COORDINATION OF ASYNCHRONOUS EVENTS

Suhas Shrikrishna Patil

June 1970

PROJECT MAC

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge Massachusetts 02139

This empty page was substituted for a blank page in the original document.

COORDINATION OF ASYNCHRONOUS EVENTS*

Abstract

The way activity in a system proceeds is that events occur as a result of some conditions and lead to some new conditions which make other events possible. Often it is necessary to coordinate such events to ensure proper behavior. Coordination nets for representing such coordinations and physically realizable structures for enforcing such coordinations are presented. These structures are modular and can be mechanically derived from the coordination nets. Coordinations involved in concurrent management of resources are also discussed.

*This report reproduces a thesis of the same title submitted to the Department of Electrical Engineering, Massachusetts Institute of Technology, in partial fulfillment of the requirements for the degree of Doctor of Science, May, 1970.

This empty page was substituted for a blank page in the original document.

ACKNOWLEDGEMENT

I wish to thank Professor Jack B. Dennis, my supervisor, for his able guidance, and Professors Robert M. Fano and Robert M. Graham, my readers, for their helpful suggestions. I am grateful to my office-mates Prakash Hebalkar and Murray Edelberg for invaluable interaction, and to Carla Vogt for reading the manuscript. Dr. Anatol W. Holt merits thanks for introducing me to Petri nets and for valuable technical discussions. Thanks are due to Professor Chung C. Liu for his encouragement and counsel. Especial thanks are due to Miss Vienna Berardinelli for diligent typing of the manuscript.

Work reported herein was supported in part by Project MAC, an M.I.T. research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-4102(01).

TO MY PARENTS AND TEACHERS

Work reported herein was supported in part by Project MAC, an M.I.T. research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Reseach Contract Nonr-4102(01). Reproduction of this report, in whole or in part, is permitted for any purpose of the United States Government.

This empty page was substituted for a blank page in the original document.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Events and Conditions

The way in which the activity in a computer system proceeds is that actions, called events, take place in response to conditions of the system and lead to new conditions which in turn make other actions possible. For example the condition that a user is waiting for a processor together with the condition that a processor is free to be used may lead to granting of use of the processor to the user. The action of granting the processor to the user is said to be an occurrence of the event "the processor is granted to the user". As a consequence of this event the condition "the user is using the processor" comes into force. This condition continues to hold until the occurrence of an event corresponding to the release of the processor by the user. An event, which depends on certain conditions, is said to be an asynchronous event if the occurrence of the event is indefinitely delayed (in the temporal sense) until such conditions come into force, unlike a synchronous event which occurs at a definite time when the conditions are believed to hold regardless of whether the conditions actually hold or not. In the above example, the event "the processor is granted to the user" is indefinitely delayed until the processor requested by the user becomes free to be used.

1.2 Coordination of Events

Some conditions in a computer may be in conflict, e.g. the conditions that user 1 is using a processor and that user 2 is using a processor are in conflict when there is only one processor and the processor can correctly handle no more than one user at a time. Conditions like these should be prevented from occurring together for the proper operation of the computer. This requires that while one of the above condition holds, the event that brings the other condition into force should be prevented from occurring. Thus there is need to coordinate the occurrences of events.

There are many instances in a computer system requiring coordination of events. For example in Multics [1], processors, memory modules, input/output controllers and other devices are multiplexed. For proper multiplexing, the actions of connecting the devices to different users must be coordinated. Another instance requiring coordination is a computer network where independently operating computer systems communicate through a communication network.

Even though these problems take many forms and look different, they can be reduced to the problem of coordination of the occurrences of events. In order to deal with these problems satisfactorily, a suitable scheme for representing the coordination and suitable schemes for implementing such coordination are needed.

1.3 Related Work

Most earlier work has been in connection with the representation of computations, information systems and coordination of concurrent processes. Dijkstra [2] uses 'semaphores' for coordinating sequential processes, Dennis and Van Horn [3] use 'fork' and 'join' as primitives for coordinating processes within a computation. In terms of models for computations, Miller and Karp parallel program schemata [4], Slutz flow graph schemata [5] and Luconi computation schemata [6] represent a few that have been devised. These works are directed towards representation of parallel computations and investigating output functionality of such computations, rather than studying representation of the coordination of events and finding a systematic implementation for the coordination specified. In this respect the work of Petri on representation of physical phenomena [7] and the work of Holt on Occurrence Systems [8] is most significant because of their usefulness in representing the coordination of events. Significant as these works are, they fall short in two important respects: i) the complexity of representation of coordination in their models increases at an undesirable rate with the size of the problem and ii) no way for implementing the coordination is given. This thesis presents a representation scheme in which it is easier to formalize coordination, in which the complexity of representation does not increase as rapidly, and gives a method for systematically deriving asynchronous modular structures for enforcing the coordination.

Some earlier work on asynchronous modular structures was done by Muller and Bartky[9,10]. Clark and his associates are also actively working in this area [11] . The approach to asynchronous structures presented in this thesis was inspired by the work of Muller.

1.4 Plan of the Thesis

In Chapter 2 of the thesis, alternative representation schemes for representing coordination are studied with regard to their limitations and the causes of the limitations, and a representation scheme called coordination nets is presented. The choice of this scheme for representation of coordination is motivated by the precision and convenience it offers in formalizing coordination. For the purpose of implementing coordination, certain coordination nets called homogeneous coordination nets are of great importance. Fortunately coordination nets which are not homogeneous can be transformed into equivalent homogeneous coordination nets. In Chapter 3, homogeneous coordination nets are defined and a systematic transformation of coordination nets into homogeneous coordination nets is presented.

In Chapter 4, asynchronous modular structures, called coordination structures, for implementing the coordination are presented. These structures, built from a small number of tupes of modules specified in the chapter, can be systematically derived from the coordination nets. The chapter presents a systematic derivation of the structures starting from the coordination nets.

Chapter 5 of the thesis is devoted to showing that the coordination structures presented in Chapter 4 correctly implement the coordination nets. The proofs in this chapter involve examination in detail of the behaviour of the coordination structures, and may be skipped without loss of continuity.

In Chapter 6, coordination required in concurrent management of resources is studied and the structure of an arbiter for performing such coordination is presented. In Chapter 7 conclusions and suggestions for related research are presented.

CHAPTER 2

REPRESENTATION OF COORDINATION

2.1 Introduction

In order to be able to deal with problems involving coordination of events, a satisfactory representation scheme is needed to represent the coordination. In this chapter different representation schemes are examined and a scheme satisfactory from the points of view of preciseness and convenience is developed.

State diagrams used in switching circuit theory are unsuited as a representation scheme because in such diagrams the representation of concurrent activity is not natural and because their complexity increases rapidly with the size of the problem.

A partial ordering of the events can represent coordination in many cases, but a partial ordering is of limited use because of its limitations with respect to the class of coordinations that it can represent. A study of the use of partial orderings in representation of coordination and a study of the limitations of partial orderings in this regard, give useful insight into the requirements of the desired representation scheme.

2.2 Partial Orderings

In using partial orderings to represent coordinations, the events which are to be coordinated are represented as nodes and the coordination among them is represented by defining a partial ordering on these nodes. The events corresponding to the nodes which are ordered must occur in that order but the events corresponding to nodes which are not ordered may occur concurrently. In the example shown in figure 2.1 events e''_1 and e''_2 are ordered and so must occur in that order i.e. the computation $x \leftarrow x \times b$ can be initiated only after the termination of the computation $u \leftarrow f(a)$. Furthermore events e'_1 and e'_3 are not ordered meaning that they can proceed concurrently. Two events which are concurrent may occur in any order in time and may even occur coincidentally (simultaneously).

2.3 Limitations of Partial Orderings

In the example of figure 2.1, conditions c_2 and c_4 correspond to the execution of computations $x \leftarrow u \times b$ and $y \leftarrow v \times b$ respectively. If the system has two or more multipliers these two computations can be in progress at the same time without creating any problems. For this situation the partial ordering of figure 2.1 correctly represents the coordination of the events, but if the system has only one multiplier and it can handle correctly no more than one multiplication at a time the partial ordering incorrectly represents the coordination required because it permits simultaneous use of the multiplier by the two computations. In the latter case, for correct operation the two multiplica-

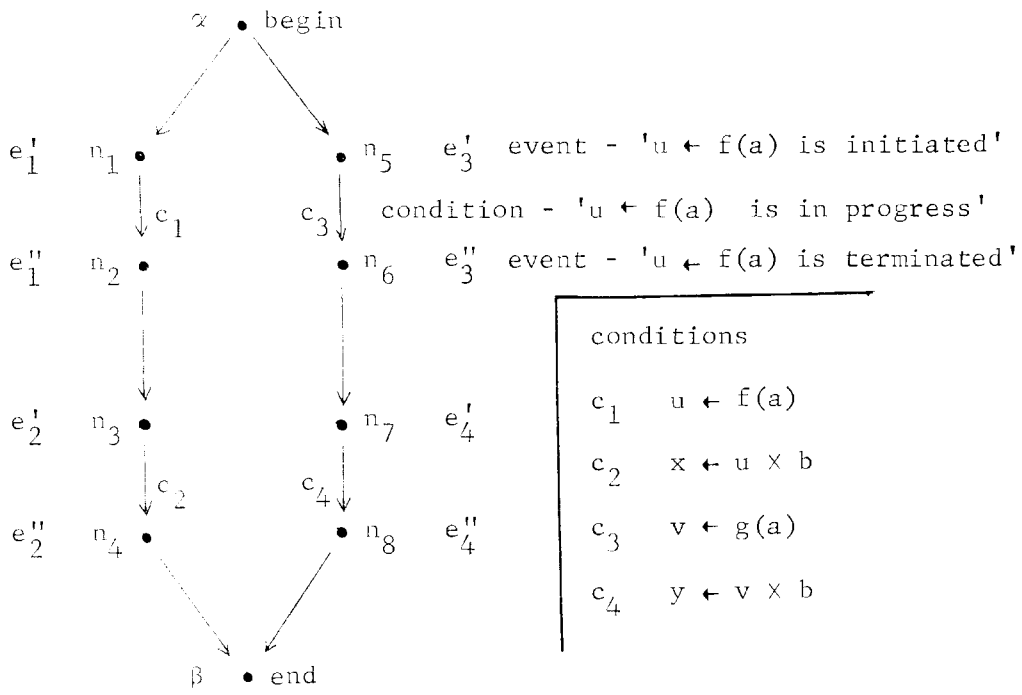


Figure 2.1 Partial Orderings as a Representation Scheme

tions must not take place simultaneously; they may take place in any order as long as they do not overlap (in time). This means that conditions c_2 and c_4 must not hold simultaneously. In using a partial ordering as a representation scheme, the only mechanism for preventing two conditions from holding simultaneously is to sequence them by ordering the event terminating one condition and the event initiating the other condition. Such ordering while preventing simultaneous holding of the conditions, introduces some undesirable side effects of its own because the ordering is fixed a priori. Consider for example the case where the functions f and g are partial functions. There are only two ways in which conditions c_2 and c_4 can be ordered: i) c_4 follows c_2 and ii) c_2 follows c_4. The partial ordering corresponding to the first choice is shown in figure 2.2 . A computation coordinated by this partial ordering does not always lead to the same result as the computation coordinated by the partial ordering of figure 2.1 as for instance when g(a) terminates but f(a) does not. This is the side effect mentioned above. The other choice, being symmetric, has a similar problem. The above example points out the inadequacy of partial orderings in representing conditions or events which may only occur in any order but not simultaneously.

Another limitation of partial orderings is that a situation in which occurrence of an event prevents or precludes the occurrence of some other event cannot be represented. This is illustrated by the following problem relating to the coordination of two cars and a gate through which they must pass.

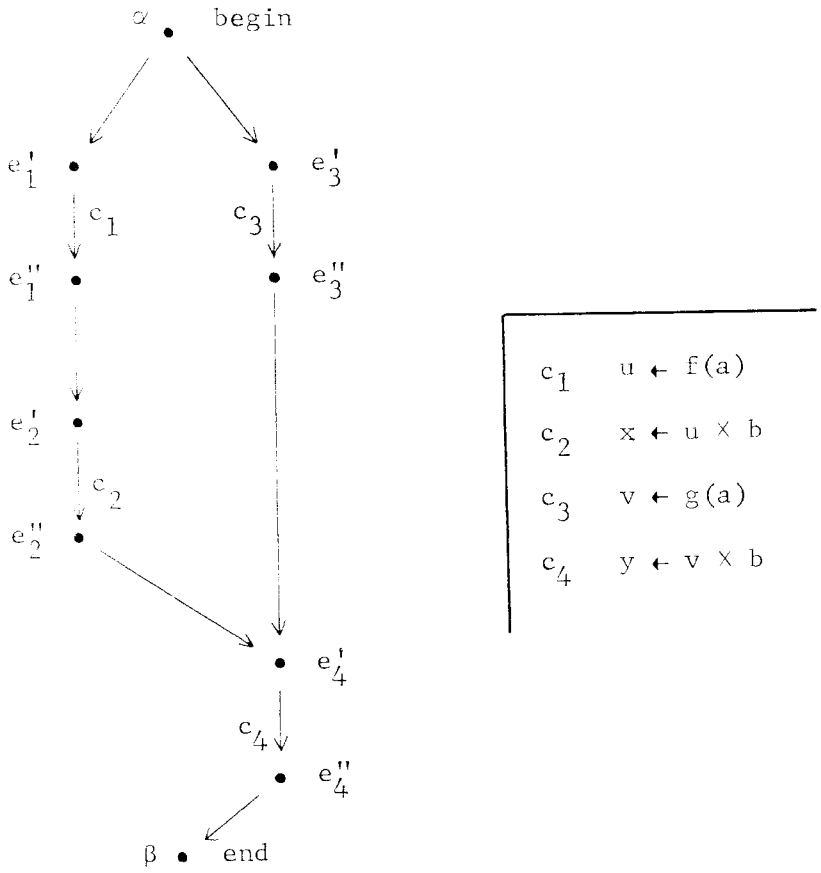
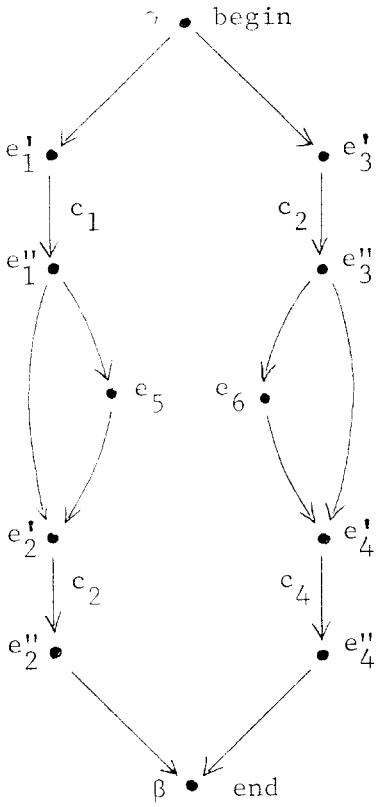


Figure 2.2 Ordering Conditions c_2 and c_4

The Problem of Two Cars and a Gate

Consider two cars and a gate through which they must pass. It is not known which car arrives at the gate first, and the gate is initially closed but can be operated by either car; when the gate is operated, it opens if it is closed and closes if it is open. Furthermore a car can operate the gate only once. The events are to be so coordinated that the gate is opened by the first car, but is undisturbed by the second car so that both cars can pass through the gate as they arrive.

The partial ordering shown in figure 2.3 is an attempt to represent coordination of the events for the desired operation of the gate. Since either car may be the first one to arrive at the gate, both cars should be able to open the gate by operating it. The car that follows the first car should not operate the gate or it will shut the gate that is already open! This implies that the events by means of which the two cars flip the gates must be distinct and unordered. In addition to opening the gate, the first car should do something to help the second car refrain from operating the gate. This means that the occurrence of event e_1'' should preclude the occurrence of event e_6 if it has not occurred already, and similarly the occurrence of event e_3'' should preclude the occurrence of event e_5 if it has not occurred already. Since no events can be removed from the domain of a partial ordering without redefining it, the coordination of events required in the above problem cannot be represented through a partial ordering.



the gate is closed initially

e''_1: car A arrives at the gate

e_5: car A operates the gate

c_2: car A is passing through the gate

e''_3: car B arrives at the gate

e_6: car B operates the gate

c_4: car B is passing through the gate

Figure 2.3 The Problem of Two Cars and the Gate

2.4 Causes of the Limitations

At the root of the inadequacies of partial orderings in representing the coordination of events is the fact that a partial ordering is fixed. A partial ordering is adequate for representing the temporal relationship between events which have occurred already; in such a partial ordering the occurrences which are ordered are said to have occurred in that order, while nothing is known about the temporal relationship among those occurrences which are unordered. If the partial ordering of the occurrences of events is considered a history of events, specification of the coordination of events through a partial ordering completely predetermines the history of the events. These limitations can be removed by using structures like Petri nets [7].

The Petri nets of interest here are the modified Petri nets used by Holt in the "Final Report of the Information System Theory Project" [8]. A brief introduction to Petri nets is given below.

2.5 Petri Nets

Definition A Petri net N is a directed graph defined as a quadruplet, $\langle T,P,A,B^0\rangle$, where

$T = \{t_i,\cdots t_m\}$	is a finite set of transitions	the nodes of the graph
$P = \{p_1,\cdots,p_n\}$	is a finite set of places	
$A = \{a_1,\cdots,a_k\}$	is a finite set of directed arcs of the form $\langle x,y\rangle$ which either connect a transition to a place or a place to a transition.	
$B^0 \subset P$	is the initial stone distribution, the set of places which have stones initially.	

An arc $\langle x,y\rangle$ is said to be directed from x to y. The input places of a transition are the places from which there are directed arcs to the transition. In a similar way the output places of a transition are defined to be the places which have incident arcs from the transition. The sets of input and output places of a transition t are denoted by $I(t)$ and $O(t)$ respectively.

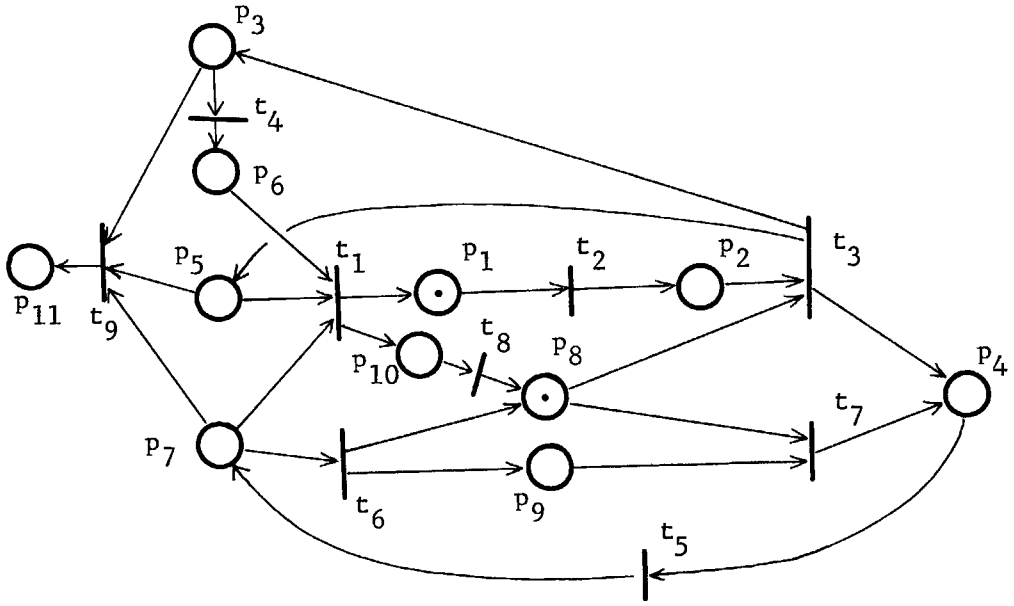
A place is capable of having a stone. A place is said to be full if it has a stone and empty if it does not.

Simulation

A transition is said to be enabled when (all) its input places are full. A transition can occur only when it is in enabled condition. When a transition occurs it removes a stone from each of its input places and puts a stone in each of its output places. Note that the number of stones in the net is not invariant with respect to the occurrence of a transition unless the transition has the same number of output places as input places. When two or more transitions share an input place, only one of the transitions occurs even when more than one is enabled, for the occurrence of any one of them terminates the enabled condition of the others. (Two transitions are said to conflict if they share an input place and can be in enabled condition at the same time.)

The simulation rule states that transitions occur on their own whenever they are enabled, subject to the restriction mentioned above that a transition must pick a stone from each of its input places in order to occur. Implicit in the simulation rule is the fact that transitions which do not share input places proceed independently of one another while transitions that share input places interact with each other.

In the Petri-net presented in figure 2.4, only transition t_2 is in an enabled condition (i.e ready to occur) initially. The occurrence of this transition removes the stone from place p_1 and places a stone in place p_2. In so doing transition t_2 enables transition t_3. When transition t_3 occurs, transitions t_4 and t_5 are enabled.



$N = \langle T, P, A, B^0 \rangle$

where

$T = \{t_1, t_2, \ldots, t_9\}$

$P = \{p_1, p_2, \ldots, p_{11}\}$

$A = \{\langle p_5, t_1 \rangle, \langle p_6, t_1 \rangle, \ \ldots \ , \langle t_9, p_{11} \rangle\}$

$B^0 = \{p_1, p_8\}$

Figure 2.4 A Petri Net

Since these transitions do not have any input places in common, they can proceed independently of each other. If transition t_5 happens to occur before transition t_4, places p_3, p_5 and p_7 get stones and transition t_9 also gets enabled. Transitions t_4 and t_9 are in conflict as both need the stone in place p_3. Therefore either transition t_9 or transition t_4 occurs. If the former transition occurs, the net comes to a rest as no transition is in an enabled condition following this transition. On the other hand if transition t_4 occurs, transitions t_1 and t_6 are enabled. These transitions are also in conflict, and therefore only one of them occurs. If transition t_1 occurs, places p_1 and p_{10} get stones. At this point transition t_8 and t_2 are enabled. If t_8 occurs, the net reaches the condition it started from and the action continues as before.

It should be observed that in the above net no place can have more than one stone in it at any time. A net with this property is called a safe net.

Figure 2.5 shows a Petri-net representing the coordination involved in the problem discussed earlier with figure 2.1 which could not be represented as a partial ordering of events. The important thing to note in figure 2.5 is that conditions c_2 and c_4 are concurrent but do not hold at the same time i.e. they are not coincident.

Figure 2.6 shows a Petri net representing the coordination of the cars in the problem discussed earlier with figure 2.3 .

