In general, there will be several such partial histories possible for each direction. In fact, some of the partial histories may be extendible arbitrarily far, in which case there may be an infinite number of distinct partial histories possible. But since we're dealing with finite systems, there is a finite way of characterizing the set of backwards and forwards partial histories. The cones described in Section 5.3 can be used to 'slice up' the subsets that generate the partial histories. This produces a finite set of 'history segments' as shown in Figure 6.8. This approach is especially advantageous since the n'th transactions relative to q are determined by the occurrences in the n'th history segment relative to q (see Corollary 5.1). 'Postdiction graphs' and 'prediction graphs' reflect the different ways in which history segments may be connected together.

In order to construct postdiction graphs and prediction graphs, some preliminary definitions are required.

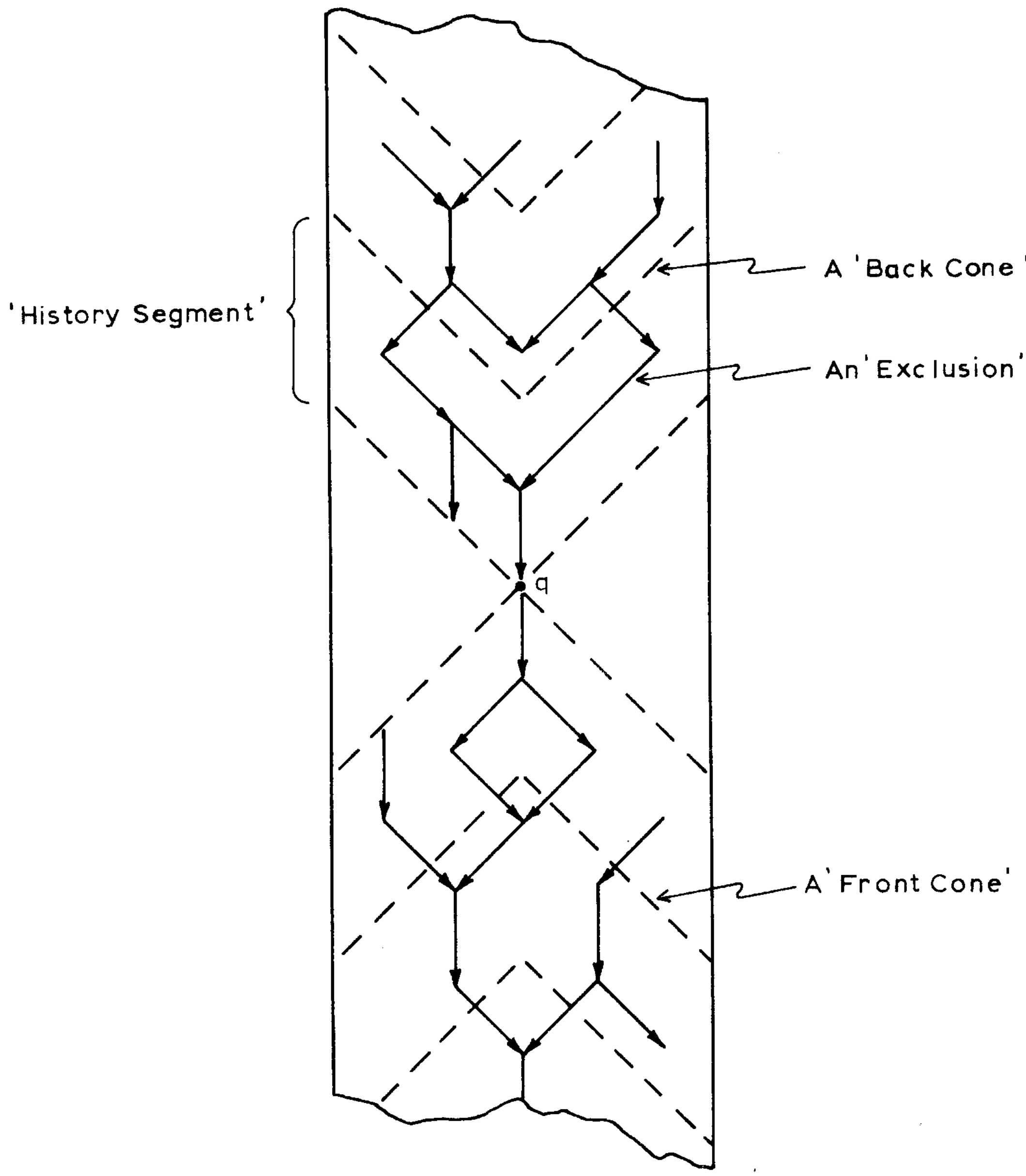


Figure 6.6 Subnets Associated with Excluded Mode

Definition: In a directed graph, a <u>chain</u> is a sequence of arcs such that each arc in the sequence has one endpoint in common with its predecessor and its other endpoint in common with its successor. (The difference between a path and a chain is that a path must traverse an arc only in the forwards direction, while a chain may traverse an arc in either direction.) The set of chains of the directed graph G is denoted C(G).

Definition: If c is a chain in a directed graph and xyz is a subsequence of c, then this subsequence counts as a <u>forwards (backwards) crossing</u> of y iff there is an arc leading from x to y (y to x) and an arc leading from y to z (z to y).

Definition: If c is a chain in the directed graph G and A is a set of vertices in G, then

$\|c\|_A$ is the number of forwards crossings in c of an element A minus the number of backwards crossings in c of an element in A.

Definition: For $m \in E^*$ and $M \in \mathfrak{M}$,

$$v^-(m,M) = \{e \in E \mid \exists c \in C(N)\colon {}^{\bullet}c = e \wedge c^{\bullet} \in m \wedge X_c \cap X_M = \phi \wedge \|c\|_{\cup \phi_{Z^*}^-(m)} \geq 0\} \qquad \text{(a)}$$

'For each event e in $v^-(m,M)$, there exists a chain c from e to an event in m such that c does not intersect the mode M and the number of forwards crossing of states in $\cup\phi_{Z^*}^-(m)$† by c is greater than or equal to the number of backwards crossings.'

$$v^+(m,M) = \{e \in E \mid \exists c \in C(N)\colon {}^{\bullet}c \in m \wedge c^{\bullet} = e \wedge X_c \cap X_M = \phi \wedge \|c\|_{\cup \phi_{Z^*}^+(m)} \geq 0\} \qquad \text{(b)}$$

'For each event e in $v^+(m,M)$, there exists a chain c form an event in m to e such that c does not intersect the mode M and the number of forwards crossings of states in $\cup\phi_{Z^*}^+(m)$ by c is greater than or equal to the number of backwards crossings.'

$v^-(m,M)$ is the set of events that can be contained in a backwards history segment for Meeting m and Mode M. $v^+(m,M)$ is the set of events that can be contained in a forwards history segment for Meeting m and Mode M.

---

† $\phi_{Z^*}^-(m)$ is the set of links in the back cone of Meeting m. $\cup\phi_{Z^*}^-(m)$ is the set of states belonging to those links.

The requirements given in the following definition will be used to generate the subnets of the system net that correspond to the history segments.

Definition: For a subnet R of the system net, a meeting m, and a mode M, we define the following requirements,

(1a) $\phi \subset E_R \subseteq v^-(m,M)$

(1b) $\phi \subset E_R \subseteq v^+(m,M)$

(2) $\forall a \in E^+$: $|a \cap E_R| \leq 1$

'R contains no more than one event from each meeting.'

(3) $S_R = ({}^{\bullet}E_R \cup E_R{}^{\bullet}) \cap (S - S_M)$

'A state is contained in R iff it is adjacent to an event in R and is not contained in the mode M.'

(4) $F_R = F \cap (S_R \times E_R \cup E_R \times S_R)$

'The arcs of R consist of those arcs in N that connect elements in R.'

(5a) $\forall s \in (S_R - \cup\phi_{Z^{\bullet}}{}^-(m))$: $({}^{\bullet}s)_R = (s^{\bullet})_R = 1$

'Within R, each state in $(S_R - \cup\phi_{Z^{\bullet}}{}^-(m))$ has exactly one input event and one output event.'

(5b) $\forall s \in (S_R - \cup\phi_{Z^{\bullet}}{}^+(m))$: $({}^{\bullet}s)_R = (s^{\bullet})_R$

'Within R, each state in $(S_R - \cup\phi_{Z^{\bullet}}{}^+(m))$ has exactly one input event and one output event.'

The functions contained in the next definition correspond to the front and back boundaries of a history segment.

Definition: For $Q \subseteq E$, $m \in E^*$, and $M \in \mathfrak{M}$,

$$b^-(Q,m,M) = {}^{\bullet}Q \cap (\cup\phi_Z*^-(m)) \cap (S-S_M)$$

$$f^-(Q,m,M) = Q^{\bullet} \cap (\cup\phi_Z*^-(m)) \cap (S-S_M)$$

$$b^+(Q,m,M) = {}^{\bullet}Q \cap (\cup\phi_Z*^+(m)) \cap (S-S_M)$$

$$f^+(Q,m,M) = Q^{\bullet} \cap (\cup\phi_Z*^+(m)) \cap (S-S_M)$$

We're now ready for postdiction graphs and prediction graphs.

Definition: The <u>postdiction graph for Meeting m and Mode m</u> is the graph $\langle u^-(m,M), w^-(m,M)\rangle$ where,

$$u^-(m,M) = \{E_R \mid R \subseteq N \text{ and } R \text{ satisfies Requirements 1a,2,3,4 and 5a with respect to } m \text{ and } M\}$$

$$w^-(m,M) = \{\langle A,B\rangle \in (u^-(m,M))^2 \mid f^-(A,m,M) \cap b^-(B,m,M) \neq \phi\}$$

Definition: The <u>prediction graph for Meeting m and Mode M</u> is the graph $\langle u^+(m,M), w^+(m,M)\rangle$ where

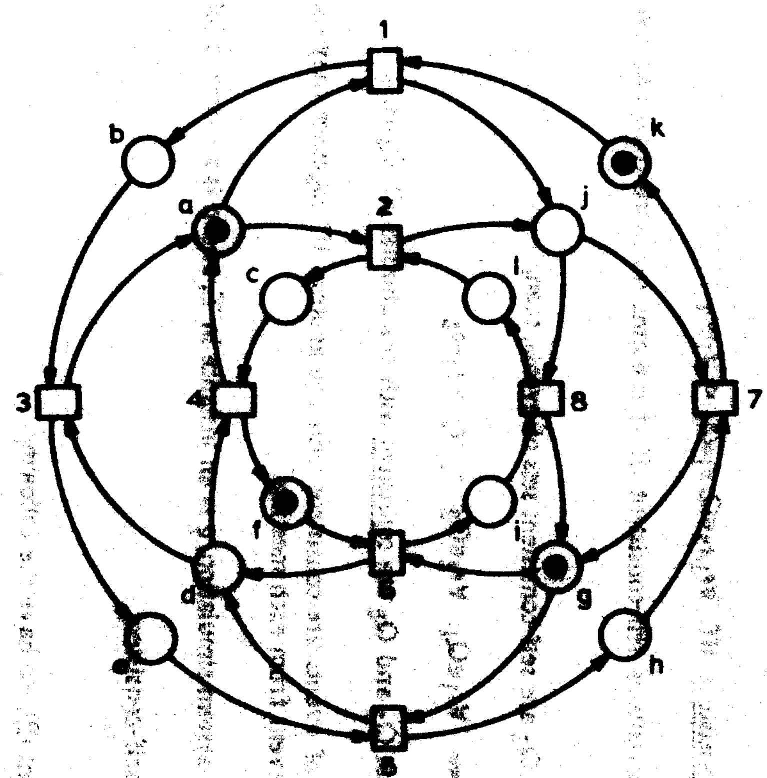
$$u^+(m,M) = \{E_R \mid R \subseteq N \text{ and } R \text{ satisfies Requirements 1b,2,3,4, and 5b with respect to } m \text{ and } M\}$$