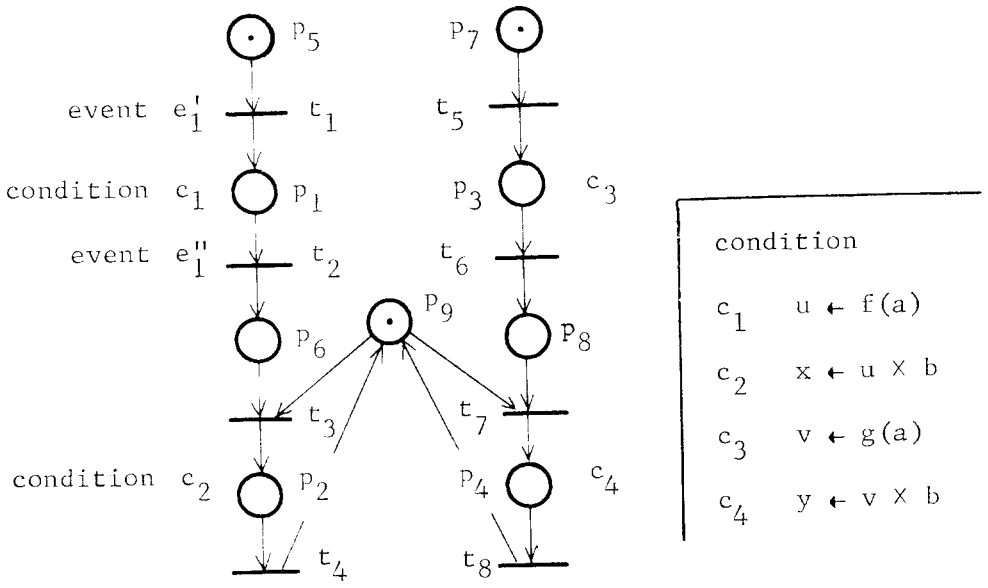
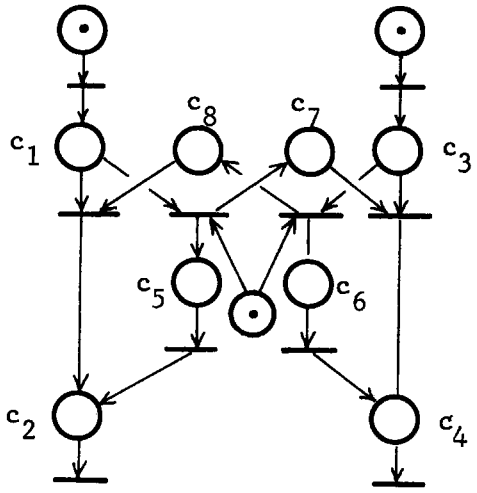


Figure 2.5 A Petri Net Representing the Coordination for the case of one Multiplyer



The gate is closed initially

c_1: car A comes to the gate

c_2: car A passes through the gate

c_3: car B comes to the gate

c_4: car B passes through the gate

c_5: car A may proceed as it has flipped the gate

c_6: car B may proceed as it has flipped the gate

c_7: car B may proceed as car A has flipped the gate

c_8: car A may proceed as car B has flipped the gate

Figure 2.6 A Petri net for the Coordination of the two Cars Passing through the Gate

The author has found Petri-nets to be adequate in representing coordinations of events, but it appears that a claim that Petri-nets provide a satisfactory formal counterpart to vague notions about coordination of asynchrounous events cannot be proved just as the claim that Turing Machines provide a satisfactory formal counterpart to the vague concept of algorithm cannot be proved. The claim must be accepted or rejected on the basis of experience, and the experience of the author and that of others [12] indicates that Petri-nets provide a satisfactory formalism for the study of coordination of asynchronous events.

Even though Petri nets are satisfactory as regards the varities of coordination they can represent, they tend to be too detailed, and complexity of representation of problems increases with the size of the problems at an undesirable rate. Any improvement that leads to a reduction in details and simplification of representation is valuable, especially because the representation scheme is being developed for specifying and formalizing coordination relating to practical problems. The representation scheme called a coordination net, which is described next, incorporates the considerations mentioned above. Coordination nets do not add more variety to the class of coordinations represented by Petri nets, but they make representation simple and manageable. Moreover, the simplicity of representation is carried over to the implementation of the coordination.

2.6 Coordination Nets

Coordination nets are generalization of Petri-nets. The transitions in coordination nets are classified into three classes viz, input-transitions, output-transitions and internal transitions, depending on the role they play in the interaction by means of which the net coordinates the events in the external-world, i.e. the world external to the net. For correct representation of such interaction, the process of the occurrence of a transition in a coordination net has been split into two steps: in the first step the transition initiates by claiming a stone from each of its input places, and in the second step the transition terminates by removing those stones from the input places and putting stones in the output places. The interaction between the net and the external world takes place as described in the next paragraph.

An input transition does not initiate until the external world reaches a certain condition, called an input-condition, which is associated with the input transition. On the other hand the initiation of an output transition indicates to the external world that it should proceed with a certain event associated with that transition. An output transition terminates only after the associated event has occurred. The internal transitions do not take part in such interaction; they can be thought of as performing some internal computation.

The presence of a stone at a place corresponds to the holding of a condition. In a similar way, the presence of stones at a number of places also constitutes effectuation of a condition. The power set of P, the set of places in the net, represents the set of all possible conditions. Petri nets as well as coordination nets specify coordination by specifying certain restrictions on the holding of conditions. In particular they rule out certain conditions. In case of a Petri net the constraints are specified by means of the structure of the net alone, but in case of a coordination net some constraints are specified by means of the structure and some by means of a constraint set which lists the conditions which are to be ruled out. A constraint set is a subset of the power set of P, and its members are called constraints.

For example in the problem of the cars and the gate (figure 2.6) the additional constraints that only one car should arrive at the gate at a time and that only one car should go through the gate at a time can be incorporated by specifying the constraint set $\{\{c_1, c_3\}, \{c_2, c_4\}\}$ which means that conditions c_1 and c_3 should not hold at the same time and that conditions c_2 and c_4 should not hold at the same time.

The conditions which are listed in the constraint set are ruled out directly, and in addition those conditions which include these conditions are also ruled out (recall that a condition is a subset of P). For example when $P = \{a,b,c,d\}$, ruling out condition $\{a,d\}$ rules out conditions $\{a,b,d\}$, $\{a,c,d\}$ and $\{a,b,c,d\}$ as well, for when any of

these conditions hold, the condition {a,d} also holds.

A coordination net C is a tuple $< N, Ct >$, where

$N \triangleq < T, P, A, B^0 >$ is a Petri-net

where

$T = T^i \cup T^o \cup T^n$ is a finite set of transitions

where

T^i is a set of transitions designated as input transitions

T^o is a set of transition designated as output transitions (sets T^i and T^o are not necessarily disjoint)

T^n is a set of transitions which are neither input transitions nor output transitions i.e. they are internal transitions.

$P = \{p_1, \ldots, p_n\}$ a set of finite number of places

$A = \{a_1, \ldots, a_k\}$ a set of finite number of directed arcs of the form $< x,y >$ which connect places to transitions and transitions to places.

$B^0 \subset P$ is the set of places which have stones initially (this is referred to as the initial stone distribution).

$Ct \subset \mathcal{P}(P)$ is a constraint set which contains the constraints i.e. the sets of places which should not have stones at the same time. For example the constraint set $\{\{p_1, p_4\}, \{p_1, p_2, p_3\}\}$ specifies that places p_1 and p_4 should not have stones at the same time and places p_1, p_2 and p_3 should not have stones at the same time. Input and output places of a transition are not permitted to be part of the same constraint.

A transition is said to be enabled if its input places have unclaimed stones. An enabled input transition is said to be ready to initiate if the external world has reached the input-condition associated with the transition. The other types of transitions, namely internal transitions and output transitions, are ready to initiate when they are enabled. In order to initiate, a transition must be ready and must claim a stone at each of its input places. Thus when two or more transitions have a common input place, only one of them initiates even if all are ready as initiation of any one of them terminates the ready condition of the others. Internal transitions and the input transitions terminate immediately following their initiation, but an output transition terminates only when the event associated with that transition has occurred. A transition is said to be active if it has initiated but not terminated. In the process of initiation, a transition merely claims the stones in input places: At termination the transition removes a stone from each of the input places and puts a stone in each of the output places.

In addition to satisfying the requirement for initiation of a transition, viz that the transition be ready and that it claim a stone at each input place, the initiation of the transition must not violate the constraints specified by the constraint set.

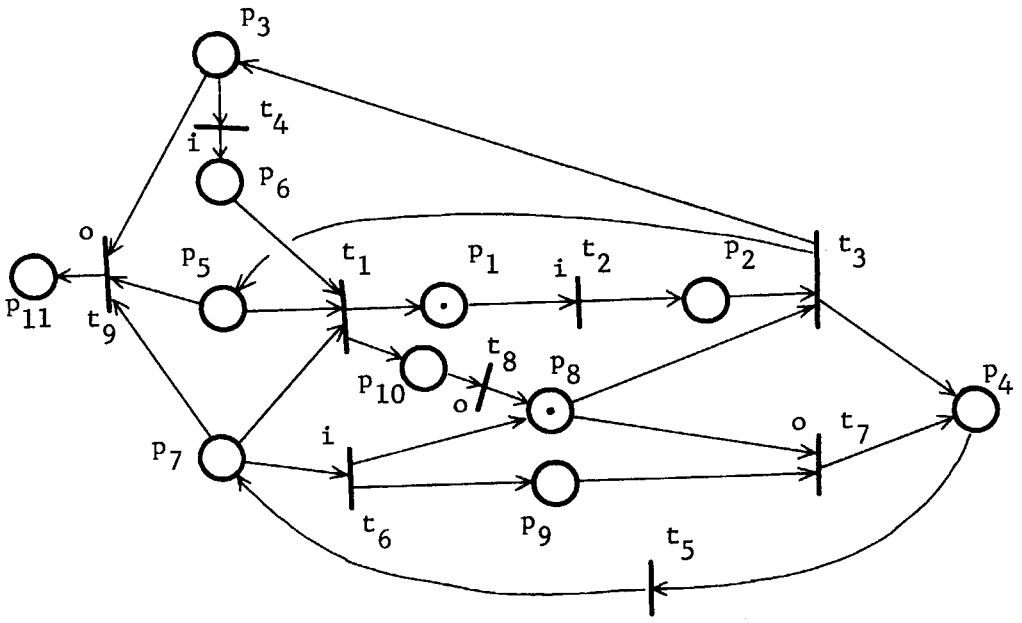
Places which have stones and places which are output places of active transitions are said to be <u>active places</u>. A set of active places is said to be <u>admissible</u> if the set does not contain any collection of places which is a member of a constraint set, e.g. if $\{\{p_1,p_4\}, \{p_1,p_2,p_3\}\}$ is the constraint set then the set of active places $\{p_2,p_3,p_4\}$ is admissible but $\{p_1,p_3,p_4\}$ is not.

The simulation of a coordination net proceeds through initiation and termination of transitions in such a manner that at all times the set of active places in the net is admissible. Thus a ready transition can be initiated only if the set of active places it leads to is admissible, i.e. the output places of the transition can be added to the set of active places without making the set of active places inadmissible. A transition can terminate without regard to the constraints because it only reduces the number of active places.

The simulation rule given in the paragraph above implies that transitions which neither have input places in common nor have output places that belong to a common constraint proceed independently. Transitions for which this does not hold interact with each other i.e. they are not independent of each other.

An Example Figure 2.7 shows a coordination net. The Petri-net part of this coordination net was discussed earlier in the section on Petri-nets. The constraint set of this net consists of only one constraint, $\{p_3,p_5,p_7\}$, which specifies that these places should not be active at the same time, and therefore they cannot have stones at the same time.

Only the input transition t_2 is enabled initially. Being an input transition, the transition becomes ready (to initiate) only when the associated input-condition is attained in the external world. When this happens, the transition initiates and then terminates by removing the stone at place p_1 and putting a stone at place p_2. The occurrence of transition t_1 is thus complete. Transition t_3 is now ready. Since this transition is an internal transition, it occurs without regard to the condition of the external world. The occurrence of this transition removes stones from places p_2 and p_8 and puts stones in place p_3, p_4 and p_5. At this point transition t_5 is ready to occur, but it cannot initiate as that would make places p_3, p_5 and p_7 active - a condition ruled out by the constraint $\{p_3,p_5,p_7\}$. Thus the occurrence of transition t_5 is delayed until transition t_4 has occurred. When these transitions have occurred, places p_6, p_5 and p_7 have stones. At this point if the external world has reached the input-condition associated with the input transition t_5, transition t_1 and t_6 are in conflict, and even though both of them are ready, only one of them occurs. In the following discussion only the case in which transition t_1 occurs is explained for the other case is not much different from this one.



$C = \langle N, Ct \rangle$

where

$N = \langle T, P, A, B^0 \rangle$

where

$T = \{t_1, t_2, \ldots, t_9\}$

$P = \{p_1, p_2, \ldots, p_{11}\}$

$A = \{\langle p_5, t_1 \rangle, \langle p_6, t_1 \rangle, \ \ldots \ , \langle t_9, p_{11} \rangle\}$

$B^0 = \{p_1, p_8\}$

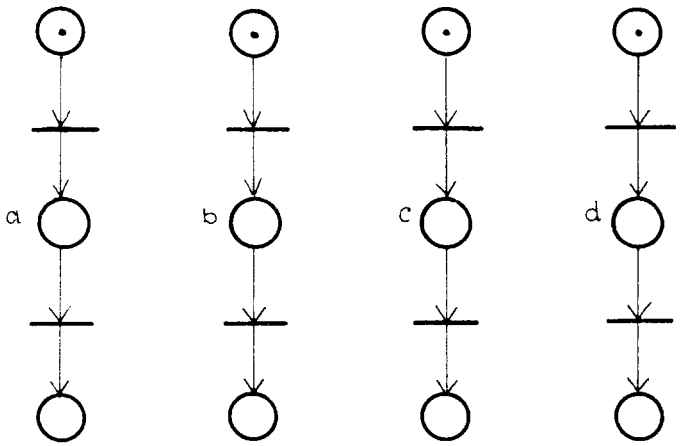
$Ct = \{\{p_3, p_5, p_7\}\}$

Figure 2.7 A Coordination Net

If transition t_1 occurs, places p_1 and p_{10} get stones and the output transition t_8 is initiated. The initiation of this transition informs the external world that it should proceed with the event associated with the transition. When this event has occurred, the transition terminates and place p_8 gets a stone, and the net reaches the condition it started out with. The action thus continues.

It should be observed that transition t_9 can never get a chance to occur because the constraint $\{p_3,p_5,p_7\}$ rules out the possibility of p_3, p_5 and p_7 being full at the same time.

An illustrative example showing case of representation offered by coordination nets over Petri nets is presented in figure 2.8 and 2.9 both of which represent the same coordination.



$$Ct = \{\{a,d\},\{a,b,c\}\}$$

Figure 2.8 A Coordination Net

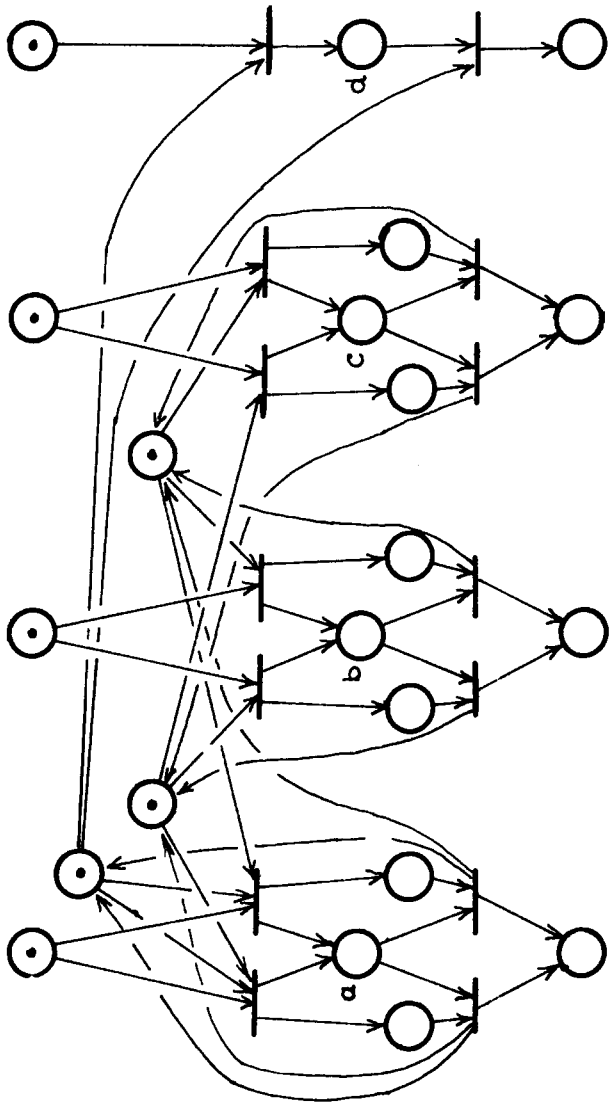


Figure 2.9 A Petri net Representing the same Coordination as the Coordination net of figure 2.8

CHAPTER 3

HOMOGENEOUS COORDINATION NETS

3.1 Introduction

The choice of a representation scheme on the basis of ease of representation of problems is justified when the representation of a problem in the chosen scheme can be transformend into a representation which is easy to implement. For example, high level languages are used in programming because a programmer can specify the algorithm in his mind more easily in a high level language than in a machine language; the algorithm specified in the high level language can be transformed into an equivalent algorithm in the machine language through the mechanical process of compilation.

The choice of coordination nets as the representation scheme was influenced by these considerations. Coordination nets of a particular form called homogeneous coordination nets are particularly suited for implementation of coordinations. Coordination nets used in formalizing problems may or may not be homogeneous, but fortunately the non-homogeneous coordination nets can be mechanically transformed into equivalent homogeneous coordination nets.

3.2 Homogeneous Coordination Nets

Homogeneous coordination nets are coordination nets in which the output places of transitions have a certain equivalence with respect to the constraint set.

It should be recalled that a constraint is a set of places, and if x is a constraint and $x \subset y \subseteq p$ then x implies constraint y. This allows the reduced constraint set to be defined as the smallest subset of the constraint set that implies the entire constraint set. The reduced constraint set for a constraint set Ct is denoted by $R(Ct)$. The domain of a constraint set is said to be the set of places which participate in the constraints.

The domain of influence of a place is the set that is obtained by deleting the place from the set of constraints (in the reduced constraint set) that involve that place. The domain of influence of a place p_i is denoted by $DI(p_i)$

Thus when $R(Ct) = \{\{p_1, p_4\}, \{p_1, p_2\}\}$ then $DI(p_1) = \{\{p_4\}, \{p_2\}\}$.

and $DI(p_4) = \{\{p_1\}\}$

Two places are said to be constraint equivalent if they have the same domain of influence. This equivalence relation is denoted by CE. A transition is said to be a homogeneous transition if its output places are constraint equivalent, and a coordination net is said to be a homogeneous coordination net if its transitions are homogeneous.

In a homogeneous coordination net, as the output places of transitions are constraint equivalent, the constraints associated with the output places can be associated with the transition themselves. The way this fact is used in the implementation of coordination nets presented in the next chapter is that the constraints are brought into force at initiation of the transition and are kept in force until all of the stones put into the output places by the transition are used up by some other transitions, that is, the constraints come into force the moment the output places become active and remain in force until all output stones put into the output places by the transition are removed by other transitions.

The constraint equivalence relation partitions the set P of places into equivalence classes $P_1,\ldots,P_n$. The constraint set that is obtained by replacing the places in the reduced constraint set by corresponding equivalence classes is called the reduced constraint set with respect to equivalence classes of places and is denoted as RP(Ct).

Thus if $P = \{p_1,\ldots,p_8\}$

and

$$Ct = \{\{p_1,p_5\}, \{p_1,p_2,p_3\}, \{p_5,p_4\}, \{p_5,p_2,p_3\}$$

$$\{p_1,p_6,p_3\}, \{p_5,p_6,p_3\}\}$$

Then

$$P_0 = \{p_7,p_8\} \quad P_1 = \{p_1,p_5\} \quad P_2 = \{p_2,p_6\} \quad P_3 = \{p_3\} \quad P_4 = \{p_4\}$$

and $RP(Ct) = \{\{P_1,P_4\},\{P_1,P_2,P_3\}\}$

The constraint set conditional to places $p_1,\ldots,p_k$ refers to the constraint set obtained by removing places $p_1,\ldots,p_k$ from the constraints. The conditional constraint set gives the constraints given that certain places, in this case $p_1,\ldots,p_k$, have stones, and is denoted by $Ct/\{p_1,\ldots,p_k\}$. Thus in the above example $Ct/\{p_1,p_3\} = \{\{p_4\}, \{p_2\}, \{p_5,p_4\}, \{p_5,p_2\}, \{p_6\}, \{p_5,p_6\}\}$

Similarly $RP(Ct)/\{p_1,p_3\} = \{\{p_4\},\{p_2\}\}$

3.3 Transformation into Homogeneous Nets

A non-homogeneous coordination net can be transformed into a homogeneous coordination net by applying a straightforward transformation to each non-homogeneous transition in the net. Basically the transformation breaks up a non-homogeneous transition into a number of homogeneous transitions (figure 3.1).

What the transformation does is to introduce intermediate transitions, one for each equivalence class of places in the set of output places, between the transition under consideration and its output places; the intermediate transitions now do the task of putting stones in the output places of the original transition.

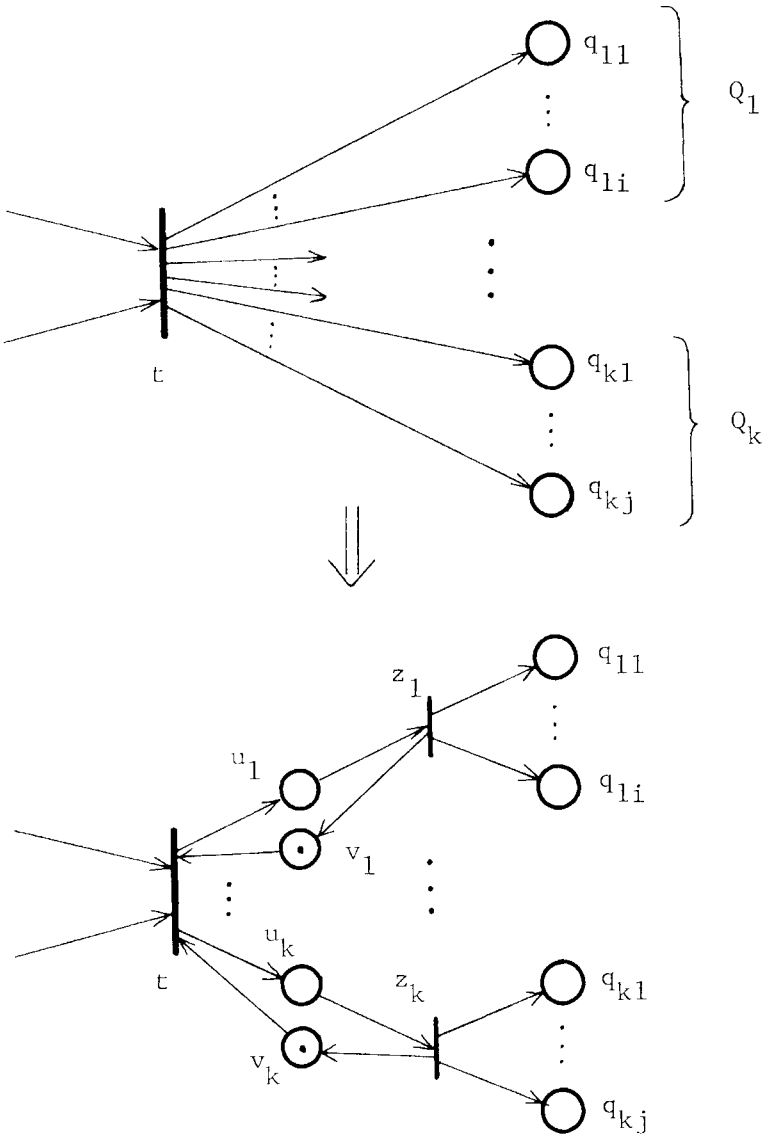


Figure 3.1 Transformation of a Non-homogeneous Transition into Homogeneous Transitions

Consider a non-homogeneous transition t that is to be transformed into a set of homogeneous transitions. As the transition is non-homogeneous, the output places of the transition fall into some constraint equivalence classes $Q_1,\ldots,Q_k$. The transformation introduces a transition Z_i and two places u_i and v_i for each equivalence class Q_i (see figure 3.1). In the transformed net, the places in Q_i, instead of directly being output places of transition t, are output places of an intermediate transition z_i which is connected to the transition t by intermediate places u_i and v_i ; u_i is an output place of t and an input place of z_i, v_i is an output place of z_i and an input place of t, and places v_i's have initial stones. The new constraint set is obtained by adding to the existing constraint set, the collection of constraints that is obtained by selecting the constraints that involve the output places of the original transition and substituting in them u_i for those places. In other words the new constraint is such that the conditional constraint set with respect to u_i is the same as the condidional constraint set with respect to the set of output places of the original transition.