$$w^+(m,M) = \{\langle A,B\rangle \in (u^+(m,M))^2 \mid f^+(A,m,M) \cap b^+(B,m,M) \neq \phi\}$$

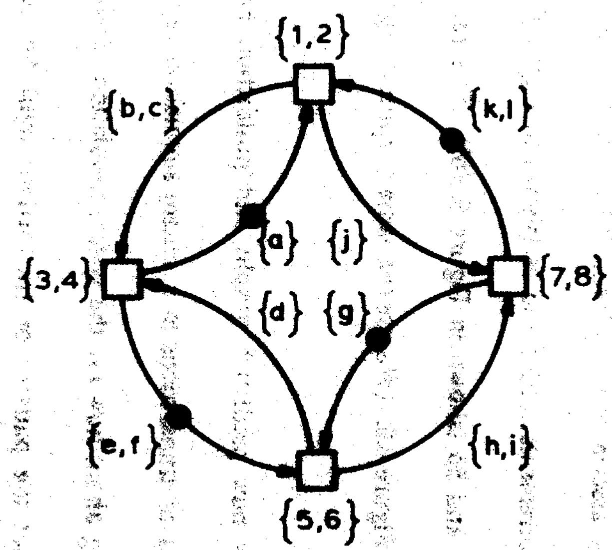
To help clarify these ideas, we'll work through an example. Of the three systems considered above, the circulating bit pipeline is the most interesting from the standpoint of prediction and postdiction. We've redrawn its initialized system net and its initialized control structure in Figure 6.7. (The parts and modes are shown in Figure 3.14.) Let's consider the meeting {5,6}. For m={5,6}, we have,

$$\phi^-(m) = \{\{a\},\{d\},\{h,i\},\{k,l\}\} \text{ and } \cup\phi^-(m) = \{a,d,h,i,k,l\}$$

$$\phi^+(m) = \{\{b,c\},\{e,f\},\{g\},\{j\}\} \text{ and } \cup\phi^+(m) = \{b,c,e,f,g,j\}$$



(a) Initialized System Net



(b) Initialized Control Structure

Figure 6.7 Circulating Bit Pipeline

Now suppose that M is the mode associated with Events 2,4,6 and 8. Then there are two subnets of the system net satisfying Requirements Ia,2,3,4 and 5a (shown in Figure 6.8(a)) and two subnets satisfying Requirements Ib,2,3,4, and 5b (shown in Figure 6.8(b)). The resulting postdiction and prediction graphs are shown in Figure 6.9. (We use the same graphical representation as for state graphs.)

Our task now is to show how prediction graphs and postdiction graphs can be used to predict and postdict the patterns of transactions in a system simulation relative to some instance. Of course, we can't say anything about how far the system simulation extends, either forwards or backwards, but we can say that the patterns of transactions must be 'consistent' with certain requirements.

Definition: For $Q \subseteq E$,

Q is <u>self-consistent</u> iff $\forall e_1, e_2 \in Q$: $e_1 \propto e_2 \Rightarrow e_1 = e_2$

'A set of events is self-consistent iff it contains no more than one event from each meeting.'

Definition: If Q_1 and Q_2 are self-consistent sets of events, then,

$Q_1 \approx Q_2 \Leftrightarrow \forall e_1 \in Q_1$: $\forall e_2 \in Q_2$: $e_1 \propto e_2 \Rightarrow e_1 = e_2$

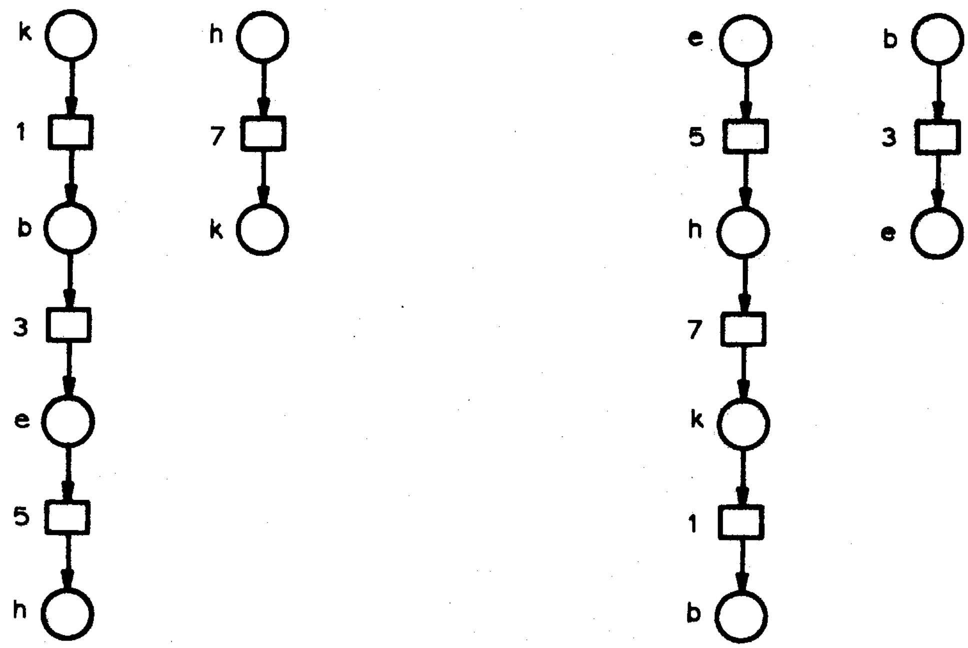
We say that Q_1 and Q_2 are <u>consistent</u> with one another iff $Q_1 \approx Q_2$.

'Two sets of events are consistent with one another iff together they contain no more than one event from each meeting.'

Property 6.1: If T is a system simulation, q is an instance in T, and n is a nonzero integer, then,

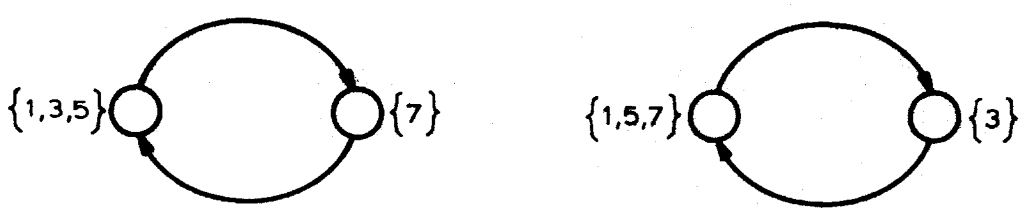
$t_n(q,T)$ is self-consistent

And, from Requirement (2), we have the following.



(a) **Backwards 'History Segments'** (b) **Forwards 'History Segments'**

Figure 6.8



(a) **Postdiction Graph for Meeting {5,6} and Mode {2,4,6,8}**

(b) **Prediction Graph for Meeting {5,6} and Mode {2,4,6,8}**

Figure 6.9

Property 6.2: $\forall m \in E^*$: $\forall M \in \mathfrak{M}$

$\forall Q \in u^-(m,M)$: Q is self-consistent

$\forall Q \in u^+(m,M)$: Q is self-consistent

'Each set of events associated with a node in a postdiction (prediction) graph is self-consistent.'

The next four theorems comprise our results on prediction and postdiction. Unfortunately, they are quite cumbersome. They should be regarded as only first tentative steps in the area of prediction and postdiction.

Theorem 6.1: (a) If T is a backwards-extendible system simulation, q is an occurrence in T, $m=[\hat{q}]_{\approx\mathcal{C}}$, and $M \in I(\hat{q})$, then $\exists Q \in \nu^-(m,M)$:

$\hat{q} \in Q \ \wedge \ t_{-1}(q,T) \approx Q$

and there exists a backwards-extendible system simulation T' with an occurrence q' such that,

$\hat{q}' = \hat{q}$
$\forall n \in Z^-$: $t_n(q,T) \subseteq t_n(q',T')$
$\forall e \in E^*$: $t_{-1}(q',T') \cap e \neq \phi$
$Q = \nu^-(m,M) \cap t_{-1}(q',T')$

(b) If T is a forwards-extendible system simulation, q is an occurrence in T, $m=[\hat{q}]_{\approx\mathcal{C}}$, and $M \in I(\hat{q})$, then $\exists Q \in \nu^+(m,M)$:

$\hat{q} \in Q \ \wedge \ t_{+1}(q,T) \approx Q$

and there exists a forwards-extendible system simulation T' with an occurrence q' such that,

$\hat{q}' = \hat{q}$
$\forall n \in Z^+$: $t_n(q,T) \subseteq t_n(q',T')$
$\forall e \in E^*$: $t_{+1}(q',T') \cap e \neq \phi$
$Q = \nu^+(m,M) \cap t_{+1}(q',T')$

Proof: We prove just Part (a).

Because T is backwards extendible, there exists a system simulation T′ with an occurrence q' such that,

$\hat{q}' = \hat{q}$
$\forall n \in \mathbb{Z}$: $t_n(q,T) \subseteq t_n(q',T')$
$\forall a \in E^*$: $t_{-1}(q',T') \cap a \neq \phi$

Now since T is backwards extendible, it must be possible to select T′ so that it too is backwards extendible. Let R be the subnet of N defined as follows.

$E_R = v^-(m,M) \cap t_{-1}(q',T')$
$S_R = ({}^\bullet E_R \cup E^\bullet{}_R) \cap (S - S_M)$
$F_R = F \cap (S_R \times E_R \cup E_R \times S_R)$

We can deduce the following about R:

(a) $\phi \subsetneq E_R \subseteq v^-(m,M)$	$\hat{q} \in E_R$ and def. of E_R
(b) $\forall a \in E^*$: $\lvert a \cap E_R \rvert \leq 1$	$E_R \subseteq t_{-1}(q',T')$
(c) $S_R = ({}^\bullet E_R \cup E^\bullet{}_R) \cap (S - S_M)$	def. of S_R
(d) $F_R = F \cap (S_R \times E_R \cup E_R \times S_R)$	def. of F_R
(e) $\forall s \in (S_R - \cup \phi_{Z^*}{}^-(m))$: $({}^\bullet s)_R = (s^\bullet)_R = 1$	def. of R and Cor. 4.1