The collection of transition that results from the transformation behaves as the original transition because on termination of the given transition, the intermediate transitions can occur unobstructed as the constraints associated with u_i's cover the constraints associated with the output places of the original transition.

3.4 Conflicts and Conflict Clusters

Two transitions in a coordination net are said to conflict if in some simulation of the net they may be enabled together but the occurrence of either one of them terminates the enabled condition of the other (because they have a common input place). Alternatively two transitions t_i and t_j conflict if $I(t_i) \cap I(t_j) \neq \Phi$ and there is some simulation for which they are in enabled condition at the same time. This relation between transitions is denoted by $C_f(t_i,t_j)$. The relation C_f is symmetric but not necessarily transitive.

A conflict should not be regarded as an error or as an unwanted situation, for conflicts provide a means for introducing a controlled amount of uncertainty into coordination nets. In the problem of cars and gate discussed in the preceding chapter, conflicts play an important role in coordinating the cars.

Conflicting transitions cannot proceed independently; their activities must be coordinated as occurrence of either one must prevent occurrence of the other. The conflict relation C_f for a net can be obtained by systematically simulating the net. Such a simulation, even though finite, may be long. It is not essential that the relation C_f be known completely for implementing a coordination net, for a relation called pseudo conflict relation, which can be easily obtained from

the net, can be used in coordinating the transitions in place of the conflict relation.

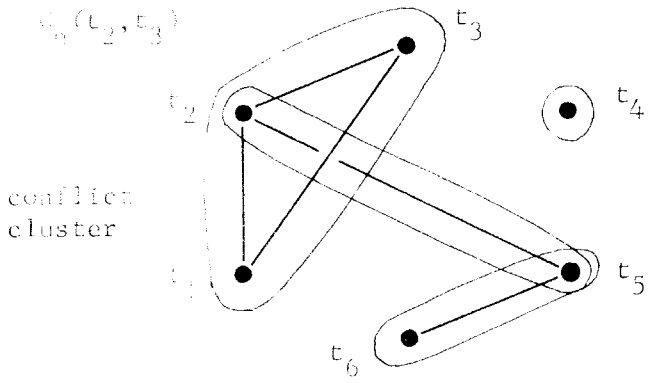
Pseudo Conflicts

Two transitions are said to be in pseudo conflict if they have an input place in common. Pseudo conflict is represented by a binary relation S where $S(t_i,t_j) \triangleq (I(t_i) \cap I(t_i) \neq \Phi)$.

For two transitions to be in conflict, they must have at least one input place in common, $C_f(t_i,t_j) \Rightarrow S(t_i,t_j)$. Moreover, if a binary relation $P(t_i,t_j)$ is defined to hold if t_i and t_j can occur coincidentally (simultaneously) in some simulation of the net, then $S(t_i,t_j) \Rightarrow \neg P(t_i,t_j)$. Therefore the following relation holds.

$$\forall\, t_i,t_j \in T \quad (C_f(t_i,t_j) \Rightarrow S(t_i,t_j)) \wedge (P(t_i,t_j) \Rightarrow \neg S(t_i,t_j))$$

This relation among C_f, S and P indicates that any relation C_n which is in between C_f and S (including them) can be used to coordinate the transitions, that is it will prevent conflicting transitions from occurring coincidentally but will not disturb the transitions which should be allowed to occur coincidentally.



$\{tc = \{\{t_1, t_2, t_3\}, \{t_2, t_5\}, \{t_4\}, \{t_5, t_6\}\}$

Figure 3.2 Conflict Clusters

Conflict Clusters

A conflict cluster is a maximal set of transitions such that the transitions in the set are pairwise related by C_n, i.e., a transition in the set conflicts with every other transition in the set. The set of (all) conflict clusters of a coordination net is denoted by C_{fc}. Figure 3.2 presents the concept of conflict cluster graphically. In this figure, an edge between two transitions indicates that they conflict (according to the relation C_n). The conflict clusters are indicated by drawing enclosures around the transitions in the clusters. The domain of a set of conflict clusters is said to be the set of transitions that participate in the conflict clusters, and is denoted by $D(C_{fc})$.

CHAPTER 4

COORDINATION STRUCTURES

4.1 Introduction

This chapter presents asynchronous modular structures, called coordination structures, for implementing safe homogeneous coordination nets (for meaning of safe nets see page 25). Coordination structures can be derived systematically from the nets using the method presented in this chapter. It is assumed that the nets do not have any places which are both input and output places of the same transitions. This is not an undue restriction as nets can be made free of such places by introducing intermediate transitions (and places) between such places and the transitions; those places then become output places of the intermediate transitions instead of the original transitions.

The modules used in the construction of coordination structures are specified in this chapter. The thesis, however, does not give hardware implementation of these modules, but the author believes that a neet systematic implementation of these modules can be worked out. A brief discussion on hardware implementation of these modules is presented in the appendix.

4.2 Asynchronous Modular Structures

Asynchronous modular structures [13] are constructed from a small number of types of asynchronous modules which are interconnected by links at certain well defined points (of the modules) called ports. The modules are connected to each other only through their ports, and at most one link may be connected to a port.

Figure 4.1 presents an asynchronous modular structure. This structure permits events a and b to occur concurrently but prevents event c from occurring until both events a and b have occurred. The structure consists of four asynchronously operating modules connected by links. The links carry signals in both directions (figure 4.2), a ready signal in the forward direction and an acknowledge signal in the reverse direction. The signalling discipline requires that the first signal be a ready signal and that the ready signals be separated by acknowledge signals. A ready signal indicates that the module which receives it may begin with its activity. A ready signal is later acknowledged by returning an acknowledge signal. The acknowledge signal need not be returned immediately following the receipt of the ready signal. In that case the acknowledge signal conveys completion of some action in addition to acknowledging receipt of the ready signal. The link described above is called a simple link. The operation of the modules used in the structure shown in figure 4.1 is explained below.

The wye module sends a ready signal on both links 1 and 2 upon receiving a ready signal on link 0, and returns an acknowledge signal on link 0 upon receiving the corresponding acknowledge signals on both

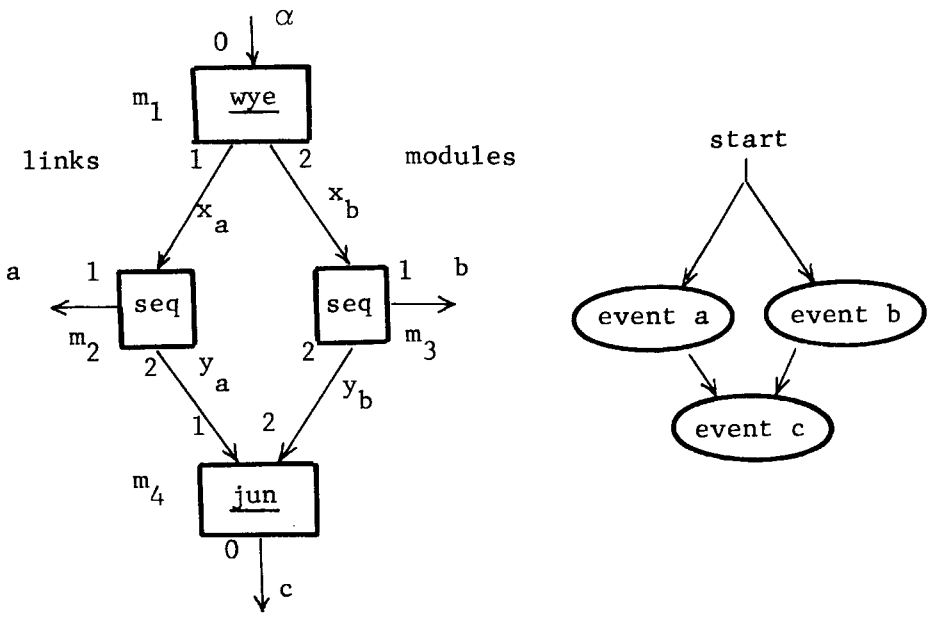


Figure 4.1 An Asynchronous Modular Structure

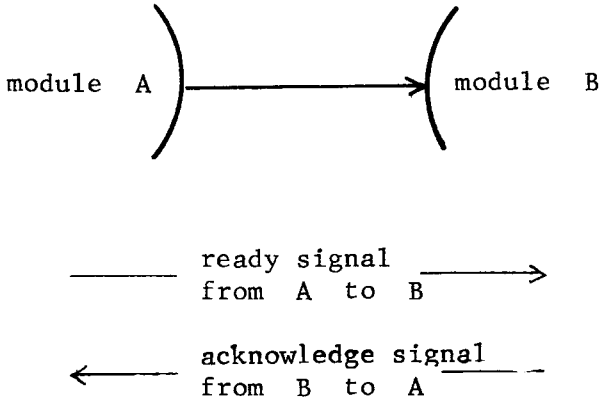


Figure 4.2 Signals on Links

links 1 and 2 . The seq (sequence) module sends a ready signal on link 1 upon receipt of a ready signal on link 0 and, later, sends a ready signal on link 2 following the receipt of an acknowledge signal on link 1 in response to the ready signal which was sent out on that link. When an acknowledge signal is received on link 2, the module returns an acknowledge signal on link 0. It thus serves to order the activities of the two group of structures connected to the ports 1 and 2, whence the name "sequence". A jun (junction) module sends a ready signal on link 0 when it receives ready signals on both links 1 and 2, and when this signal is acknowledged, the module returns acknowledge signals on links 1 and 2 .

The operation of the structure is as follows. Upon receiving a ready signal on link α , the structure enables events a and b by sending ready signals to them. When both of these events have occurred as indicated by the acknowledge signals on links a and b , the structure enables event c . Following the occurrence of event c an acknowledge signal is returned on link α signalling the completion of the occurrences of all events associated with the structure.

4.3 Implementation of Coordination

Coordination of events can be carried out only if the occurrences of events can be controlled, because the absence of control precludes any enforcement of coordination. In order to force coordination on occurrences of events, it should be possible for the coordination structure to delay the occurrence of the events for as long as is deemed necessary. Such a control can be had if the occurrence of an event depends upon receipt of a signal from the coordination structure. The signal from the coordination structure can be thought of as permission for the event to occur. Moreover when the occurrence is completed, the event should send a signal to the coordination structure so that the structure can up-date its knowledge of the status of events.

In a coordination net, the transitions referred to as input transitions are a means through which the external world, i.e., the world in which the events occur, indicates to the net that certain conditions have been reached in the external world, say a processor has become available for use. In coordination structures, the external world sends such information by sending a ready signal to the structure on a link assocated with an input transition. The coordination structure records this information and returns an acknowledgment (i.e. an acknowledge signal). The coordination structure instructs the external world to proceed with an event by sending a ready signal on the appropriate link, and when the occurrence of the event is completed, the external world sends an acknowledge signal to the structure on the link. The links are thus the means of communication between the coordination

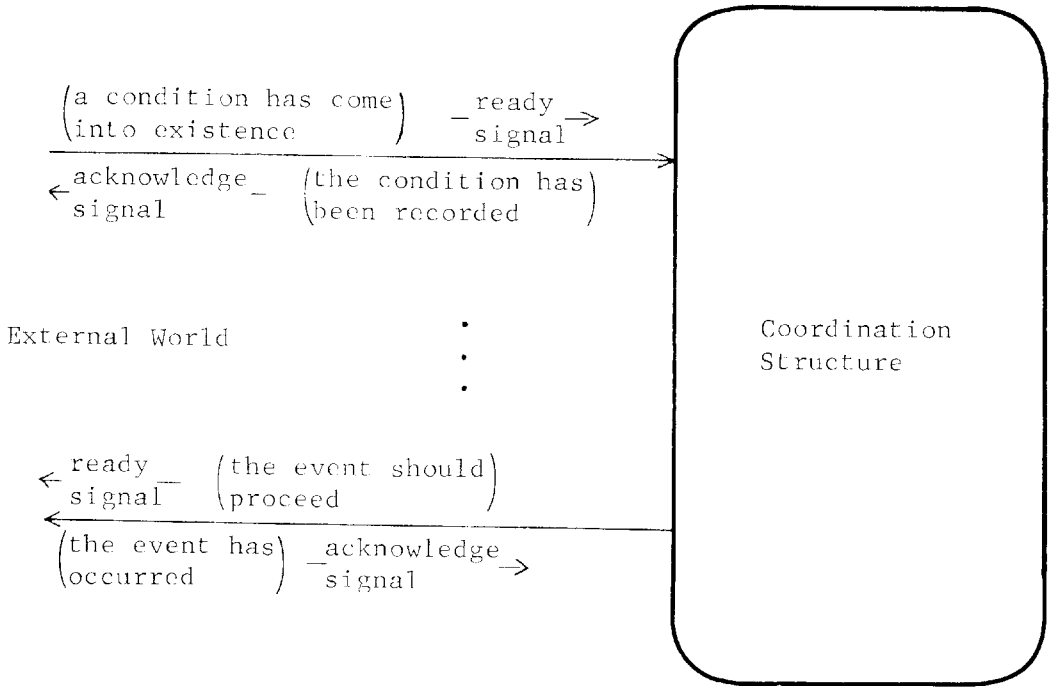


Figure 4.3 Communication between the External World and the Coordination Structure

structure and the external world; the links incident on the structure bring information regarding the condition of the external world while those emergent from the structure relay permission for the external world to proceed with events (figure 4.3).

Coordination structures are partitioned functionally into four parts: i) the precedence structure, ii) the constraint structure, iii) the conflict structure and iv) the initialization structure (figure 4.4). The precedence structure establishes precedence among events, that is it ensures that events do not occur until the events which should precede them have occurred. The constraint structure ensures that the constraints are not violated, the conflict structure resolves conflicts among transitions if and when they arise, and the initialization structure establishes the initial condition in the precedence structure as well as the constraint structure to correspond to the initial distribution of stones in the net.

The sub-structures listed above are derived in terms of asynchronous modules which are connected with links. The links used in such interconnection are described below.

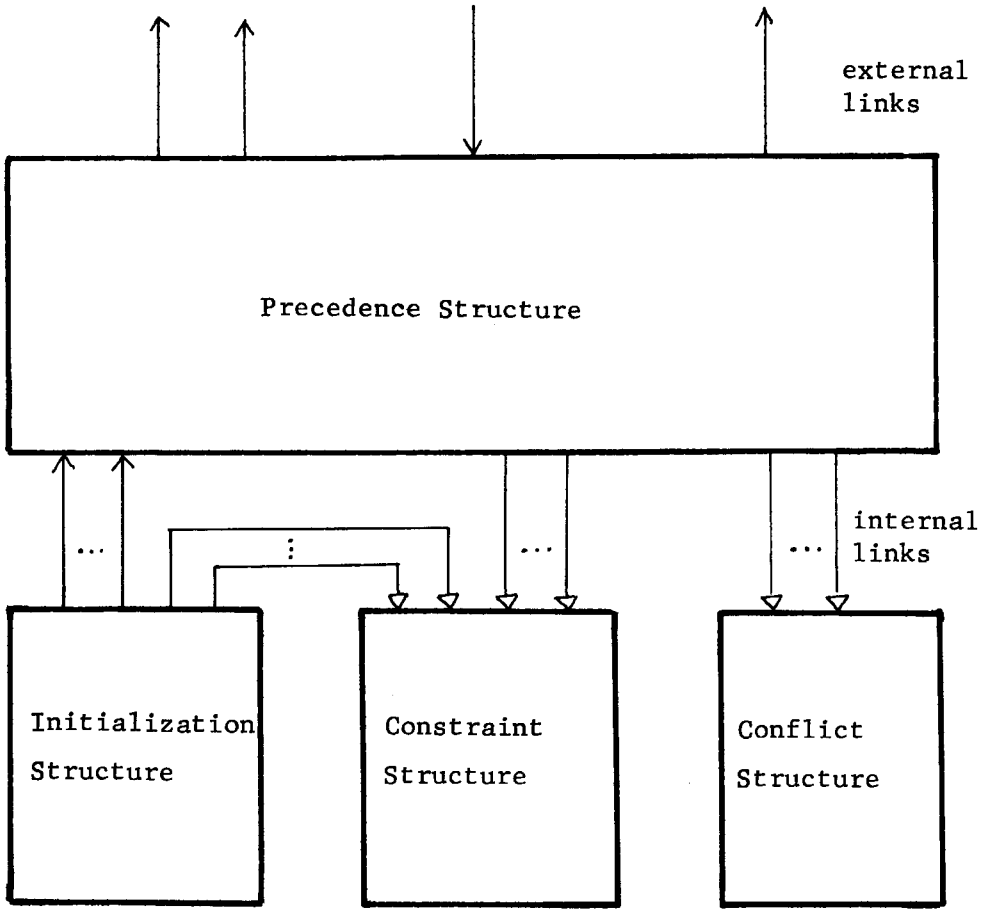


Figure 4.4 Coordination Structures

4.4 Links

In coordination structures four types of links are used:

i) single links, ii) double links, iii) triple links, and iv) mixed-triple links. These links are called compound links as they are themselves composed of two basic types of links called simple links and decision links. The single, double and triple links consist of one, two and three simple links respectively, and the mixed-triple link consists of two decision links and one simple link (figures 4.5 and 4.6). The basic links within the compound links are called sublinks, and they operate independently of each other; the only inter-dependence of signals on them is that established by the modules they connect.

The operation of a simple link was explained earlier in section 4.2 . The decision link is similar to a simple link, but it has two types of acknowledge signals, a positive acknowledge signal and a negative acknowledge signal to convey a decision, whence the name "decision link".

4.5 Signalling Discipline and Communication Cycles

The first signal on a basic link must be a ready signal, and the ready signals must be separated by acknowledge signals. A communication cycle on a basic link consists of a ready signal followed by an acknowledge signal. Communication over a compound link takes place in units of cycles where a cycle consists of certain cycles on the sublinks. Thus a communication cycle on a double link consists of a cycle on each of its sublinks, that on a triple link consists of a cycle on sublink e

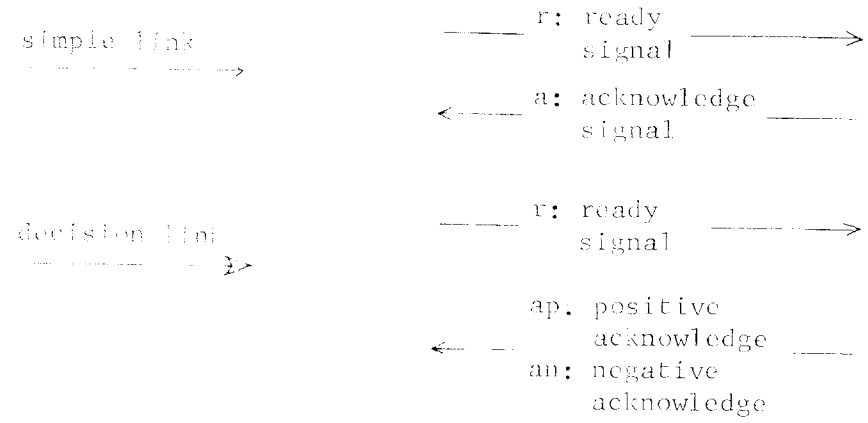


Figure 4.1 The Basic Links

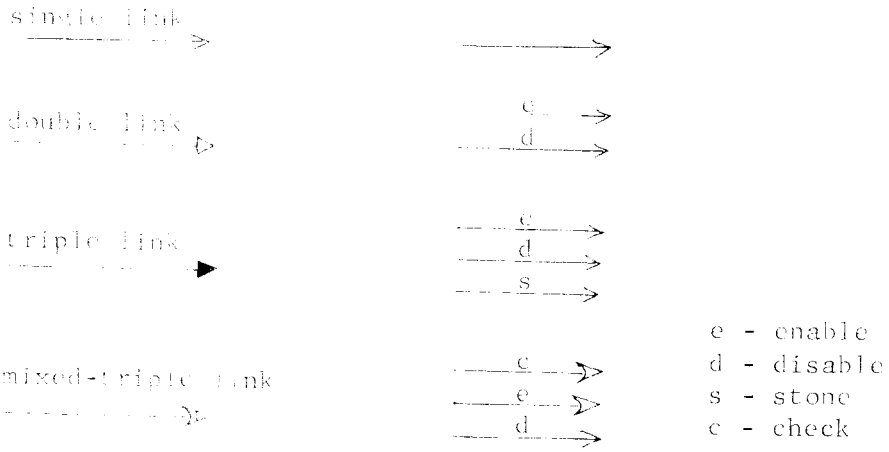


Figure 4.2 The Compound Links

and a cycle on either sublink d or sublink s , and that on a mixed-triple link consists of a cycle on either of sublinks c or e and a cycle on sublink d . The cycles on sublinks that belong to a communication cycle on a compound link are said to correspond to each other. The only inter-relationship among the cycles on sublinks is that established by the modules they connect.

4.6 Overview of the Structure and Operation of Coordination Structures

It was stated earlier that coordination structures consist of four parts (figure 4.4): i) the precedence structure, ii) the conflict structure, iii) the constraint structure, and iv) the initialization structure. The precedence structures are systematically derived from coordination nets by substitution of modular structures for places and transitions in the net (figure 4.7).

The structure substituted for a place consists of a P-module with modules to accomodate multiple fan-in and fan-out of arcs, and the structure substituted for a transition consists of a T-module again with modules to accomodate fan-in and fan-out . The links connecting these structures correspond to the arcs joining places and transitions in the coordination nets. The fan-in and fan-out modules used in conjuction with a P-module are called IP and EP modules, and the fan-in and fan-out modules used in conjuction with the T-module are called IT and ET modules respectively.