In other words, R satisfies Requirements 1a,2,3,4, and 5a. Thus $E_R \in u^-(m,M)$. We know that $\hat{q}$ is an element of both $v^-(m,M)$ and $t_{-1}(q',T')$, and therefore, $\hat{q} \in E_R$. Finally, because $t_{-1}(q,T)$ and E_R are both subsets of $t_{-1}(q',T')$, it follows that $t_{-1}(q,T) \approx E_R$. □

Theorem 6.2: (a) If T is a backwards-extendible system simulation, h is a holding in T, $M \in I(\hat{h})$, and m is the unique input meeting of $[\hat{h}]_\alpha$, then $\exists Q \in u^-(m,M)$:

$\hat{h} \in (Q \cap m)^\bullet \ \wedge \ t_{-1}(h,T) \approx Q$

and there exists a backwards-extendible system simulation T′ with a holding h′ such that

$\hat{h}' = \hat{h}$
$\forall n \in \mathbb{Z}$: $t_n(hT) \subseteq t_n(h',T')$
$\forall a \in E^*$: $t_{-1}(h',T) \cap a \neq \phi$
$Q = v^-(m,M) \cap t_{-1}(h',T')$.

(b) If T is a forwards-extendible system simulation, h is a holding in T, $M \in I(\Sigma)$, and m is the unique output meeting of $[\Sigma]_{oc}$, then $\exists Q \in \pi^{+}(m,M)$:

$$\hat{h} \in {}^{*}(Q \cap m) \wedge t_{+1}(h,T) \approx Q$$

and there exists a forwards-extendible system simulation T′ with a holding h' such that,

$$\hat{h}' = \hat{h}$$
$$\forall n \in Z^{+}:\ t_n(h,T) \subseteq t_n(h',T')$$
$$\forall e \in E^{+}:\ t_{+1}(h',T') \cap e \neq \phi$$
$$Q = \pi^{+}(m,M) \cap t_{+1}(h',T')$$

Proof: We prove just Part (a).

Because T is backwards extendible, there exists a system simulation T′ with a holding h' such that,

$$\hat{h}' = \hat{h} \quad (1)$$
$$\forall n \in Z^{-}:\ t_n(h',T') \quad (2)$$
$$\forall e \in E^{+}:\ t_{-1}(h',T') \cap e \neq \phi \quad (3)$$

Now since T is backwards extendible, it must be possible to select T′ so that it too is backwards extendible. Let q be the initiating occurrence of h' in T′. By Line (3), q exists. We have $m = [\Sigma]_{oc}$, and by Corollary [illegible] the same construction as in the proof of Theorem [illegible], we get $\exists Q \in \pi^{-}(m,M)$:

$$\hat{q} \in Q \quad \text{and} \quad Q = \pi^{-}(m,M) \cap t_{-1}(q,T')$$

Because $\hat{q} \in (Q \cap m)$ and $\hat{q} = \hat{h}$,

$$\hat{h} \in (Q \cap m)^{*} \quad (4)$$

Because q initiates h',

$$t_{-1}(h',T') = t_{-1}(q,T')$$

It follows that,

$$Q = \pi^{-}(m,M) \cap t_{-1}(h',T') \quad (5)$$

And finally, because $t_{-1}(h,T)$ and Q are both subsets of $t_{-1}(h',T')$,

$$t_{-1}(h,T) \approx Q \quad (6)$$

Lines 1-6 comprise the desired result. □

Definition: Within the context of a prediction or postdiction graph $<u,w>$, we write $A\nabla B$ to mean $<A, B>\in w$. For $A\in u$,

$^{\nabla}A = \{B|B\nabla A\}$
$A^{\nabla} = \{B|A\nabla B\}$

Theorem 6.3 (a) If T is a system simulation, q is an occurrence in T, $m = [\hat{q}]_{\alpha}$, $M\in I(\hat{q})$, $Q\in u^{-}(m,M)$, k is a negative integer, and there exists a backwards-extendible system simulation T′ with an occurrence q' such that,

$\hat{q}' = \hat{q}$
$\forall n\in\mathbb{Z}^{-}$: $t_n(q,T) \subseteq t_n(q',T')$
$\forall a\in E^{*}$: $t_k(q',T') \cap a \neq \phi$
$Q = v^{-}(m,M) \cap t_k(q',T')$

then for $^{\nabla}Q \neq \phi$, $\exists U\in{}^{\nabla}Q$:

$t_{k-1}(q,T) \approx U$

and there exists a backwards-extendible system simulation T′′ with an occurrence q'' such that

$\hat{q}'' = \hat{q}'$
$\forall n\in\mathbb{Z}^{-}$: $t_n(q',T') \subseteq t_n(q'',T'')$
$\forall a\in E^{*}$: $t_{k-1}(q'',T'') \cap a \neq \phi$
$U = v^{-}(m,M)\cap t_{k-1}(q'',T'')$

(b) If T is a system simulation, q is an occurrence in T, $m=[\hat{q}]_{\alpha}$, $M\in I(\hat{q})$, $Q\in u^{+}(m,M)$, k is a positive integer, and there exists a forwards-extendible system simulation T′ with an occurrence q' such that

$\hat{q}' = \hat{q}$
$\forall n\in\mathbb{Z}^{+}$: $t_n(q,T) \subseteq t_n(q',T')$
$\forall a\in E^{*}$: $t_k(q',T') \cap a \neq \phi$
$Q = v^{+}(m,M) \cap t_k(q',T')$

then for $Q^{\nabla} \neq \phi$, $\exists U \in Q^{\nabla}$:

$$t_{k+1}(q,T) = U$$

and there exists a forwards-extendible system simulation T′′ with an occurrence q'' such that,

$$\hat{q}'' = \hat{q}'$$
$$\forall n \in Z^{+}:\ t_n(q',T') \subseteq t_n(q'',T'')$$
$$\forall e \in E^{*}:\ t_{k+1}(q'',T'') \cap e \neq \phi$$
$$U = v^{+}(m,M) \cap t_{k+1}(q'',T'')$$

Proof: We prove Part (a).

Because T′ is backwards extendible, there exists a system simulation T′′ with an occurrence q'' such that,

$$\hat{q}'' = \hat{q}'$$
$$\forall n \in Z^{-}:\ t_n(q',T') \subseteq t_n(q'',T'')$$
$$\forall e \in E^{*}:\ t_{k-1}(q'',T'') \cap e \neq \phi$$

Because T′ is backwards extendible, it must be possible to select T′′ so that it too is backwards extendible. Assume that $t_k(q,T) = Q$ and ${}^{\nabla}Q \neq \phi$. Let E_R be the subset of N defined as follows.

$$E_R = v^{-}(m,M) \cap t_{k-1}(q'',T'')$$
$$S_R = ({}^{\bullet}E_R \cup E_R{}^{\bullet}) \cap (S - S_M)$$
$$F_R = F \cap (S_R \times E_R \cup E_R \times S_R)$$

Now because $t_k(q',T') \subseteq t_k(q'',T'')$ and both sets contain an event from each meeting,

$$t_k(q',T') = t_k(q'',T'')$$

We now have,

$$Q = v^{-}(m,M) \cap t_k(q'',T'')$$
$$E_R = v^{-}(m,M) \cap t_{k-1}(q'',T'')$$

It is a straightforward matter to show that,

$$f^{-}(E_R,m,M) = b^{-}(Q,m,M)$$

Since $^{\nabla}Q \neq \phi$, it follows that $b^{-}(Q,m,M) \neq \phi$ and $E_R \neq \phi$. Using the arguments in the proof of Theorem 6.1, we then have,

$E_R \in u^{-}(m,M)$

Thus, $E_R \in {}^{\nabla}Q$.

It remains to be shown that $t_{k-1}(q,T) \approx E_R$. This follows from the fact that $t_{k-1}(q,T)$ and E_R are both subsets of $t_{k-1}(q'',T'')$. □

Theorem 6.4: (a) If T is a system simulation, h is a holding in T, m is the unique input meeting of $[\hat{h}]_{\alpha}$, $M \in I(\hat{h})$, $Q \in u^{-}(m,M)$, k is a negative integer, and there exists a backwards-extendible system simulation T′ with a holding h' such that,

$\hat{h}' = \hat{h}$
$\forall n \in \mathbb{Z}^{-}$: $t_n(h,T) \subseteq t_n(h',T')$
$\forall a \in E^{*}$: $t_k(h',T') \cap a \neq \phi$
$Q = v^{-}(m,M) \cap t_k(h',T')$

then for $^{\nabla}Q \neq \phi$, $\exists\, U \in {}^{\nabla}Q$:

$t_{k-1}(q,T) \approx U$

and there exists a backwards-extendible system simulation T′′ with a holding h'' such that,

$\hat{h}'' = \hat{h}'$
$\forall n \in \mathbb{Z}^{-}$: $t_n(h',T') \subseteq t_n(h'',T'')$
$\forall a \in E^{*}$: $t_{k-1}(h'',T'') \cap a \neq \phi$
$U = v^{-}(m,M) \cap t_{k-1}(h'',T'')$

(b) If T is a system simulation, h is a holding in T, m is the unique output meeting of $[\hat{h}]_{\alpha}$, $M \in I(\hat{h})$, $Q \in u^{+}(m,M)$ k is a positive integer, and there exists a forwards-extendible system simulation T′ with a holding h' such that,

$\hat{h}' = \hat{h}$
$\forall n \in \mathbb{Z}^{+}$: $t_n(h,T) \subseteq t_n(h',T')$
$\forall a \in E^{*}$: $t_k(h',T') \cap a \neq \phi$
$Q = v^{+}(m,M) \cap t_k(h',T')$

then for $Q^{\nabla} \neq \phi$, $\exists U \in Q^{\nabla}$:

$t_{k+1}(q,T) \approx Q$

and there exists a forwards-extendible system simulation T'' with a holding h'' such that,

$$\hat{h}'' = \hat{h}'$$
$$\forall n \in \mathbb{Z}^{+}: \; t_n(h',T') \subseteq t_n(h'',T'')$$
$$\forall a \in E^{*}: \; t_{k+1}(h'',T'') \cap a \neq \phi$$
$$U = v^{+}(m,M) \cap t_{k+1}(h'',T'')$$

Proof: Similar to that of Theorem 6.3.

As a limited illustration of the preceding results, consider the system of Figure 6.7 and the postdiction graph of Figure 6.9(a). All system simulations are backwards (and forwards) extendible. Consider Event 5. It belongs to Meeting {5,6}, and its information content contains Mode {2,4,6,8}. Now if q is any occurrence of Event 5 in a system simulation T, then from Theorems 6.1(a) and 6.3(a), we have,

$$t_{-1}(q,T) \approx \{1, 3, 5\}$$

$$t_{-2}(q,T) \approx \{7\}$$

$$t_{-3}(q,T) \approx \{1, 3, 5\}$$

$$t_{-4}(q,T) \approx \{7\}$$

$$\vdots$$

The odd-numbered transactions preceding q are consistent with {1,3,5}, while the even numbered-transactions preceding q are consistent with {7}. This checks out with the system simulation in Figure 6.10. Here we have,

$$t_{-1}(q,T) = \{1, 3, 5, 8\}$$
$$t_{-2}(q,T) = \{4, 6, 7\}$$
$$t_{-3}(q,T) = \phi$$