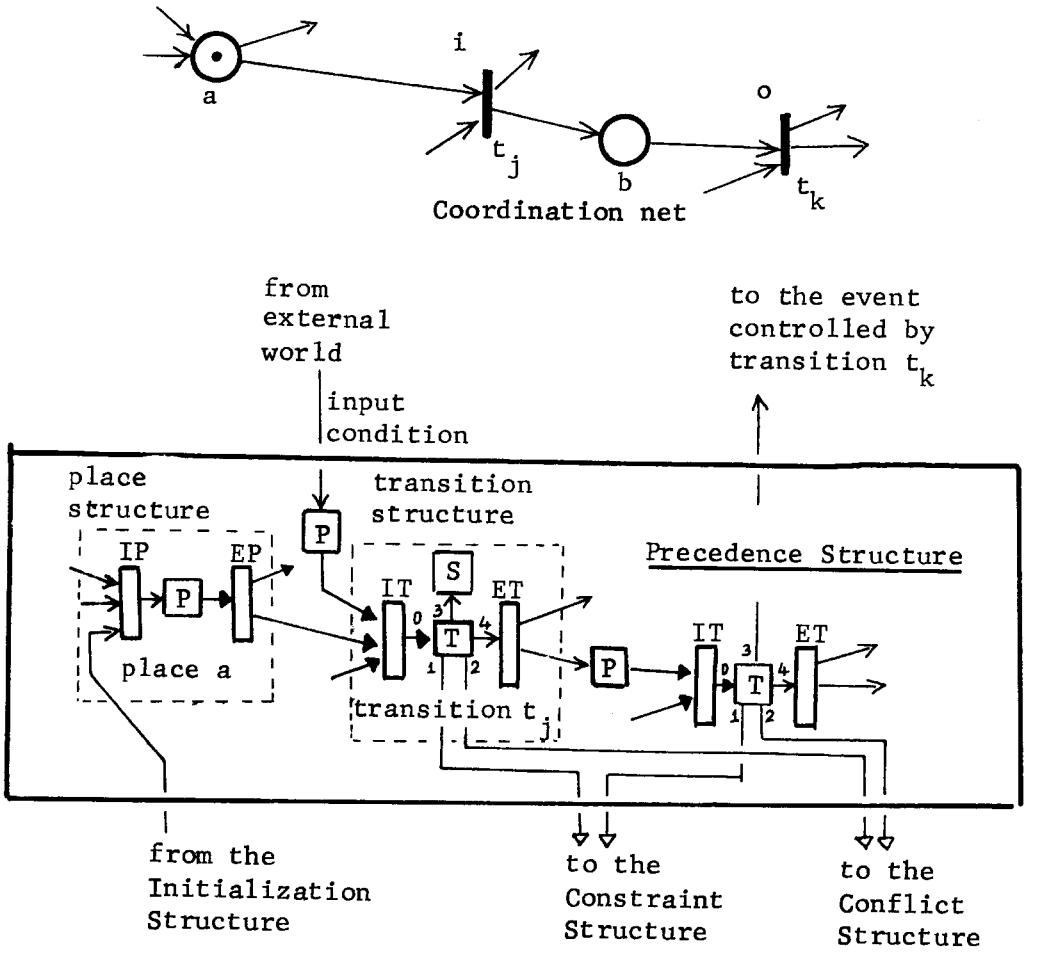


Figure 4.7 The Precedence Structure

The links at ports 1 and 2 of a T-module go to the constraint and conflict structures respectively (figure 4.7). The link at port 3 of a T-module goes to the external world if the transition is an output transition, otherwise the link is terminated by an S module (sink module). The IT-module associate with an input transition has an incident link from the external world. The incident link from the external world reaches the IT-module through a P-module which changes it to a triple link from a single link.

Sending a ready signal to the P-module in a place structure corresponds to placing a stone at a place in the coordination net. The link emerging from the P-module is a triple link. A triple link consists of three sublinks called e, d and s which stand for "enable, "disable" and "stone" respectively. On receiving a ready signal on the incident link, which corresponds to the receipt of a stone, the P-module proceeds to inform the associated transitions about the arrival of the stone by sending a ready signal on the enable sublink. A ready signal on an enable sublink is referred to as an enable signal. The EP-module associated with the place fans out the enable signal to the transitions which have this place as an input place. In order to claim the stone at the place, a transition (structure) must acknowledge the enable signal. The stone is claimed by the transition that acknowledges the enable signal first. The EP-module associated with the place transmits a stone signal (i.e. a ready signal on the stone sublink) to the transition which claims the stone, and disable signals to the other transitions. It will be seen later that the transitions are not in a race at

the EP-module because the transition which will be the first to acknowledge the enable signal is picked by the conflict structure; the conflict structure does not permit the other transitions to acknowledge enable signals until the chosen transition has claimed the stones at the input places.

The IT-module acts as a conjunctive fan-in module, it sends an enable signal to the T-module only upon receipt of enable signals on all incident links. In terms of the net, it means that the transition is not enabled until all input places of the transition have stones.

Upon receiving an enable signal, the T-module sends a request to the constraint structure by sending a ready signal on the enable sublink of the double link at port 1 of the module. When the constraint structure gives permission to the transition to proceed by acknowledging the enable signal, the T-module sends a request to the conflict structure by sending an enable signal on the double link at port 2. When the conflict structure gives permission to the transition to proceed (this corresponds to selection of this transition over the transition conflicting with it), the T-module proceeds with initiation of the transition in the following way. To claim the stones at the input places, the module acknowledges the enable signal on link 0. This acknowledge signal reaches the input places through IT and EP modules. The acknowledge signals from this transition are guaranteed to be the first to arrive at the input places as the conflict structure does not permit the transitions that are in conflict to acknowledge enable signals

until they are disabled by the transition in the process of claiming the stones.

A disable signal (i.e. a ready signal on the disable link) to a transition structure from a place informs the transition that the stone at that place has been claimed by some other transition. The disable signal resets the structure associated with the transition to the same condition it would be in if the enable signal had not been received. After this resetting the transition acknowledges (receipt of) the enable and disable signals.

Thus the transition which is given permission by the conflict structure claims the stones at its input places and disables the other transitions which have those places as input places, that is it disables the conflicting transitions. As soon as the conflicting transitions are disabled and they have acknowledged the enable and disable signals from the places, the places send stone signals to the transitions (a stone signal is a ready signal on the stone sublink of a triple link) signifying that the stones have been reserved for it. On receiving a stone signal from each of the input places, the IT-module associated with the transition sends a stone signal to the T-module associated with the transition. The T-module then disables the part of the conflict structure associated with it (because it does not need the help of the conflict structure - remember the conflicting transitions have been disabled already) and initiates the transition by sending a ready signal on the link 3 of the module (the T-module). If this transition is an output transition, this ready signal initiates the associated event in

the external world, and when the occurrence of the event is completed, as indicated by an acknowledge signal on the link, the T-module terminates the transition by acknowledging the stone signal on link 0 and sending a ready signal on the link 4. The acknowledge signal on link 0 has the effect of removing stones from the input places (the place structure gets reset to a stable condition in which it can accept signals corresponding to stones), and the ready signal on link 4 has the effect of putting stones at the output places.

The action of a transition structure is completed when the stones put out by the transition are used up by other transitions. When this happens, the T-module receives an acknowledge signal on link 4. In response to this acknowledge signal, the T-module disables the constraint structure associated with the transition, and the action of the transition structure is completed.

The nature of the conflict structure, the constraint structure and the initialization structure is briefly described below. A detailed description of these structures is given later in this chapter. The conflict structure (figure 4.32) has a conflict module for each conflict cluster of transitions in the precedence structure. A conflict cluster is a set of transitions which mutually conflict, and therefore only one of them may proceed at a time. When conflicting transitions send requests to the conflict structure, the conflict module picks one of them arbitrarily - the picked transition is allowed to proceed while the other transitions are blocked. After a transition clears through

all of the conflict clusters it is involved in, the conflict structure allows it to proceed with initiation. Before it initiates, the transition that is allowed to proceed by the conflict structure disables the transitions which conflict with it. Thus the transitions which are disallowed are disabled by the transitions which are allowed to proceed.

The constraint structure (figure 4.31) ensures that the constraints specified in the constraint set are not violated. However, the reason why it is shown to have inputs from transitions rather than places is that the only way to prevent a stone from getting into a place is by preventing a transition from putting a stone into it. The constraint structure allows a transition to proceed only if its output places can be added to the places already active without causing the set of active places to become inadmissible. This is achieved by means of constraint modules associated with the constraints in the set of constraints. Each constraint module ensures that the particular constraint it represents is not violated.

The initialization structure (figure 4.33) puts initial stones in places in the precedence structure just like transitions and resets itself as the stones are used up by transitions.

4.7 Naming Scheme for Signals

To facilitate discussion, a uniform naming scheme has been adopted for signals. The ports of modules are uniquely identified by m_k:p where m_k refers to the module and p identifies the port of the module. Moreover m_k:px.t refers to a signal on sublink x at port

m_k:px.t

m_k name of a module.

p name of the port of the module.

x name of the sub-link i.e. null, c, e, d or s.

t type of the signal

r - ready signal

a - acknowledge signal

an- negative acknowledge signal

ap- positive acknowledge signal

Figure 4.8 Signal Naming Scheme

p of the module m_k, and t gives the type of the signal that is, t indicates whether the signal is a ready signal, an acknowledge signal, a positive acknowledge signal or a negative acknowledge signal (figure 4.8). Therefore a signal m_k:le.an refers to a negative acknowledge signal on sublink e of port 1 of module m_k. If the link emerges from this port, then this signal is an input signal, otherwise it is an output signal. The link at port j of a module is often referred to as link j when the module is identifiable from the context.

The types of modules in terms of which the coordination structures are derived are specified using a formalism called P-nets. This formalism should be viewed as a linguistic means for specifying the modules; natural language is not satisfactory for this purpose because specification of modules in them is both lengthy and imprecise. P-nets are essentially coordination nets written in a different form. These nets are introduced in the next section.

4.8 P-nets

Definition A P-net is a directed graph defined by a tuple

$\ll N, P, L, B^0 \gg, Ct >$ where

$N = \{n_1, \ldots, n_n\}$ is a finite set of nodes of type "signal" (in the interpretation of the nets signals are associated with these nodes).

$P = \{p_1, \ldots, p_m\}$ is a finite set of nodes of type "place"

$L = \{\ell_1, \ldots, \ell_r\}$ is a finite set of directed links of the form $<x,y>$ which connect signal nodes to other signal nodes, places to signal nodes and signal nodes to places (note that places are not directly connected to other places by links).

$B^0 \subset P$ is the set of places having stones initially.

$Ct \subset \mathcal{P}(P)$ is a contraint set over nodes (places) in P.

A link is said to be a precedence link if it connects a signal node to another signal node, a conflict link if it connects a place node to a signal node, and a loader link if it connects a signal node to a place node. That is, a link $< x,y >$ is a precedence link if $x \in N$ and $y \in N$, a conflict link if $x \in P$ and $y \in N$, and a loader link if $x \in N$ and $y \in P$. To enhance the transparency of the P-nets these links are drawn differently - a precedence link is drawn with a standard arrow, a conflict link is drawn with a solid arrow, a and the loader link is drawn dotted.

An interpretation for a P-net refers to an association of signals, not necessarily distinct, with the signal nodes (figure 4.9 and 4.11). Some of the signal nodes may not be associated with any signal at all. A node which is not associated with any signal is said to be a null node. In P-nets, the signals associated with signal nodes are written in place of the nodes themselves.

In the simulation of P-nets, the signal nodes act as transitions in coordination nets. The signal nodes of a P-net are of three types: i) input signal nodes ii) output signal nodes and iii) internal nodes. In a P-net specifying an asynchronous module or an asynchronous system the signals associated with input signal nodes correspond to the input signals the system can accept and the signals associated with the output signal nodes are the signals which the system may send out. An input signal node can occur only when the associated input signal is received by the system, and when it occurs the node is

said to accept that input signal. An input signal can be accepted only by one input signal node. Thus even when more than one input signal node associated with an input signal is ready to occur, only one of them can occur. An output signal node occurs whenever it can occur, and its occurrence causes the system to send out the associated signal.

A P-net corresponds directly to a coordination net. The coordination net corresponding to a P-net is obtained by i) replacing the signal nodes by transitions ii) replacing the precedence links by places each with one incident link and one emerging link, iii) introducing a place for each signal and iv) drawing arcs from the places corresponding to input signals to the assocated input transitions, and drawing arcs from the output transitions to the places corresponding to the associated output signals (figure 4.10). The place nodes in the net are left untouched. A signal is sent to the system by placing a stone in the place associated with that input signal, and the net sends out signals by placing stones in the places associated with the output signals; the input-output places are the boundaries through which the system interacts with other systems connected to it. A P-net is merely a concise way of drawing the coordination nets (figures 4.9 and 4.10); therefore in a formal sense, the simulation of a P-net corresponds to the simulation of the coordination net it represents. Simulation of a P-net can also be described directly by the following rules:

i) Initially only those signal nodes which have no incident precedence links and all of whose input places have stones are ready to occur.

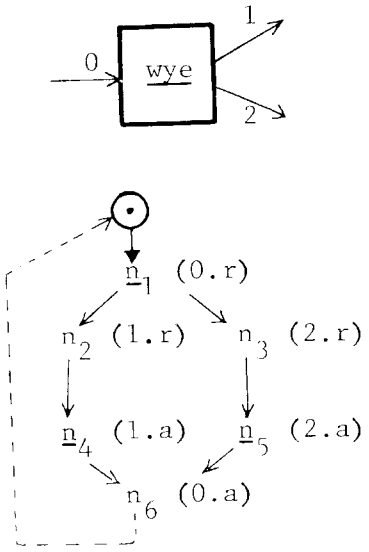


Figure 4.9 A P-net definition of the wye Module

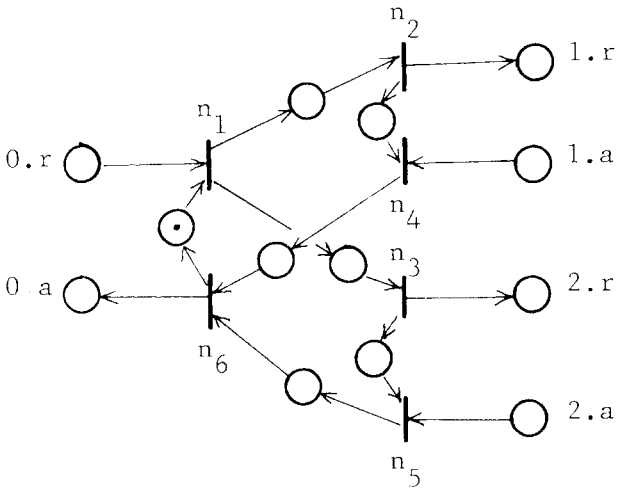


Figure 4.10 Coordination net corresponding to the P-net of Figure 4.9

ii) An input signal node does not occur until the associated input signal is received by the system, and in the process of occuring the input signal node is said to use up the signal.

iii) When a signal node occurs, it gives permission to all signal nodes that are its immediate successors by precedence links to occur, and puts stones in its output places.

iv) A signal node becomes ready to occur when it has permission to occur from all immediate predecessor signal nodes, its input places have stones and, if it is an input signal node, the system has received the associated input signal but this input signal has not been accepted by some other signal node. If the stone from an input place of a ready signal node should be removed by some other signal node, the ready status of the node is suspended until the place gets a new stone. Similarly if the input signal associated with a ready input signal node should be accepted by some other signal node, the ready status of the node is suspended until a new input signal is received.

v) The occurrence of an output signal node causes the system to send out the output signal associated with that node.

Figures 4.9 and 4.11 show two P-nets. The P-net of figure 4.9 specifies the wye module discussed ealier and figure 4.11 specifies a more complicated module called the IT-module. The operation of these P-nets is explained below. In these P-nets the input signals are underlined as an aid to understanding.

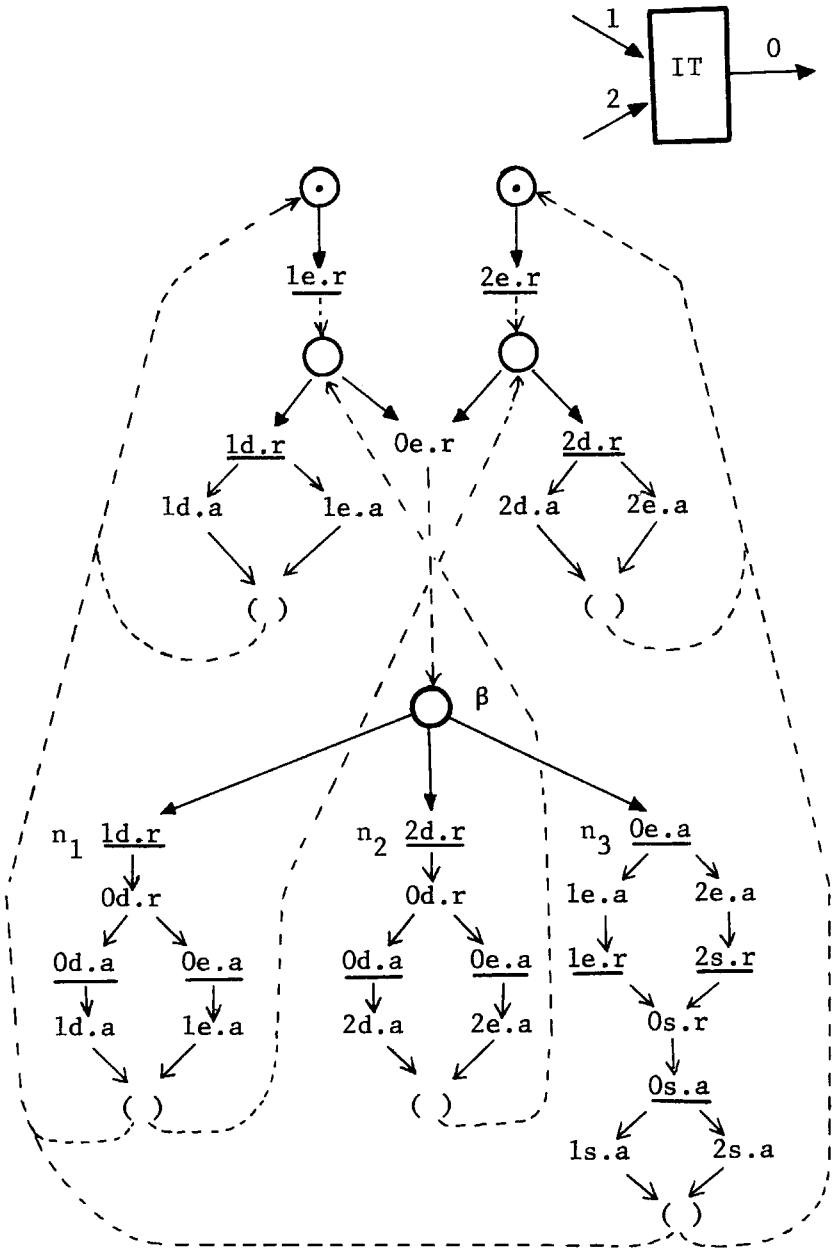


Figure 4.11 A P-net definition of the IT-module

In the P-net of figure 4.9, no signal node is ready to occur until signal 0.r is received, that is a ready signal is recieved on link 0 of the module. When this ready signal is received, signal node n_1 accepts the input signal by occurring, and gives nodes n_2 and n_3 permission to occur. Nodes n_2 and n_3 are output nodes; their occurrence causes the module to send out ready signals on links 1 and 2. When these ready signals are acknowledged, signal nodes n_4 and n_5 occur permitting node n_6 to occur. The occurrence of node n_6 causes the module to acknowledge the ready signal on link 0, and the action of the module is completed as it returns to its initial condition.

Figure 4.11 shows a P-net for a more complicated module. This P-net has situations where signal nodes conflict. For example, signal nodes n_1 , n_2 and n_3 conflict as they share a common input place β. When place β has a stone, one of these nodes occurs depending on which signal is received first. In the case that signals associated with more than one of these nodes are received at the same time, the module picks one of the nodes arbitrarily.

4.9 The Modules

The asynchronous modules used in construction of coordination nets are specified in this section.

P-module

A P-module is used in the construction of a place structure and in transforming a simple link into a triple link. The P-module has one incident simple link and one emergent triple link. A schematic diagram of the P-module and its P-net specification are shown in figure 4.12 . Upon receiving a ready signal on the incident link, the module sends an enable signal on the emergent link. When this enable signal is acknowledged, the module sends a stone signal on the emergent link, and finally when the stone signal is acknowledged, the module acknowledges the ready signal on the incident link and returns to its initial condition. Note that the disable sublink of the emergent link is not used by the module.

IP-module

The IP-module is for disjunctive fan-in of simple links. It is used in conjunction with a P-module to accomodate multiple incident links - whence the name "IP". The function of the IP-module is to connect (logically) an incident link to the emergent link when a ready signal is received on the incident link. The connection remains in effect for one communication cycle on the link. The P-net specification of the IP-module is shown in figure 4.13 .

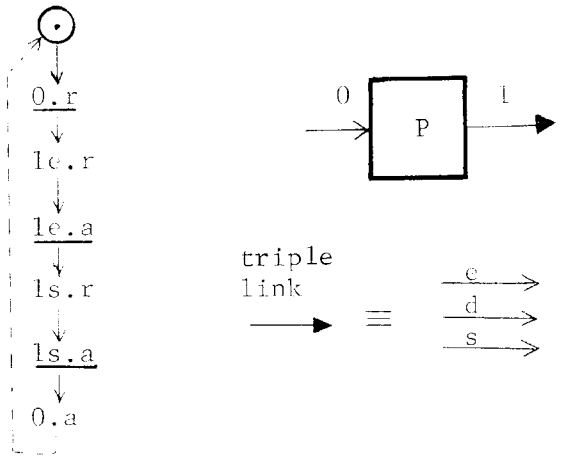


Figure 4.12 The P-module

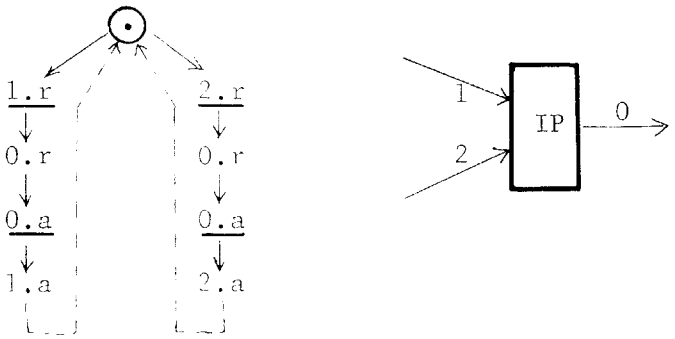


Figure 4.13 The IP-module

EP-module

The EP-module is for fan-out of the triple link emergent from a P-module. The function of the EP-module is to send enable signals on the emergent links when an enable signal is sent to it on the incident link, and to send a stone signal to one of the emergent links and a disable signal to the other emergent link when a stone signal is sent to it depending on which of the emergent links acknowledges the enable signal first (figure 4.14). The operation of the EP-module is described in detail below.