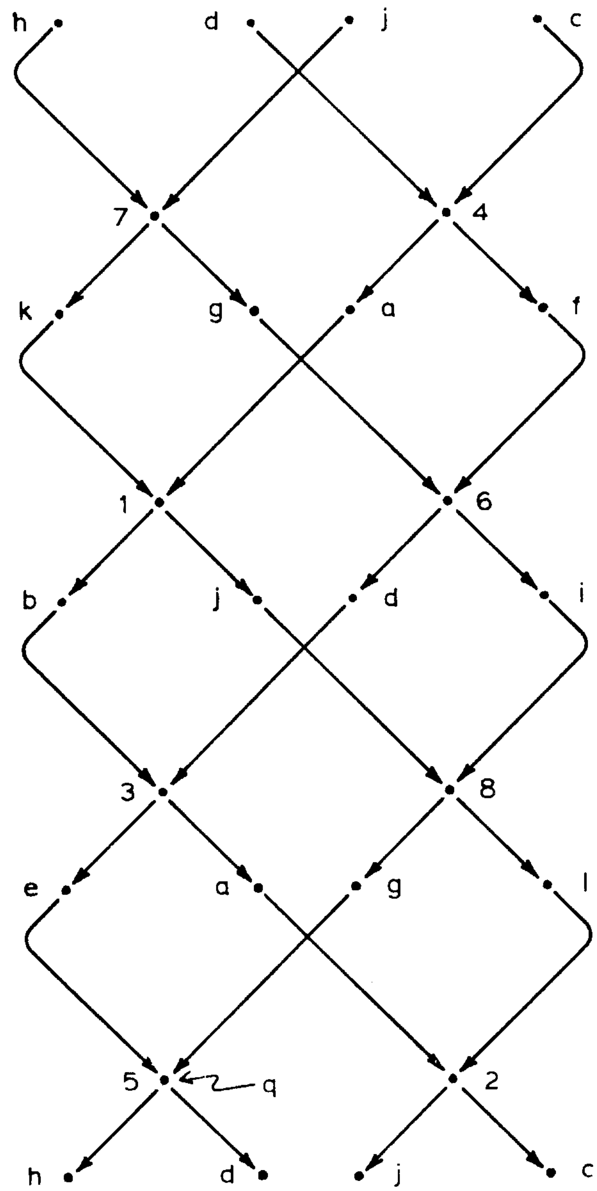


Figure 6.10 A System Simulation

CHAPTER 7

CONCLUSIONS

7.1. Evaluation:

With the theory introduced in the preceding chapters, we are now able to treat important kinds of Petri nets that were previously outside the scope of any theory. The net representing the half adder is just one example. From all indications, the class of nets contained in our theory is rich and varied.

The theory has the following advantages:

(1) The range of concepts expressible within the theory is extremely broad. Among those concepts are some that are fundamental. 'Space', 'time', 'information', and 'causality' are the most notable.

(2) The theory takes into account the distributed nature of systems. Concurrency is the key concept here, and concurrency is embedded in the fabric of the theory.

(3) Because the theory does not rely on the notion of 'total system state', the complexity of a system model is reduced significantly.

(4) Identifying the system net with the system 'hardware', and the set of initial conditions with the system 'software' is a step towards an integrated approach to both hardware and software.

(5) The techniques of the theory lend themselves to automation.

7.2. Future Work:

The work that needs to be done falls into two categories: theory and metatheory. The metatheory is concerned with four related topics: (1) foundations, (2) semantics, (3) methodology, and (4) scope.

(1) foundations - The theory we've presented depends upon five axioms. We've tried to make those axioms plausible, but clearly more work needs to be done. The goal here should be to reduce those five axioms to another set of axioms that are more or less self-evident.

(2) semantics - A number of concepts have been introduced in the theory, and we need to understand the meanings of those concepts. The two that are of the most concern are <u>parts</u> and <u>modes</u>. We've said that parts are associated with strictly sequential behavior, and that modes are associated with steady-state behavior. But we need to know much more about these concepts - in particular, how they relate to concepts already familiar to us. (Note that foundations and semantics are intertwined.)

(3) methodology - For the theory to be a practical tool, there has to be a methodology for applying the theory. A set of practical examples is necessary in establishing such a methodology.

(4) scope - The scope of a theory is the range of problems to which it is suited. We must find out for which problems the above theory is suited and for which it is not suited.

In the mathematical development of the theory, there are several areas that deserve attention.

(1) For a particular system net, there may be several ways of choosing a covering of parts and a covering of modes. We need to determine precisely the effects of those choices. We already know that the control structure and the information contents of the system elements are, in general, affected.

(2) The four theorems of Chapter 6 are quite cumbersome, and are only the first tentative steps in the area of prediction and postdiction. Much more work remains to be done. (In this area, Theorem 4.3 ought to play an important role.)

(3) The ability to predict and postdict system behavior should provide the key to answering the following questions about a system. These questions were posed in Section 1.3.

Under what conditions will a certain pattern of behavior be produced?

What are the consequences of a decision within a system?

What are the effects of a system modification?

How does behavior in one part of a system influence behavior in another part?

How do the outputs of a system depend upon the inputs? (i.e., What is the 'function' of the system?)

A systematic technique needs to be developed for each of these questions.

(4) Within this thesis, we have not touched upon probabilistic considerations. This is a major area, and one which will require considerable effort. That effort will entail relating the approach presented here with the ideas of Information Theory. In particular, it will be necessary to relate the notion of information content to Shannon's information measure.

(5) In Section 5.5, we introduced the synchrony property for control structures. This property allowed us to define instants of time. It might be interesting to investigate other possible constraints on the control structure. (These new kinds of space/time frameworks might correspond to non-euclidean spaces.)

The success of these efforts will determine the fruitfulness of the ideas presented in this thesis. In any event, we hope to have stimulated the reader to thinking about the issues raised.

REFERENCES

1. Baker, H. G., Equivalence Problems of Petri Nets, S.M. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, June 1973.

2. Berge, C., Graphs and Hypergraphs, North-Holland Publishing Co., 1973.

3. Commoner, F. G., Deadlocks in Petri Nets, Report CA-7206-2311, Applied Data Research, Inc., Wakefield, Mass., June 1972.

4. Commoner, F., A. W. Holt, S. Even, and A. Pnueli, "Marked Directed Graphs", Journal of Computer and System Sciences, Vol. 5, October 1971, pp. 511-523.

5. Furtek, F. C., Modular Implementation of Petri Nets, S.M. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, September 1971.

6. Furtek, F. C., "Asynchronous Push-Down Stacks", Computation Structures Group Memo 86, Project MAC, Massachusetts Institute of Technology, August 1973.

7. Genrich, H. J., Einfache Nicht-sequentielle Prozesse (Simple Nonsequential Processes). Bericht Nr. 37, Gesellschaft fur Mathematik und Datenverarbeitung, Bonn. 1971.

8. Hack, M. H. T., Analysis of Production Schemata by Petri Nets, Technical Report MAC-TR-94, Project MAC, Massachusetts Institute of Technology, February 1972.

9. Hack, M. H. T., "Extended State-Machine Allocatable Nets", Computation Structures Group Memo 78-1, Project MAC, Massachusetts Institute of Technology, June 1974.

10. Holt, A. W., et al., Final Report of the Information System Theory Project, Technical Report No. RADC-TR-68-305, Rome Air Development Center, Griffis Air Force Base, New York, September 1968.

11. Holt, A. W. and F. G. Commoner, Events and Conditions, Part I, Applied Data Research, Inc., New York, 1970. (Chapter I, II and IV appear in Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York, 1970, pp. 3-31.)

12. Holt, A. W., Events and Conditions, Part 2.

13. Holt, A. W., Events and Conditions, Part 3. (Reprinted in Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, pp. 33-52.)

14. Holt, A. W., "Communication Mechanics", Applied Data Research, Inc., Wakefield, Mass., 1974.

15. Holt, A. W., "Information as a System-Relative Concept", Applied Data Research, Inc., Wakefield, Mass., September 1974.

16. Jump, J. R., and P. S. Thiagarajan, "On the Equivalence of Asynchronous Control Structures," SIAM Journal of Computation, Vol. 2, No. 2, June 1973, pp. 67-87.

17. Patil, S. S., Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination among Processes, Computation Structures Group Memo 57, Project MAC, Massachusetts Institute of Technology, February 1971.

18. Petri, C. A., Communication with Automata, Supplement I to Technical Report RADC-TR-65-377, Vol. 1, Rome Air Development Center, Griffiss Air Force Base, New York, 1966. [Originally published in German: Kommunikation mit Automaten, Schriften des Rheinisch-Westfalischen Institutes fur Instrumentelle Mathematik an der Universitat Bonn, Hft. 2, Bonn, 1962.

19. Petri, C. A., "Fundamentals of a Theory of Asynchronous Information Flow", Proceedings of IFIP Congress 1962, North-Holland Publishing Co., Amsterdam, 1962.

20. Petri, C. A., "Grundsatzliches zur Beschreibung Diskreter Prozesse", Colloquium uber Automatentheorie, Birkhauser Verlag, Basel, 1967.

21. Petri, C. A., Series of talks given at Applied Data Research, Inc., Wakefield, Mass., August 1973.

22. Petri, C. A., "Concepts of Net Theory," Proceedings of Symposium on Mathematical Foundations of Computer Science, Mathematical Institute of the Slovak Academy of Sciences, High Tatras, September 1973.

23. Ramchandani, C., Analysis of Asynchronous Concurrent Systems by Petri Nets, Technical Report MAC-TR-120, Project MAC, Massachusetts Institute of Technology, February 1974.

INDEX

alternative 44
alternative classes 50
alternativeness relation for Part P 44

back cone 113
backwards conflict 97
backwards conflict classes 97
basic circuit 102
blank circuit 102
boundary 31

causal connection 34
causality relation 30
chain 156
circuit 28
coincident 31
concurrent 34
consistent 160
control structure 50

distance 142

elementary causal connections 31
elementary circuit 28
event component 26
event graph 24
event-graph decomposable 26
events 24
extendible 151

flow relation 24
follows 34
forwards conflict 97
forwards conflict classes 97
front cone 113
fundamental circuit 132

holdings 30

image 34
in phase 131
information content 75

initial conditions 24
initialized Petri net 24
initiate 31
input events 24
input states 24
instant of time 139
interval of time 139

link 50

meeting 50

net 23
nonsynchronous 129

occurrences 30
output events 24
output states 24

Petri net 24
phase relation 131
phase transitions 131
phases 131
postconditions 24
precedes 34
preconditions 24

self-consistent 160
simulation 30
simultaneity classes 139
simultaneity relation 139
simultaneous 139
state component 26
state graph 24
state-graph decomposable 26
states 24
strand 35
subnet 23
synchronic delay 108
synchronic distance 125
synchronous 129
synchronous system 129

synchrony property 129
system 40
system space 129

terminate 31
time 139
time interval 136
token loading 102
transaction 146