On receiving an enable signal on the incident link, the module sends enable signals on the emergent links. In the case of an EP-module in a place structure of a coordination net, the enable signal on the incident link indicates that a stone has arrived at the place, and the information about the arrival of the stone should be sent to the transitions; the EP-module sends this information to the transitions by sending enable signals on the emergent links. At this point one of two things may happen: i) some transition may acknowledge the enable signal indicating that it has claimed the stone, or ii) a disable signal may be received on the incident link indicating that the stone has been claimed by some transition other than those associated with the emergent links. If a disable signal arrives first, the module sends disable signals on the emergent links, and when these disable signals and the enable signals are acknowledged, it acknowledges the enable and disable signals on the incident link.

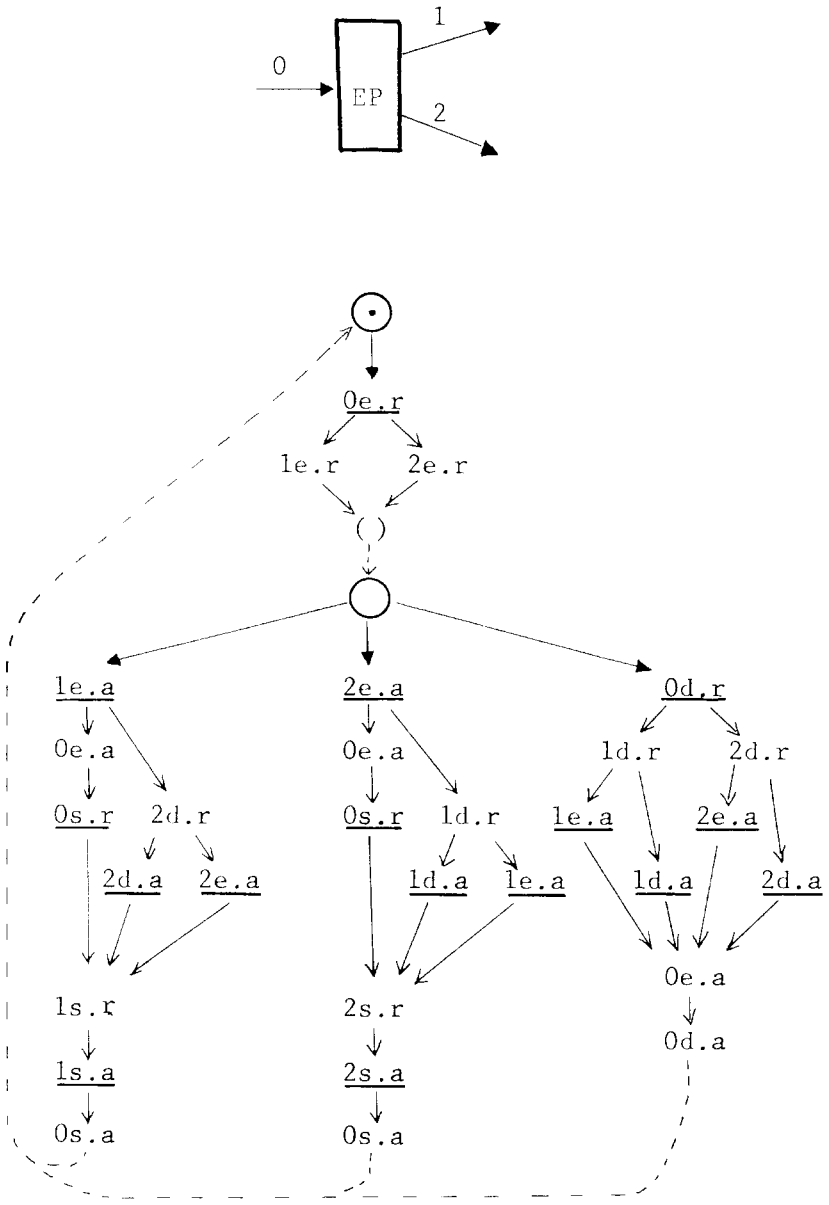


Figure 4.14 The EP-module

In case a disable signal is not received on the incident link before an enable signal is acknowledged on an emergent link, the following happens. If the acknowledgement is received on link 1, the module is in a way claimed by that link, and therefore, the module acknowledges the enable signal on the incident link and sends a disable signal on the other emergent link. The module then waits for a stone signal on the incident link and for the completion of the communication cycle on the other emergent link. When this happens, the module sends a stone signal on link 1, the link which claimed the module. When this stone signal is acknowledged, the module acknowledges the stone signal on the incident link and returns to its initial condition.

A communication cycle on a triple link involving a disable signal is said to be a void cycle, and one involving a stone signal is said to be a stone cycle. Thus the operation of the module can be restated as follows: if the module is given a void cycle on the incident link, the module gives void cycles to the emerging links, while if the module is given a stone cycle then it gives a stone cycle to that emerging link which acknowledges the enable signal first, and a void cycle to the other link.

IT-module

The IT-module is for conjunctive fan-in of triple links incident on the transition structures. Briefly speaking, the operation of the IT-module is to produce a stone cycle on the emergent link when it gets a stone cycle from each of its incident links. The P-net specification

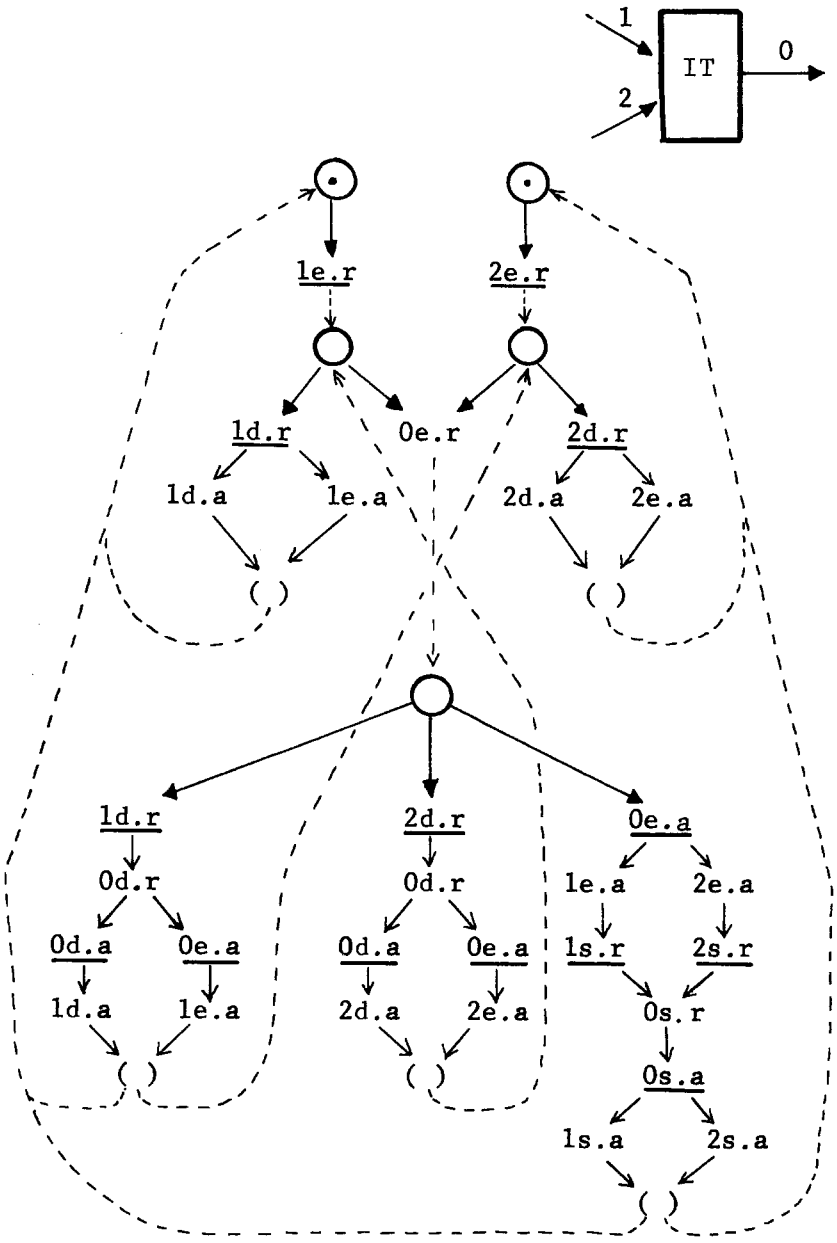


Figure 4.15 The IT-module

of the IT-module is given in figure 4.15. The module is called an IT module because the module is used to converge the incident links on the T-module in the transition structure.

When the module receives enable signals on both incident links, it sends an enable signal on the emergent link and waits for the enable signal to be acknowledged. If the module receives a disable signal on any of the incident links while it is waiting for the enable signal on the emergent link to be acknowledged, it takes the following action: Consider the case when the module receives a disable signal on link 1. In this case, the module sends a disable signal on the emergent link and waits for the enable and disable signals to be acknowledged. The effect of this disable signal on the structure connected to the emergent link is to request acknowledgement of the enable and disable signals. On receiving these acknowledge signals, the module acknowledges the enable and disable signals on link 1 and returns to the condition it would be in if the enable signal (on link 1) had not been received.

If no such disable signal is received before the enable signal on the emerging link is acknowledged, the module proceeds with the following action. It acknowledges the enable signals on both incident links and waits for stone signals from them. When stone signals are received, it sends a stone signal on the emergent link and waits for the stone signal to be acknowledged. When the stone signal is acknowledged, the module acknowledges the stone signals on both incident links and returns to the initial condition.

T-module

The T-module plays a central role in the transition structure. In addition to the T-module, a transition structure has an IT-module for a conjunctive fan-in of incident links and an ET-module for fan-out of the emergent link; link 0 of the T-module comes from the IT-module and link 4 goes to the ET-module. Links 1 and 2 of the T-module go to the constraint structure and conflict structure respectively (figures 4.7 and 4.16). If the transition is an output transition, link 3 of the module goes to the external world, otherwise it is terminated by a sink module.

The function of the T-module is to get permission from the constraint and conflict structures on behalf of the transition it represents and to initiate and terminate the transition. A P-net specification of the T-module is shown in figure 4.16 . If the reader remembers the operation of the transition structure from the overview given earlier and is able to understand the P-net of the T-module he may skip the following explanation of the operation of the T-module.

When an enable signal is sent to the T-module on link 0, it sends a request to the constraint structure by sending an enable signal on link 1, and waits for this signal to be acknowledged, that is it waits for a permission from the constraint structure. While the module is waiting for this signal, it may receive a disable signal on link 0. If such a disable signal is received, the module sends a disable signal on link 1 to cancel the request to the constraint structure. As a result

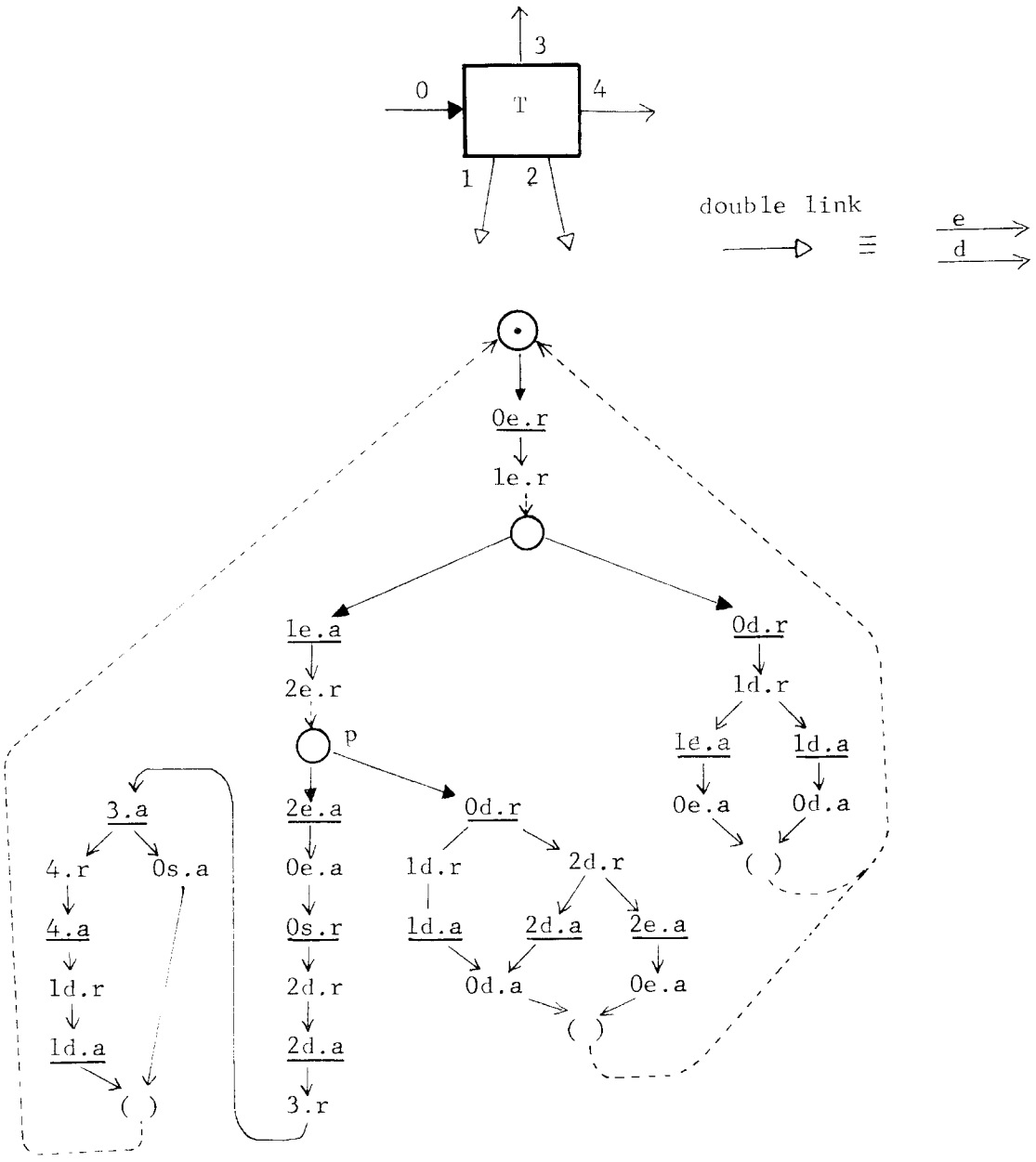
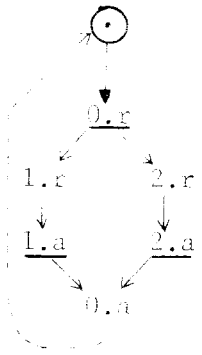


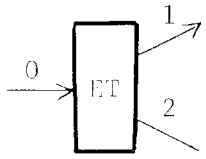
Figure 4.16 The T-module

the constraint structure acknowledges both the enable and the disable signals. On receiving these acknowledge signals the module acknowledges the enable and disable signals on link 0.

If a disable signal is not received on link 0 before the enable signal on link 1 is acknowledged, the module sends an enable signal to the conflict structure on link 2 and waits for the signal to be acknowledged. As in the previous case if a disable signal is received on link 0 before the enable signal is acknowledged, the module sends a disable signal on link 2. The effect of this disable signal on the conflict structure is to obtain acknowledgement of the enable and disable signals. When these acknowledge signals are received, the module acknowledges the enable and disable signals on link 0.

If a disable signal is not received on link 0, the module proceeds with the following action on receiving an acknowledgement for the enable signal on link 2. It acknowledges the enable signal on link 0 and waits for a stone signal on that link. On receiving the stone signal, the module sends a disable signal on link 2 to complete the communication cycle on link 2, and on completion of the cycle, that is on receiving acknowledgement for the disable signal, the module initiates the associated transition by sending a ready signal on link 3, and waits for the ready signal to be acknowledged. When the ready signal is acknowledged, the module proceeds with the termination of the transition by acknowledging the stone signal on link 0 and sending a ready signal on link 4. The module then waits for the ready signal on link 4





ET-module is also called a wye module

Figure 4.17 The ET-module

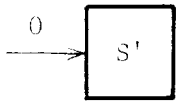


Figure 4.18 The Sink Module S'

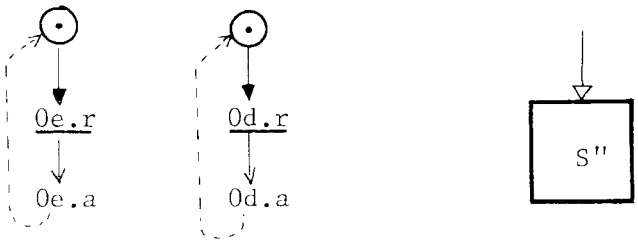


Figure 4.19 The Sink Module S''

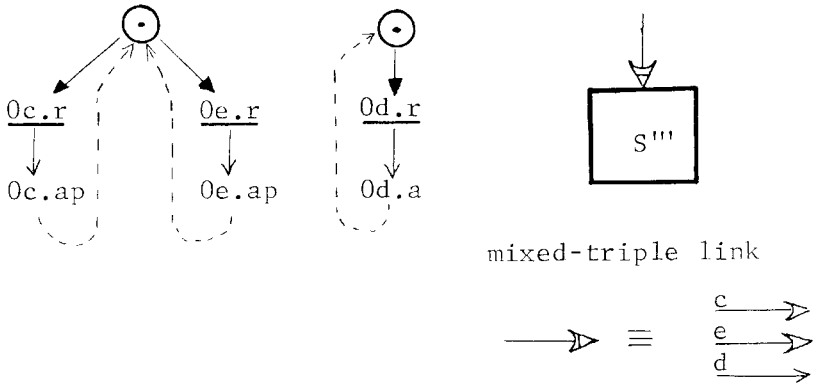


Figure 4.20 The Sink Module S'''

Operation of Constraint, Conflict and Initialization Structures

As the modules discussed next are used in these structures, an overview of the arrangement of modules in them and some understanding of their operation should be helpful. The arrangement of modules in these structures is shown in figures 4.31, 4.32 and 4.33 respectively.

In the constraint structure (see figure 4.31) the mixed-triple links are associated with the classes of constraint equivalent places (see Chapter 3), while the constraint modules are associated with the constraints specified by the constraint set of the net. The mixed-triple links originate from R-modules. The double links from constraint equivalent transitions are combined in an IR-module before giong to the associated R-module (recall that output places of constraint equivalent transitions are constraint equivalent).

A transition sends a request to the constraint structure by sending an enable signal. If the constraints associated with the output places of the transition are already in force on account of some other constraint equivalent transition, the IR-module grants the request immediately by acknowledging the enable signal. If the constraints are not in force, the IR-module forwards the enable signal to the R-module to begin the process of establishing the constraints. Upon receiving the request, the R-module sends a check signal on the mixed-triple link. If the check signal is acknowledged negatively, the R-module sends a fresh check signal; this continues until the R-module either gets a positive acknowledgement for the check signal or receives a disable signal on the incident link. Upon receiving a positive acknowledgement for the check

signal the R-module sends an enable signal on the mixed-triple link. A positive acknowledgement of the enable signal means that the constraints associated with the transition have come into force. The R-module then acknowledges the enable signal on its incident link, and the acknowledge signal reaches the transition through the IR-module. A negative acknowledgement for the enable signal implies that the R-module should start all over again by sending a check signal.

From the time the enable signal is acknowledged positively on the mixed-triple link to the time a disable signal is received on it, the link is said to be active. A contraint module ensures that not all of the links incident on it are active at the same time. In this way the constraint modules enforce the constraints associated with them.

The conflict structure (figure 4.32) has double links, associated with transitions, which pass through conflict modules. In the conflict structure, there is one conflict module for each conflict cluster of transitions in the net. Two conflicting transitions, therefore, have atleast one conflict module in common. Thus if two conflicting transitions send enable signals to the conflict structure, only one of them is permitted to proceed as a conflict module common to them permits enable signal from only one of them to proceed; the other enable signal is blocked at that module.

The task of the initialization structure is to set up the initial constraints and to send signals equivalent to stones to the places in the structure that correspond to the places in the net that have initial stones.

IR-module

The IR-module is used to connect the double links from (constraint equivalent) transitions to the R-module that is associated with these transitions. The function of the IR-module is to request the R-module to set up the associated constraints upon receiving a request from a transition. In a way the IR-module performs a logical OR operation - it causes the constraints to be established when a request is received from a transition associated with either of the incident links, and it relinquishes the constraints only when none of the transitions need the constraints. The P-net specification of the IR-module is shown in figure 4.21 .

When the module is in the idle condition, and a transition sends a request to it by sending an enable signal on one of the incident links, the module sends out an enable signal on the emergent link and waits for the signal to be acknowledged. When the module receives this acknowledge signal it acknowledges the enable signal on the incident link making the link active. If the other link also sends an enable signal at this point, the module immediately acknowledges this signal; the emergent link of the module is now in the active condition on account of both incident links. So long as either of the incident links is active the module does not disable its emergent link. When both links are disabled, the module disables the emergent link by sending out a disable signal. This disable signal is acknowledged in due time and the module returns to its initial condition.

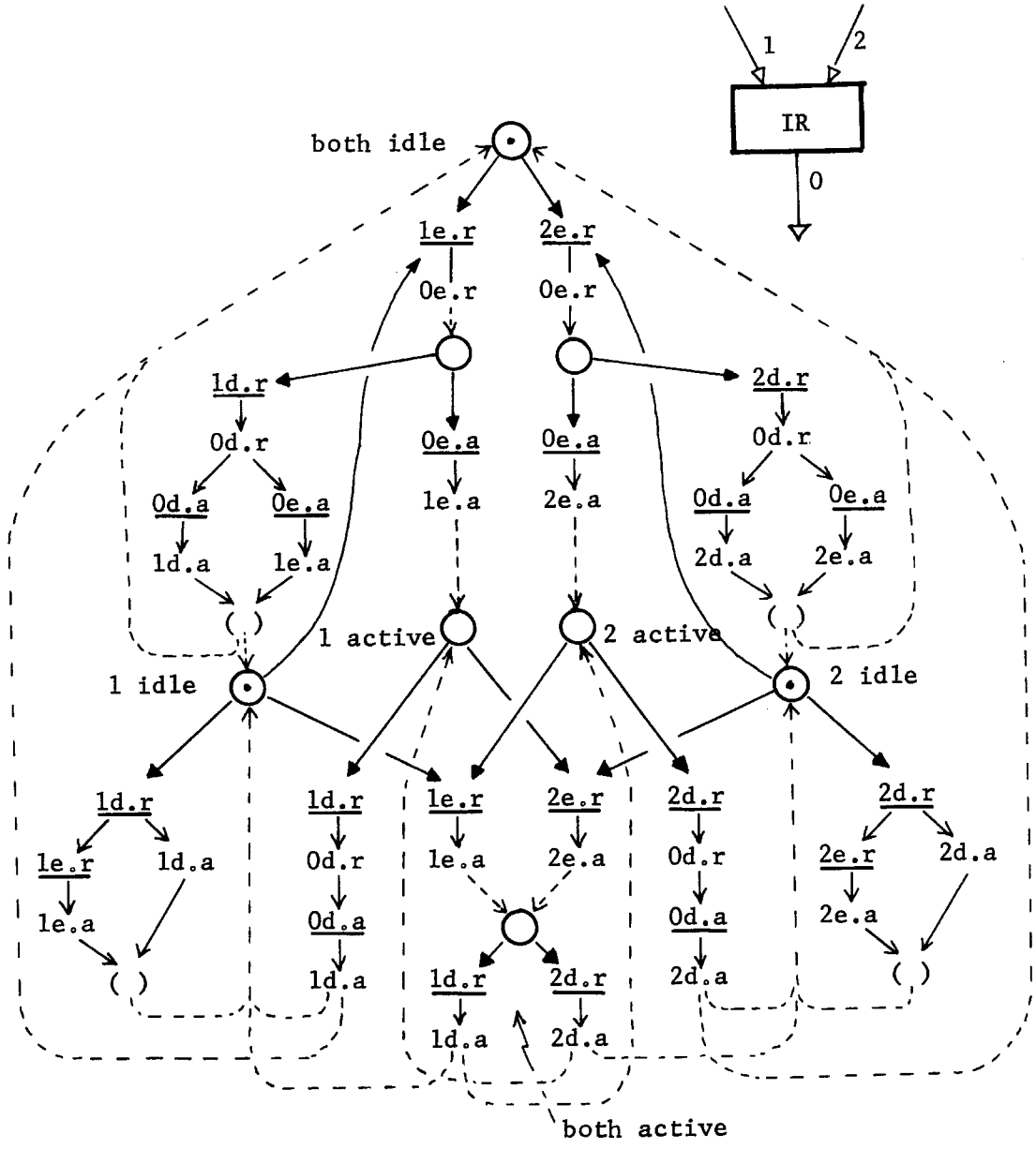


Figure 4.21 The IR-module

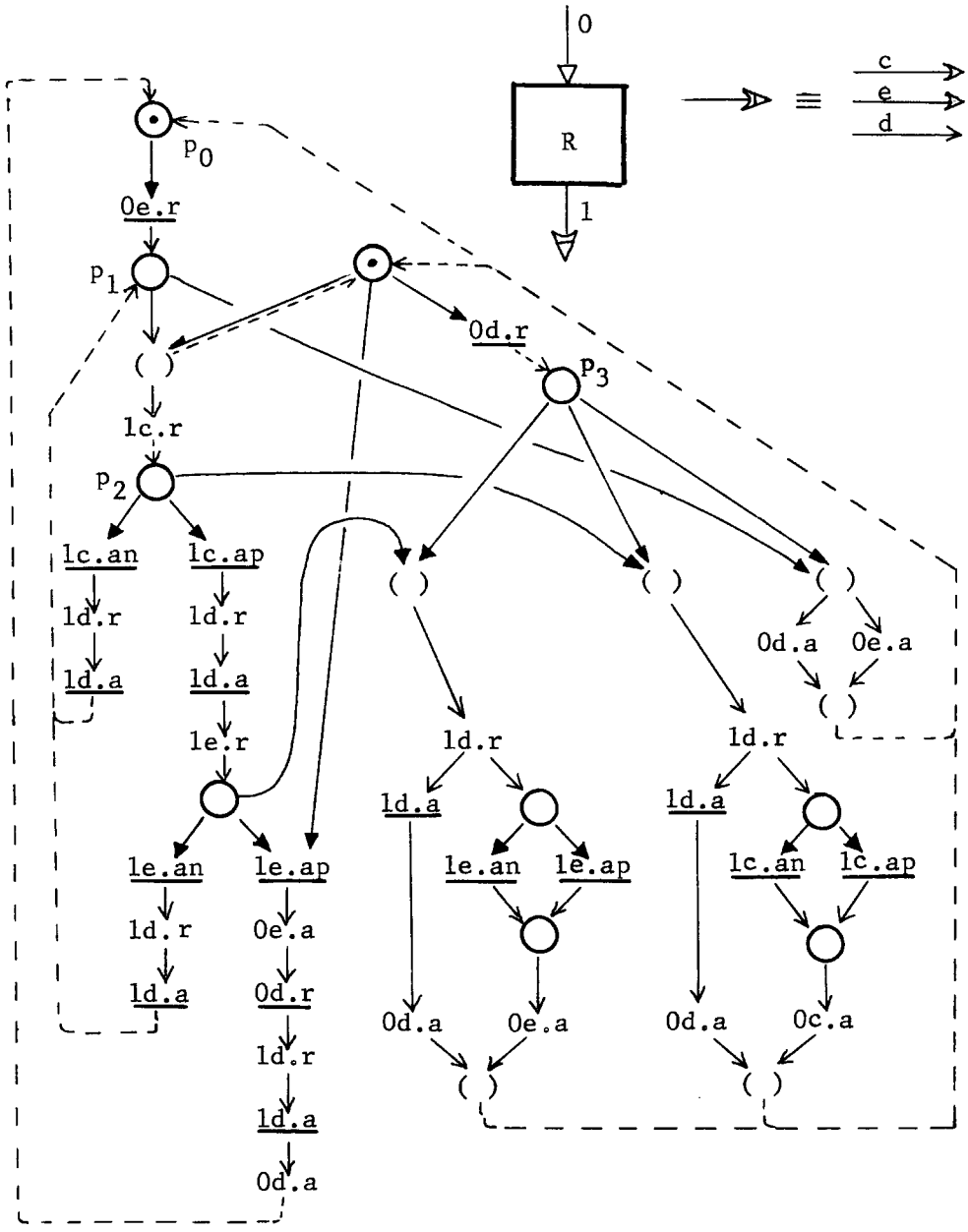
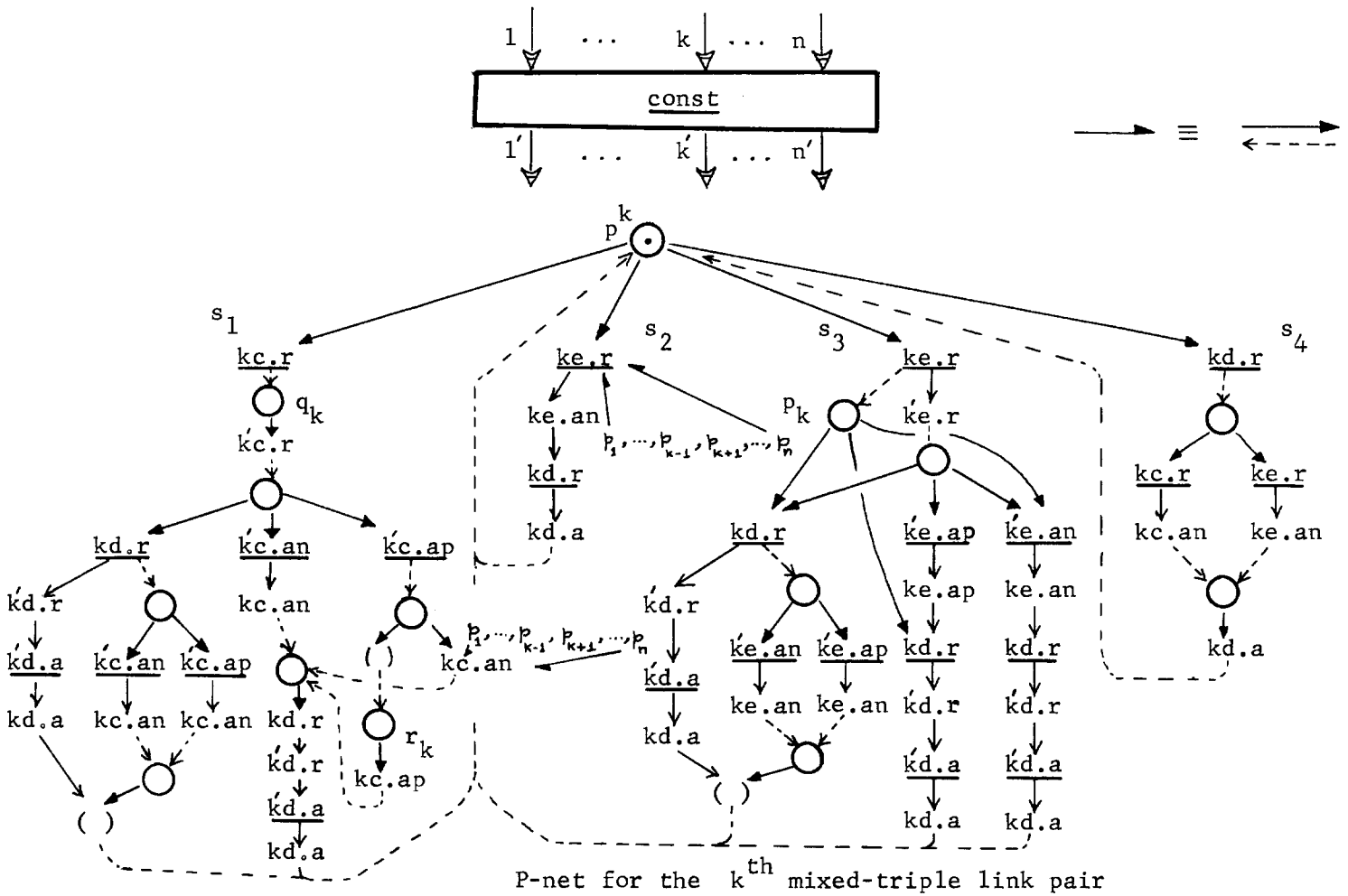


Figure 4.22 The R-module

Constraint Module

The constraint (const) module has a number of pairs of incident-emergent mixed-triple links (figure 4.23) . If a mixed-triple link is said to be in enabled condition during the interval between the occurrence of an enable signal and the occurrence of either a negative acknowledge signal on the enable sublink or a disable signal on the disable sublink, a simplefied statement of the operation of the constraint module is possible. The constraint module terminates signals on the incident links to ensure that not all of the emergent links of the module are in enabled condition at the same time. Alternatively, the module prevents all of the incident links from being active at the same time where a link is said to be active if the enable signal has been acknowledged positively and the disable signal has not been received. The P-net specification of the module is given in figure 4.23 . In the following discussion the constraint module is said to be saturated when all but one of its emergent links are either enabled or active.

In the constraint structure the R-module sends a check signal on the mixed-triple link to determine if and when it should send an enable signal to try to bring into force the constraints associated with it; the R-module sends an enable signal only on receiving a positive acknowledgement for the check signal. The constraint module, if it is



P-net for the k^{th} mixed-triple link pair

$$Ct = \{\{p_1,\ldots,p_n\},\ \{r_1,p_2,\ldots,p_n\},\ \ldots\ ,\{p_1,\ldots,p_{k-1},r_k,p_{k+1},\ldots,p_n\},\ \ldots\ ,\{p_1,\ldots,p_{n-1},r_n\},$$
$$\{q_1,p_2,\ldots,p_n\},\ \ldots\ ,\{p_1,\ldots,p_{k-1},q_k,p_{k+1},\ldots,p_n\},\ \ldots\ ,\{p_1,\ldots,p_{n-1},q_n\}\}$$

Figure 4.23 The Constraint Module

not saturated, sends the check signal to the emergent link, otherwise it blocks the check signal until it becomes unsaturated. On sending a check signal on the emergent link if it receives a positive acknowledgement, it returns a positive acknowledgement for the check signal on the incident link. On the other hand if it receives a negative acknowledgement or it is saturated when the acknowledgement is received, it returns a negative acknowledgement for the check signal on the incident link.

In response to an enable signal, the module sends an enable signal on the emergent link if it is not saturated, otherwise it returns a negative acknowledgement for the enable signal immediately.

The effect of a disable signal received on an incident link is to force the part of the module associated with that link to reset to the initial condition after completing the communication cycle on the emergent link.

The constraint module presented here is a composite module in that it has n incident-emergent link pairs. This module can be constructed in a manner similar to that presented in reference [14] .

Conflict Module

The conflict module has a number of incident-emergent double link pairs. The conflict module allows only one of the emergent links to be in enabled condition at a time. If more than one of the incident links send enable signals to the module, it arbitrarily allows only one of

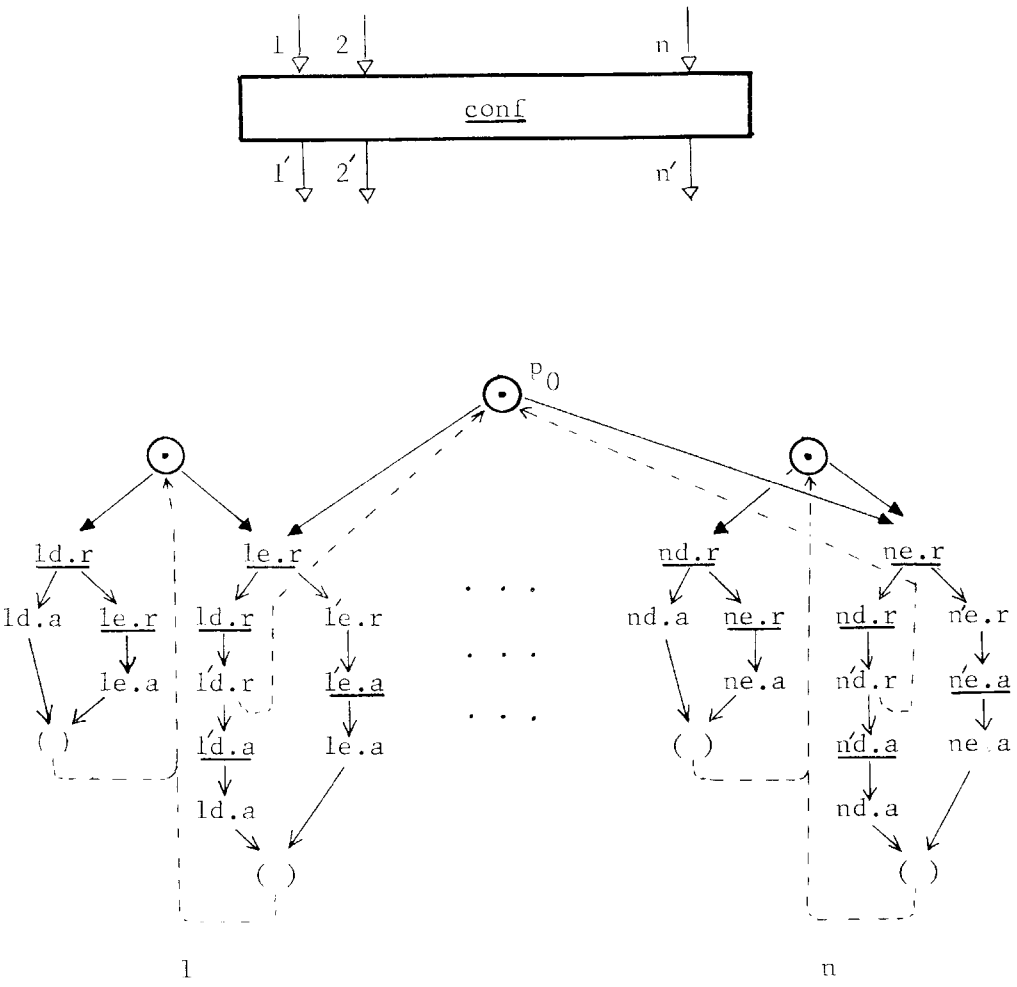


Figure 4.24 The Conflict Module

the enable signals to proceed to the corresponding emergent link and blocks the other enable signals. In this case the module is said to be engaged by that link, and as long as the module is engaged by that link, it blocks enable signals received on the other links. The P-net specification of the conflict module is given in figure 4.24 .

The disable signal on a link instructs the module to acknowledge the enable signal even if it is blocked. If the disable signal is on the link which has engaged the module, the module is released so that it may be engaged by some other link which might be waiting to engage the the module.

The conflict module is also a composite module. It can be constructed from elementary modules in the manner presented in reference [14] .

Junction Module

The junction module froms a conjunctive fan-in of simple links. Upon receiving enable signals on both incident links, it sends an enable signal on the emergnet link and waits for an acknowledge signal, and when the acknowledge signal is received it acknowledges the enable signals on the incident links. The P-net specification of the junction module is shown in figure 4.25 .

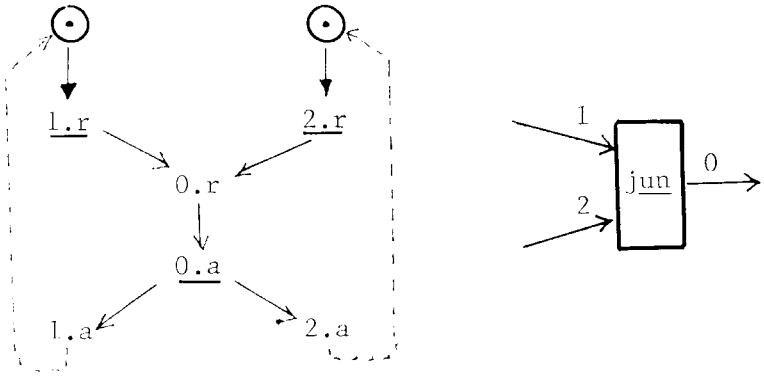


Figure 4.25 The junction Module

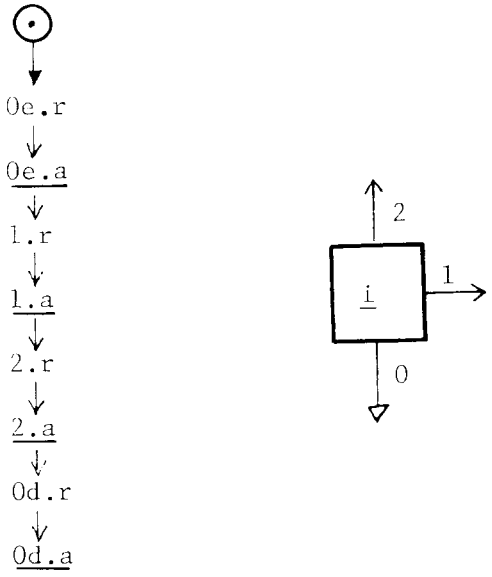


Figure 4.26 The i Module

i-module

The i-module, initialization module, has three links: two single links and one double link (figure 4.26) . The coordination structures use i-modules in their initialization structures (figure 4.33). Link 0 of an i-module goes to the constraint structure, link 2 goes to the places to be initialized and link 1 goes to a junction module (figure 4.33).

To establish the initial constraints, the i-module sends an enable signal to the constraint structure on link 0. When this enable signal is acknowledged, it sends a ready signal to the junction module to indicate that the constraints associated with that i-module have been established. When all i-modules in the initialization structure send ready signals to the junction module, the junction module acknowledges all signals to indicate that the i-modules may proceed to initialize the places by sending ready signals on link 2 of the modules. On receiving acknowledgement for the ready signal on link 2, the module relinquishes its hold on the constraints by sending a disable signal to the constraint structure.

Composite Modules

Unlike the elementary modules which have a predetermined number of links, the composite modules have an arbitrary but finite number of links. The composite modules are constructed from the elementary modules by connecting a number of elementary modules (figure 4.27).

4.10 Hardware Implementation of the Modules

This thesis does not go into hardware implementation of the modules specified in this chapter, but a discussion on this topic is given in Appendix I. Even though the modules communicate with each other asynchronously, they can be synchronous internally. In this type of design, the inputs of the module are sampled at regular intervals, and on the outcome of sampling, the new outputs are produced using a combinational circuit. Modules designed this way are slow in operation. The author believes that a neet implementation of these modules in terms of elementary building blocks, called micro-modules, can be worked out (see Appendix I and reference 15). Micro-modules are elementary circuits for performing logical operations on signals where signals are represented by changes in levels of wires rather then the levels themselves.

4.11 Coordination Structures

The substructures of coordination structures were briefly described in section 4.3 . A detailed description of the structures and a method for obtaining them from the nets are presented in this section.

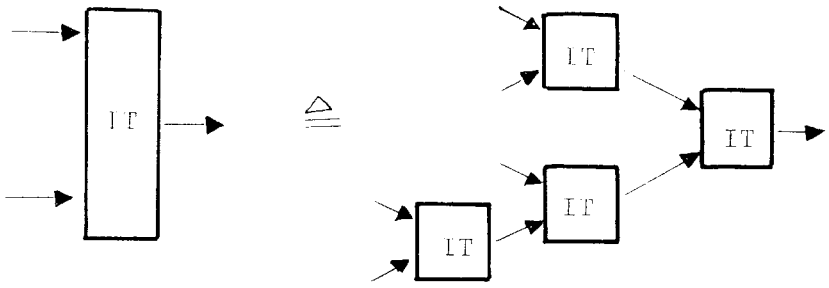


Figure 4.27 A Composite Module

It should be recalled that the precedence structure implements precedence among events making sure that an event does not occur until those events which should precede it have indeed occured, the constraint structure ensures that the constraints are not violated, the conflict structure resolves conflicts among transitions if and when they arise, and the initialization structure establishes the initial condition in the precedence and constraint structures to correspond to the initial condition of the net.

Precedence Structure

A precedence structure is obtained from the coordination net by substituting modular structures called place structures and transition structures for the places and transitions in the net (figures 4.28, 4.29 and 4.30).

In the place structure the P-module is identified with the place in the coordination net. The IP and EP modules accomodate multiple fan-in and fan-out of the incident and emergent links. A place structure has an incident link from the initialization structure if the associated place has an initial stone. The other links incident on a place structure come from transitions for which that place is an output place. The links emergent from a place structure go to the transitions for

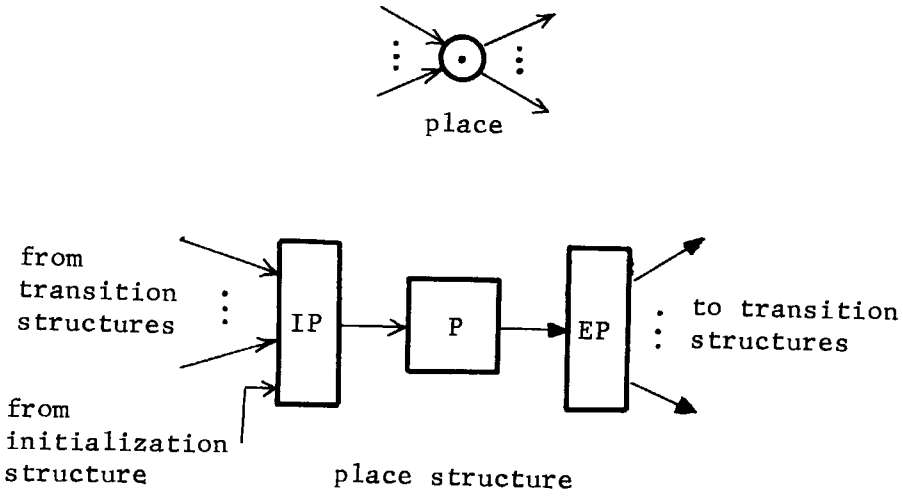


Figure 4.28 Place Structure

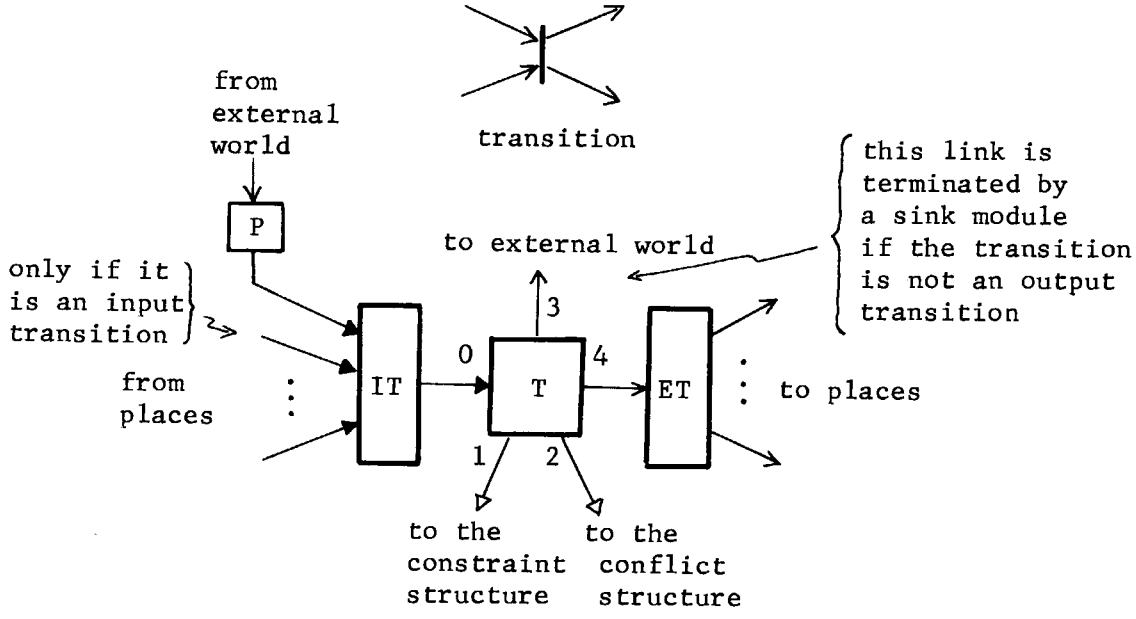


Figure 4.29 Transition Structure

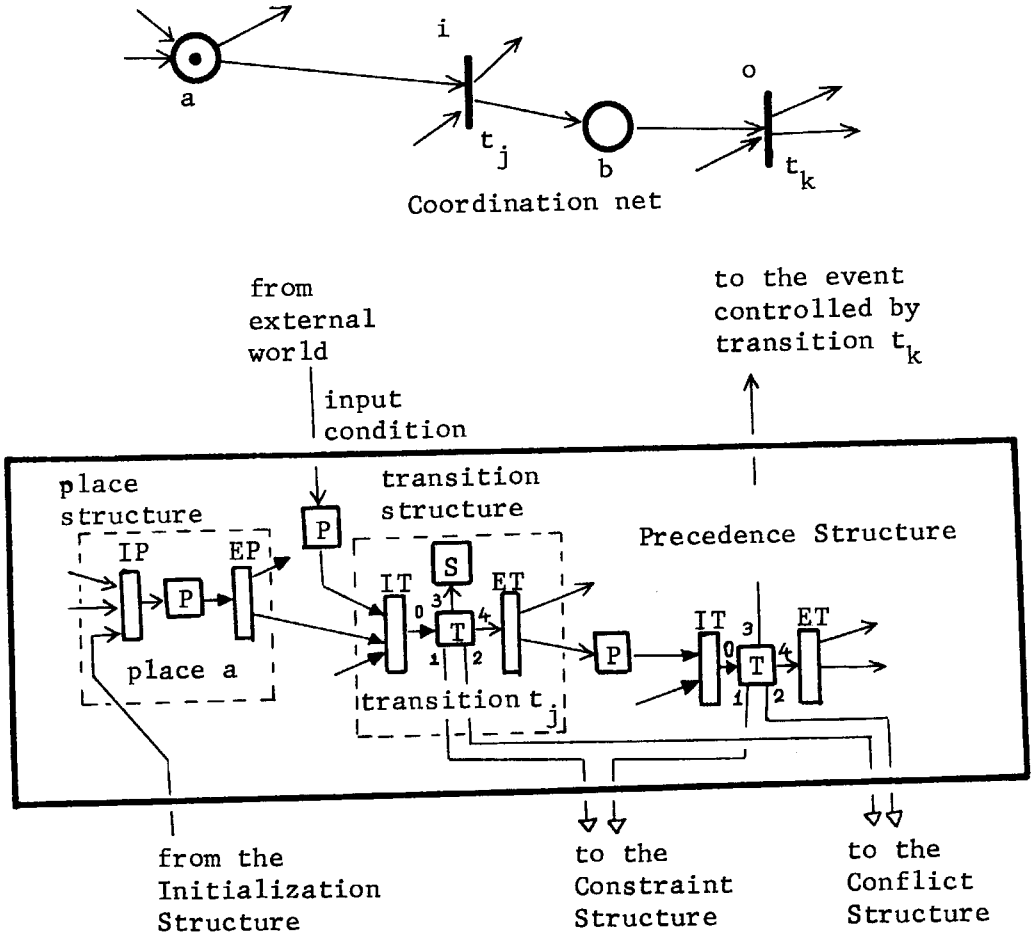


Figure 4.30 The Precedence Structure

which that place is an input place. Thus the links other than the one from the initialization structure correspond to arcs in the net. In a transition structure (figure 4.29), the T-module is identified with a transition in the coordination net, and associated IP and EP modules accomodate multiple fan-in and fan-out of the incident and emergent links. If the transition is an input transition, it has an incident link from the external world which is first converted into a triple link before it is made incident on the IT-module as the incident links of the IT-modules must be triple links. If the transition is an output transition, link 3 of the T-module goes to the external world, otherwise it is terminated by a sink module. Links 1 and 2 of the T-module go to the constraint and conflict structures respectively. The links incident on the structure, other than the link from the external world, come from the place structures corresponding to the input places of the transition, and the emergent links go to the structures corresponding to output places of the transition.

Constraint Structure

The constraint structure has a mixed-triple link associated with each class of constraint equivalent places which participate in constraints; the mixed-triple links originate at R-modules and pass through a set of constraint modules before they are terminated by sink modules (figure 4.31). The IR-modules in the structure accomodate multiple fan-in of incident links on the R-modules which come from transition structures in the precedence structure and initialization modules in

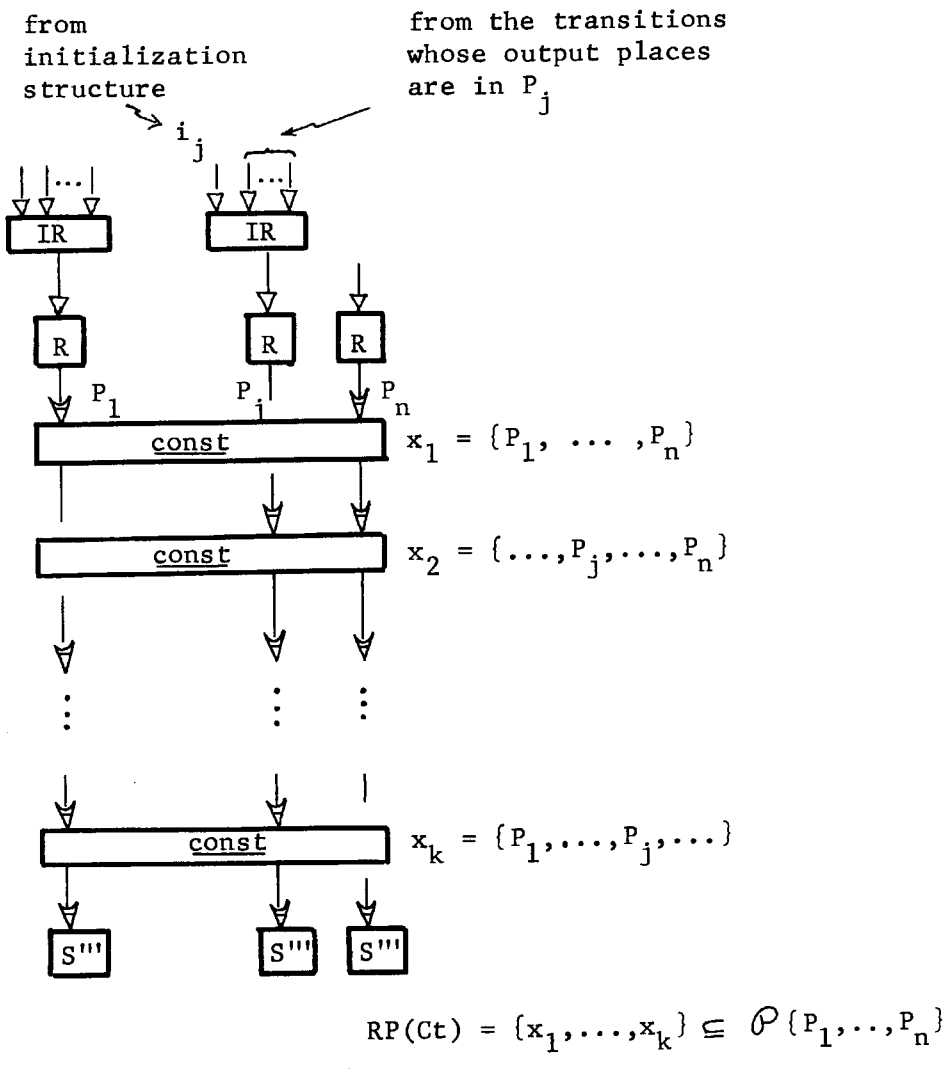


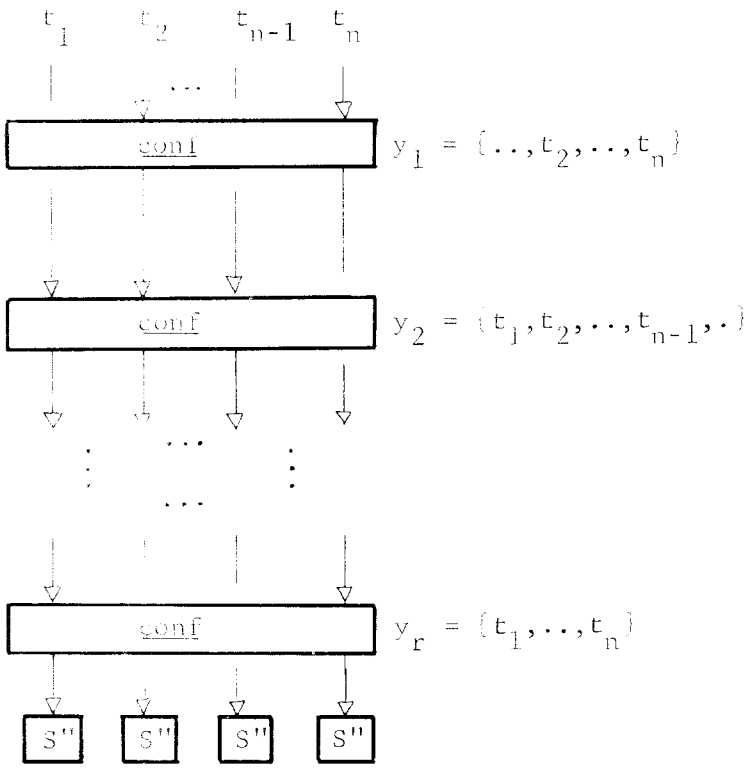
Figure 4.31 Constraint Structure

the initialization structure. The link from a transition structure goes to the R-module associated with the equivalence class of places to which the output places of the transition belong (note that the output places of a transition belong to the same equivalence class because the nets have been transformed into homogeneous nets). The mixed-triple link emergent from the R-module passes through a set of constraint modules which are associated with the constraints in which the equivalent class of places assoicated with the link are involved. The constraint modules correspond to the members in the reduced constraint set RP(Ct) derived from the constraint set Ct (see Chapter 3). Each member of RP(Ct) denotes a set of equivalence classes of places which should not be active at the same time. The operation of the constraint structure is explained in the next section.

Conflict Structure

The conflict modules in the conflict structure are associated with the members in the set of conflict clusters (figure 4.32). It should be recalled that a cluster consists of a maximal set of transitions which mutually conflict. The input link associated with a transition goes through the constraint modules associated with the clusters which involve that transition. After the last conflict module the links are terminated by sink modules. The links associated with transitions which do not conflict with any transition are directly terminated by sink modules.

from transition structures



$C_{fc} = \{y_1,..,y_r\}$

the set of conflict clusters

Figure 4.32 Conflict Structure

Initialization Structure

The initialization structure has $k + 1$ initialization modules corresponding to the $k + 1$ constraint equivalence classes of places in B^0, the set of places which have stones initially (figure 4.33). Link 1 of the initialization modules, except that of the module $\underline{i}_0$, go to the R-modules in the constraint structure associated with the corresponding equivalent classes of places. Link 1 of the initialization module $\underline{i}_0$ is terminated by a sink module because this module corresponds to the places in P_0, the set of places which do not take part in the constraints. Link 2 of the initialization modules go to a junction module which is terminated by a sink module, and link 3 of the modules go to the places associated with them. The function of the junction module is to prevent the initialization module from putting stones in the places before initialization of the constraint structure has been completed.

4.12 Operation of Coordination Structures

The operation of coordination structures was briefly described earlier in this chapter. In this section a more complete description of the operation is presented.

The functions performed by the parts of the coordination structures are as follows: the precedence structure implements the precedence relationship among the events, the constraint structure enforces the constraints, the conflict structure resolves conflicts among transitions and the initialization structure sets up the initial condition

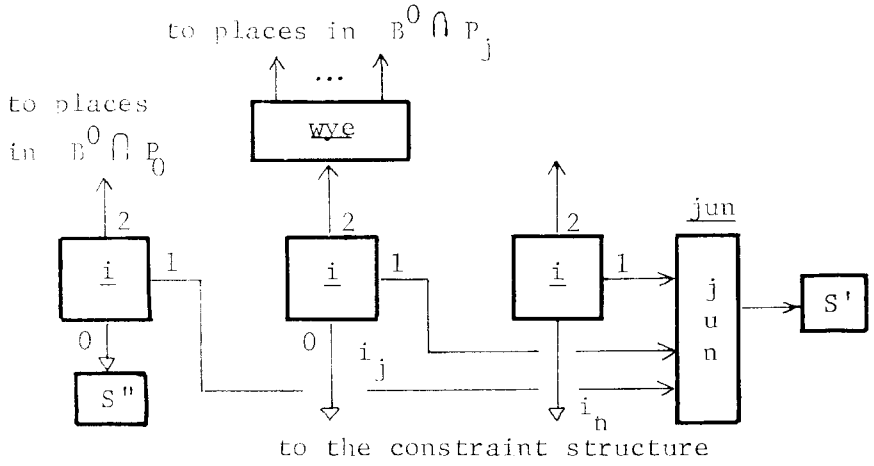


Figure 4.33 Initialization Structure

of the structure.

In the precedence structure, a ready signal on a link incident on a place structure corresponds to a stone. The ready signal reaches the P-module in the structure through the IP-module. Upon receiving the ready signal, the P-module sends enable signals to the transitions to inform them of the arrival of the stone, and depending on which transition acknowledges the enable signal first, the place structure sends a stone signal to that transition and disable signals to others. The stone signal is not sent until the disable signals have been acknowledged by the transitions that are disabled. It seems that the signals from the transition acknowledging the enable signals are in a race to claim the stone at the place, but this is not so because in the case of a conflict the conflict structure permits only one of the transitions to acknowledge the enable signal, and prevents others from acknowledging the enable signal until they are disabled.

In the transition structure, the IT-module sends an enable signal to the T-module only when it receives enable signals on all incident links, that is when all input places of the transition indicate that they have stones. On receiving an enable signal from the IT-module, the T-module sends a request to the constraint structure by sending an enable signal on link 1. If the output places of the transition do not take part in constraints, this link is terminated by a sink module, and the enable signal is immediately acknowledged, that is the transition immediately gets permission from the constraint structure. If the output places of the transition take part in constraints, the

constraint structure witholds acknowledgement of the enable signal until the output places of the transition can be admitted to the set of active places without violating the constraints, or until it receives a disable signal from the transition indicating that the transition is being disabled.

On receiving permission from the constraint structure, the T-module sends a request to the conflict structure by sending an enable signal on link 2. If conflicting transitions send requests to the conflict structure, the conflict structure gives permission to only one of them by acknowledging the enable signal; acknowledgement of the enable signals from other conflicting transitions is witheld until they are disabled. In granting permission to a transition the conflict structure blockades the transitions in conflict with the transition, i.e., requests from those transitions will be blocked if and when they are made. On receiving permission from the conflict structure, the T-module acknowledges the enable signal on link 0. This acknowledge signal reaches all incident places of the transition through the IT-module. Thus the transition that is given permission by the conflict structure is able to claim the stones at its input places as these acknowledgement signals are guaranteed to be the first to arrive at the places because the other transitions are blockaded in the conflict structure. When the stone at a place is claimed by a transition, the place structure sends disable signals to the other transitions for which the place is an input place in order to disable them. When all of there transitions are disabled, that is the transition structure and the parts of the constraint

and the conflict structures which are associated with these transitions are disabled, the place sends a stone signal to the transition which claimed the stone.

On receiving stone signals from all input places, the IT-module sends a stone signal to the T-module. The T-module then terminates the blockade set up in the conflict structure on its behalf by sending a disable signal, and initiates the transition by sending a ready signal on link 3. If the transition is an output transition, this signal goes to the external world, otherwise it is immediately acknowledged by a sink module. In case the transition is an output transition, the ready signal is acknowledged by the external world when the associated event has occurred. Upon receiving the acknowledge signal, the T-module terminates the transition by acknowledging the stone signal on link 0 and sending a ready signal to the ET-module on link 4. The ET-module then sends ready signals to the output places of the transition. These ready signals correspond to stones.

The ready signal, representing a stone, sent to a place by the transition structure is acknowledged when the stone is picked up by some other transition. The ET-module returns an acknowledge signal to the T-module when it receives acknowledge signals from all output places of the transition. The T-module then disables the part of constraint structure associated with it and the action is completed. It should be noticed that the conflict structure is used to prevent conflicting transitions from claiming stones when a transition is given permission to initiate, and the constraint structure keeps the constraints associa-

ted with the output places of the transition in force until all stones put out by the transition are picked by some other transitions. Moreover a transition which sends a request to the conflict structure either promptly gets permission or is promptly disabled by a conflicting transition Therefore a transition is never blocked indefinitely in the conflict structure. The constraint structure however block a transition for as long as its output places cannot be added to the set of active places without violating the constraints.

The details of operation of the constraint structure is given below. A transition (and an initialization module) requests permission from the constraint structure by sending an enable signal to the constraint structure. Upon receiving the request the constraint structure tries to bring into force the constraints associated with the output places of the transition, and answers the enable signal only when the constraints are brought into force unless the request is cancelled by a disable signal in the mean time. If the emergent link of the IR-module on which the link is incident is already in active condition (i.e. the enable signal has been acknowledged but a disable signal has not been received) the constraints are already in force, and the IR-module immediately acknowledges the enable signal. If the constraints are not in force, the IR-module sends an enable signal to the R-module unless it has already done so on account of some other incident link. The task of the R-module is to try to bring into force the associated constraints by sending appropriate signals to the constraint modules on the emergent mixed-triple link (figures 4.32 and 4.34).

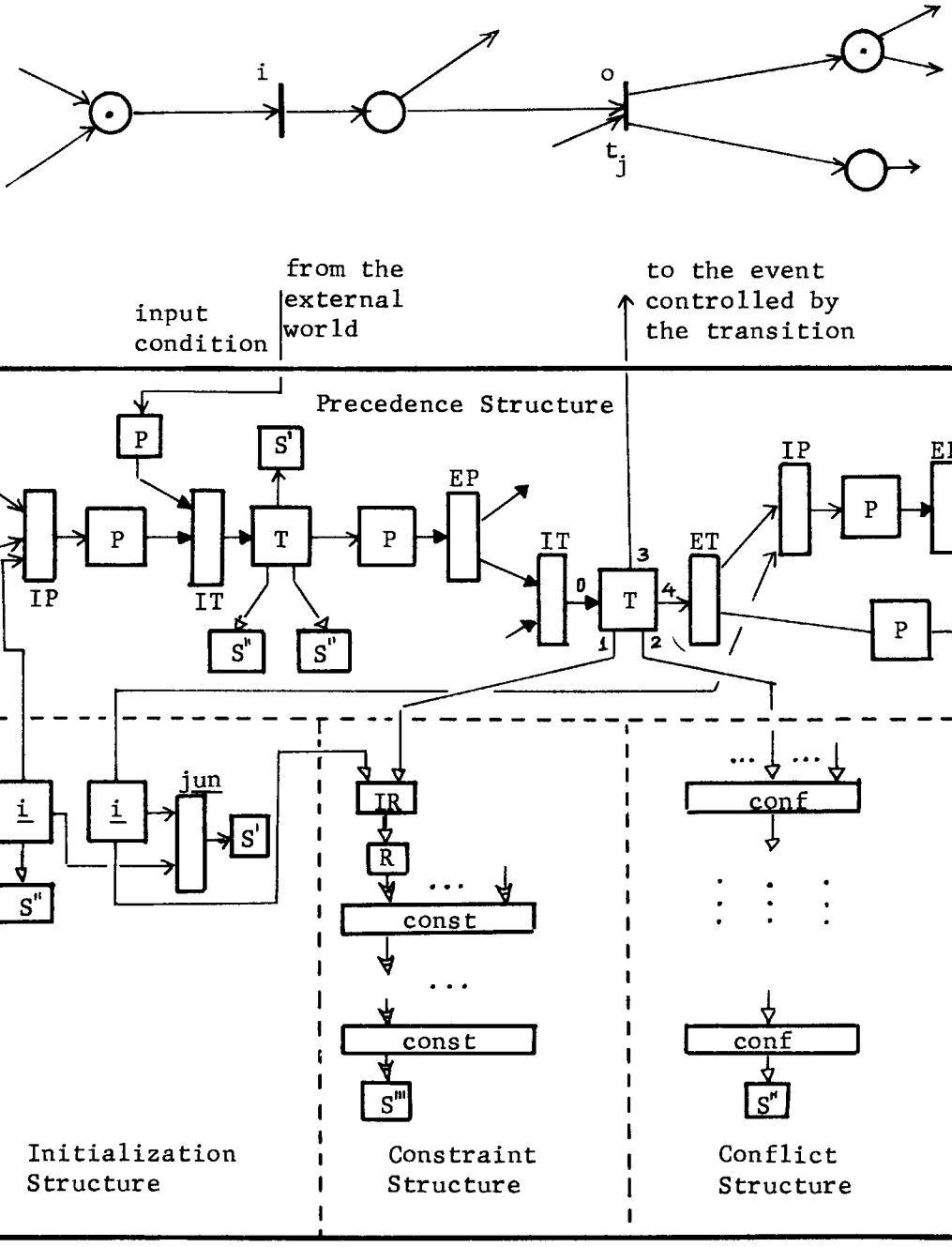


Figure 4.34 Coordination Structure

On receiving an enable signal, the R-module sends a check signal, a ready signal on the check sublink, on the mixed-triple link to determine if it should send an enable signal. A constraint module which is saturated, meaning all but one of whose mixed-triple links are active, blocks the check signal until it becomes unsaturated. When the check signal reaches the sink module it is acknowledged by a positive acknowledge signal. This signal travels back to the R-module, but if in passing through the conflict modules it encounters a saturated module, it is changed to a negative acknowledge signal. Thus the R-module may receive either a positive or a negative acknowledge signal in response to the check signal. In case of a negative acknowledge signal, the R-module sends a new check signal after completing the communication cycle, that is after completing a cycle on the disable sublink. In case of a positive acknowledge signal, the module sends an enable signal after completing the communication cycle. Since the condition of the constraint modules connected with the link is checked to be unsaturated just before the enable signal is sent, the enable signal is likely to reach the sink module unobstructed where it is acknowledged positively, but if the enable signal encounters a saturated constraint module, it is immediately acknowledged negatively by the module and the R-module has to start all over again by sending a check signal. In case the enable signal is acknowledged positively, the R-module acknowledges the enable signal on the incident link to inform the IR-module that the constraints have been brought into force. The IR-module then acknowledges the enable signals on the incident links which are waiting to be acknowledged.

On receiving a disable signal on an incident link, the IR-module immediately acknowledges the enable and disable signals on the link if the constraints are in force on account of the other incident link as well. If the other incident link is either idle or is in disabled condition, that is it does not need the constraints, the module sends a disable signal to the R-module unless it has done so already (on account of a disable signal on the other incident link). On receiving the disable signal, the R-module immediately proceeds to complete the communication cycle on the emergent mixed-triple link by sending a disable signal. The disable signal causes the constraint modules to immediately acknowledge the signals. The R-module then acknowledges the signals waiting to be acknowledged on the incident links on which disable signals were received.

Thus on receiving a disable signal on an incident link, the constraint structure relinquishes the claim of that link on the associated constraints, and when no incident link associated with an R-module has claim on the constraints, the constraints are lifted.

The conflict structure has a set of conflict module, one for each conflict cluster (figure 4.32). Thus transitions that mutually conflict have at least one conflict module in common. If a conflict module is idle and an enable signal is received on an incident link, the module allows the enable signal to pass through it and becomes engaged to that link. If more than one incident link sends enable signal to a conflict module at the same time, the link which engages the module is arbitrarily selected by the module. The module remains engaged to that link until

a disable signal is received on the link, at which time the module becomes free to be engaged once again. Enable signals sent to an engaged conflict module are blocked by the module. Thus in giving permission to a transition, the conflict structure blockades the conflicting transitions in that any requests from them are blocked. It should be recalled that a transition which is given permission by the conflict structure disables the conflicting transitions by claiming its input stones. Thus a transition which sends a request to the conflict structure either gets permission promptly or is promptly disabled by a conflicting transition.

The initialization structure first initiates the constraint structure by sending enable signals to the constraint structure. The constraint structure then brings into force the constraints associated with the places having stones initially and acknowledges the enable signals. As the initialization modules receive the acknowledge signals, they send ready signals to the junction module. The junction module acknowledges these signals when it receives signals from all initialization modules. Thus the junction module ensures that the initialization modules do not begin the action of placing stones in places before constraints associated with all places having initial stones are established in the constraint structure. On receiving acknowledge signals from the junction module, the initialization module sends stones to the places in the set B^0 by sending ready signals to the corresponding place structure. The simulation of the coordination structure begins as soon as stones are placed in places.

Simulation of the coordination net continues with initiations and terminations of transitons until a condition is reached when none of the idle transitions can be initiated until an input is received from the external world. In this case the simulation of the net awaits the input signal, and when the input signal is received, the simulation continues once again with initiations and terminations of transitions.

The next chapter gives informal proof of correctness of operation of coordination structures, and may be skipped without loss of continuity.

CHAPTER 5

CORRECTNESS OF COORDINATION STRUCTURES

5.1 Introduction

This chapter is devoted to prooving that the coordination structures implement the coordination nets correctly. Before the proofs are undertaken, a formalism for giving a precise meaning to correctness of coordination structures and for use in the proofs is introduced below.

5.2 Initiation-termination Histories of the Coordination Nets

Initiation and termination of transitions was explained earlier in Chapter 2 . Initiation and termination of input conditions and events is explained as follows. An input condition is said to initiate the moment the external world indicates to the net that the input condition has been attained, and the input condition is said to terminate when the net signals the external world that the input condition has been recorded. Initiation and termination of events correspond to their initiation and termination in the external world. Initiation-termination history of anyone of these entities, as for example a transition, is a sequence of initiations and terminations of that transition. The first element in the sequence is an initiation, and the initiations and terminations alternate. The initiations and terminations of a transition t_i are

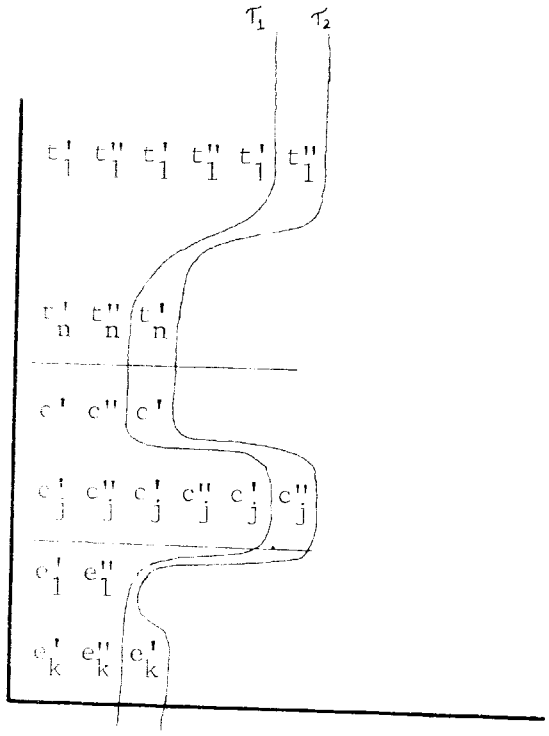


Figure 5.1 An Initiation-termination History H^T

denoted by t_i' and t_i'', that of an input condition c_j by c_j' and c_j'', and that of an event e_k by e_k' and e_k'' respectively. Thus the initiation-termination histories of a transition t_i are members of the set of strings $(t_i' t_i'')^* (t_i' + \lambda)$. An initiation-termination history of a net is an ordered set of strings denoting histories of the transitions, input conditions and events associated with the net (figure 5.1). A history is reached through a succession of time slices (figure 5.1). The history up to a time slice τ (i.e. at time τ) is denoted by H^{τ} . A slice τ_2 is said to be a successor to slice τ_1 if the individual strings in H^{τ_2} are equal to or longer than the corresponding strings in H^{τ_1} by at most one unit and there is at least one string in H^{τ_2} that is longer than the corresponding string in H^{τ_1} .

Terminology

AT^{τ} denotes the set of transitions active at time slice τ. Thus the set AT^{τ} consists of those transitions whose initiation-termination histories in H^{τ} are of the form $(t't'')^*t'$, and it is given by $AT(H^{\tau})$. B^{τ} denotes the set of places having stones at time slice τ. B^{τ} is also referred to as the stone distribution at time τ, and can be determined from C and H^{τ} in the following manner. The difference between the number of stones at a place at time τ and the initial condition is equal to the difference in the number of terminations of transitions for which that place is an output place and the number of terminations of transitions for which the place is an input place. From

C, the specification of the net, one knows which transitions have the place as an input and which have the place as an output place, and from H^τ one knows the number of initiations and terminations of these transitions until time slice τ. Thus B^τ is given by a function $B(C,H^\tau)$. B_c^τ is the set of places with stones whose stones have been claimed but have not been removed. In other words B_c^τ is the set of places which are input places of active transitions. Thus $B_c^\tau = I(t_1) \cup \cdots \cup I(t_k)$ where $AT^\tau = \{t_1 \ldots, t_k\}$. $B_u^\tau = B^\tau - B_c^\tau$ is the set of places with unclaimed stones. The sets B_c^τ and B_u^τ are denoted by $B_c(C,H^\tau)$ and $B_u(C,H^\tau)$ respectively.

An input condition is said to be active during the time interval between its initiation and termination. AC^τ represents the set of active input conditions. An active input condition is said to be in condition <u>active-α</u> prior to the termination of the input transition associated with it, and in condition <u>active-β</u> after the termination of that transition. AC_α^τ and AC_β^τ represent the set of input conditions which are active-α and active-β respectively. These sets are given by $AC_\alpha(H^\tau)$ and $AC_\beta(H^\tau)$. An input-condition is said to be inactive when it is not active. IC^τ represents the set of inactive input conditions. This set is given by the function $IC(H^\tau)$.

Definition 5.1 The set $(a_1,\ldots,a_n)$ is said to be an occurrence set if $a_1,\ldots,a_n$ are initiations and terminations of distinct transitions, input conditions and events. $H[x_1x_2\ldots x_n]$ where $x_1,\ldots,x_n$ are occurrence sets, represents the history $H^0 \cdot x_1 \cdot x_2 \cdot \ldots \cdot x_n$ where H^0 is the null history.

Definition 5.2 The feasible initiation-termination histories of a coordination net C are defined as follows:

1. A null history is a feasible history

   For a null history $B_u = B^0$, $IC = Ci$, $IE_\alpha = E$ and the other sets are empty.

2. If H^T is a feasible history then

   i) $H^T \cdot t_i'$ is a feasible history if

      (a) $I(t_i) \subset B_u^T$

      (b) $\mathcal{A}(AP^T \cup O(t_i), Ct)$ is true

      and

      (c) $\underline{ic}\ (t_i) \in AC_\alpha^T$ where interpretation $I = <\underline{ie}, \underline{ic}>$ (see section 2.6)

   ii) $H^T \cdot t_i''$ is a feasible history if

      (a) $t_i \in AT^T$

      and

      (b) $\underline{ie}\ (t_i) \in IE_\alpha^T$

   Changes in membership of there sets are given in table 5.1

   iii) $H^T \cdot C_j'$ is feasible if $C_j \in IC^T$

   iv) $H^T \cdot C_j''$ is feasible if $C_j \in AC_\beta^T$

   v) $H^T \cdot e_k'$ is feasible if $e_k \in IE_\beta^T$

   vi) $H^T \cdot e_k''$ is feasible if $e_k \in AE^T$

		B_u	B_c	AT	IC	AC_α	AC_β	IE_α	IE_β	AE
i	$.t_i'$	$-I(t_i)$	$I(t_i)$	t_i				$-\underline{ie}(t_i)$	$\underline{ie}(t_i)$	
ii	$.t_i''$	$O(t_i)$	$-I(t_i)$	$-t_i$		$-\underline{ic}(t_i)$	$\underline{ic}(t_i)$			
iii	$.c_j'$				$-c_j$	c_j				
iv	$.c_j''$				c_j		$-c_j$			
v	$.e_k'$								$-e_k$	e_k
vi	$.e_k''$							e_k		$-e_k$

Table 5.1 Changes in membership of the sets with initiations and terminations

Definition 5.3 If $H^{T'} = H^T \cdot x$ where H^T is a feasible history and $x = \{t'_i,\ldots,t'_j,t''_k,\ldots,t''_\ell,c'_m,\ldots,c'_n,c''_o,\ldots,c''_p,e'_q,\ldots,e'_r,e''_s,\ldots,e''_t\}$ is an occurrence set then $H^{T'}$ is adjacent to H^T if

i)

(a) $I(t_i),\ldots,I(t_j)$ are disjoint and $I(t_i) \cup \ldots \cup I(t_j) \subset B^T_u$

(b) $\not\Phi(AP^T \cup O(t_i) \cup \ldots \cup O(t_j), Ct)$ is true

(c) $\{\underline{ic}(t_i),\ldots,\underline{ic}(t_j)\} \subset AC^T_\alpha$

ii)

(a) $\{t_k,\ldots,t_\ell\} \subset AT^T$

(b) $\{\underline{ie}(t_k),\ldots,\underline{ie}(t_k)\} \subset IE^T_\alpha$

iii) $\{c_m,\ldots,c_n\} \subset IC^T$

iv) $\{c_o,\ldots,c_p\} \subset AC^T_\beta$

v) $\{e_q,\ldots,e_r\} \subset IE^T_\beta$

vi) $\{e_s,\ldots,e_t\} \subset AE^T$

Definition 5.4 An occurrence sequence $X = x_1\ x_2\ \ldots\ x_n$ is a sequence of occurrence sets.

Definition 5.5 The simulation sequences of a coordination net C are recursively defined as follows:

i) a null occurrence sequence is a simulation sequence

ii) if the occurrence sequence $X_{n-1} = x_1 x_2 \cdots x_{n-1}$ is a simulation sequence of a net and $H[X_n]$, where $X_n = X_{n-1}\, x_n$, is a feasible initiation-termination history of the net adjacent to the history $H[X_{n-1}]$ then X_n is a simulation sequence of the net.

Definition 5.6 A simulation sequence X of a coordination net C is said to be a complete sequence if the initiation-termination history $H[X]$ cannot be advanced by rules i and iv of definition 5.2; the simulation sequence is complete as regards initiations of transitions and terminations of input conditions. (A simulation sequence that is not complete is referred to as an incomplete simulation sequence.)

Definition 5.7 A simulation sequence X (and the associated initiation-termination history $H[X]$) of a coordination net C is said to be a terminal simulation sequence (history) if the initiation-termination history $H[X]$ cannot be advanced by rules i, ii and iv of definition 5.2; that is, the simulation sequence is complete as regards initiation of transition, termination of input condition and termination of transitions.

The simulation of a coordination net consists in advancing a simulation sequence for the net by advancing the initiation-termination history. Simulation of net is assumed to have the two properties described below.

Property i) The simulation is active; that is if a simulation sequence X^{τ} (τ refers to the time slice) is not a terminal sequence, then the simulation advances the simulation sequence by advancing the initiation-termination history.

Property ii) The simulation is attentive; that is if termination of transitions is suspended (i.e. rule ii in the definition 5.2 is suspended) following any simulation sequence X^{τ}, the simulation leads to a simulation sequence $X^{\tau'}$ that is complete (see definition 5.6).

The properties above essentially state that the simulator, the agent simulating the net, does not sit idle. A (hardware) structure implementing a coordination net that has these properties will be said to be deadlock free.

5.3 Signal Histories of the Coordination Strictures

It should be recalled that signals on simple links are drawn from the alphabet $\{r,a\}$, where r is a ready signal and a is an acknowledge signal, and that a ready signal is carried in the direction of the link while an acknowledge signal is carried in the opposite direction. Since the first signal on a link must be a ready signal and since the ready and acknowledge signals alternate, the sequences of signals on a simple link are members of the set of strings $(ra)^*(\lambda+r)$ where $*$ is the Kleene star. The sequence of signals on a link until time τ, that is time slice τ, is said to be the signal history of the link at time τ.

Similarly the signal history of a port is the sequence of signals at the port until time slice τ .

The signal history (or the total history) of a coordination structure at a time τ is an ordered set consisting of the signal histories at the head of the links incident on the structure, at the tail of the links at port 3 of the T-modules in the structure and at the head of the emergent links (figure 5.2). Signals on the incident links correspond to initiation and termination of input-conditions, signals at port 3 of the T-modules correspond to initiation and termination of transitions, and signals at the head of the emergent links correspond to initiation and termination of events. The signal history of a coordination net at time slice τ is denoted by $\underline{H}^{\tau}$.

The structure produces a signal history through a succession of time slices. The time slices are associated with eposhs, i.e. steps in the progress of time viewed in terms of the occurrence of the signals. A time slice τ_j is said to be a successor (i.e. an immediate successor) of a time slice τ_i if between the time slice τ_i and the time slice τ_j there is an occurrence of at most one signal on each link (each row in the history - see figure 5.2). The signals between the two successive slices are coincident. If the signals are subscripted to indicate which transitions, input-conditions or events they are associated with, e.g., r_{t_i} indicates that the ready signal is

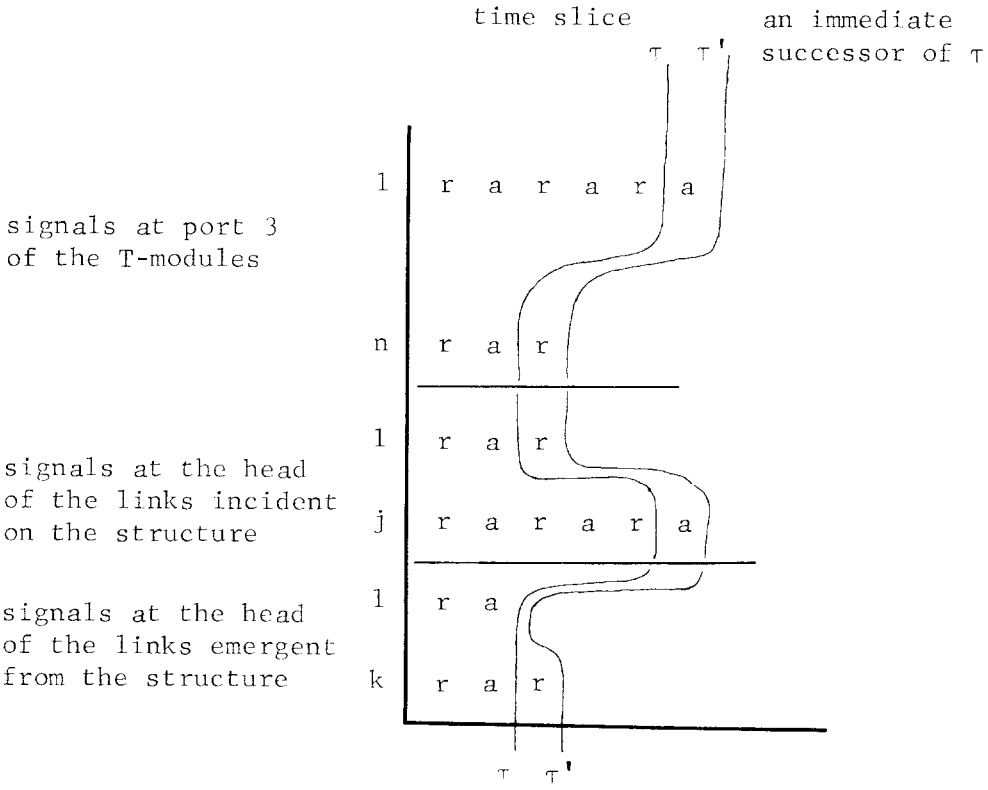


Figure 5.2 Total Signal History $\underline{H}^{\tau}$

associated with transition t_i, signal occurrence set can be defined as a set of signals associated with distinct entities (transitions, input-conditions and events). A signal occurrence sequence is a sequence of signal occurrence sets (see definitions 5.1 and 5.4).

If $\sigma = \tau(1)\,\tau(2)\,\ldots\tau$ is a sequence of successive slices (where slice $\tau(1)$ is a suceessor to the initial slice $\tau(0)$) for the signal history $\underline{H}^{\tau}$, then the signal occurrence sequence $\underline{X} = x_1 x_2 \cdots x_n$ where x_j is the set of occurrences of signals between slices $\tau(j-1)$ and $\tau(j)$, is said to correspond to the signal history $\underline{H}^{\tau}$ for the sequence of successive slices σ, and is denoted by $\underline{X}(\underline{H}^{\tau},\sigma)$

If a structure produces signal history $\underline{H}^{\tau}$ through a succession of time slices $\sigma = \tau(1)\,\tau(2)\,\ldots\tau$, then the signal occurrence sequence $\underline{X}(\underline{H}^{\tau},\sigma)$ is said to be a simulation sequence of the structure.

5.4 Representative Histories and Occurrence Sequences

A representative initiation-termination history for a signal history is obtained by substituting initiation and terminations of transitions, input-conditions and events for the signals in the history in the following manner: i) in the signal history corresponding to port 3 of the T-module associated with transition t_i, ready signals are replaced

by t'_i and acknowledge signals by t''_i, ii) in the signal history corresponding to incident link j, ready signals are replaced by c'_j and acknowledge signals by c''_j, and iii) in the signal history corresponding to an emergent link k, ready signals are replaced by e'_k and acknowledge signals by e''_k. Figure 5.3 shows the representative initiation-termination history corresponding to the signal history of figure 5.2. The representative history of a signal history $\underline{H}^{\tau}$ is denoted by $H(\underline{H}^{\tau})$.

In a similar manner a representative occurrence sequence for a signal occurrence sequence is obtained by substituting initiaion and termination of associated transitions, input-conditions and events for the signals in the occurrence sets of the signal occurrence sequence; a signal r_{t_i} is substituted by t'_i. The representative occurrence sequence for a signal occurrence sequence $\underline{X}$ is denoted by $X(\underline{X})$. Similarly the representative occurrence sequence for a signal history $\underline{H}^{\tau}$ reached through a sequence of time slices $\sigma = \tau(1)\ \tau(2)\ \ldots\tau$, which is $X(\underline{X}(\underline{H}^{\tau},\sigma))$, is denoted by $X(\underline{H}^{\tau},\sigma)$.

A simulation sequence X of a coordination net C and the simulation sequence $\underline{X}$ of the coordination structure that implements the net, correspond to each other if X and $X(\underline{X})$ are the same.

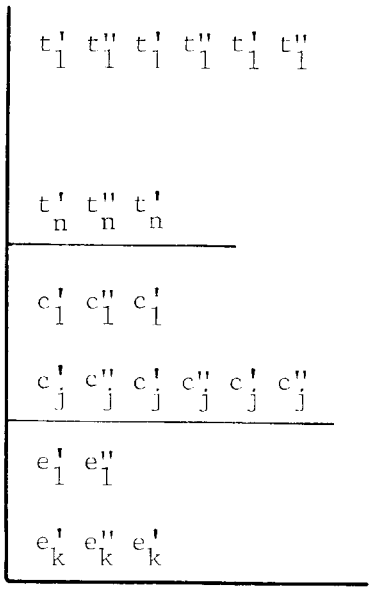


Figure 5.3 The Representative Initiation-termination History Corresponding to the Signal History of Figure 5.2

A simulation sequence $\underline{X}$ of a coordination structure is said to be a terminal simulation sequence if $X(\underline{X})$ is a terminal simulation sequence of the net that the structure implements; $\underline{X}$ is said to be complete if $X(\underline{X})$ is complete.

5.5 Statement of Correctness of Coordination Structures

The coordination structures can be said to implement the coordination nets correctly if the theorems given below hold. In theorems 2 and 3, the adjective 'promptly' means without undue delays; finite delays, as encountered in transmission of signals, are permitted but indefinite delays, as in the case the action has to await some input signal, are not permitted.

Theorem 1 Each simulation sequence $\underline{X}$ of a coordination structure S that implements a coordination net C corresponds to some simulation sequence of the net.

Theorem 2 The coordination structures are active; that is, if the simulation sequence $\underline{X}^{\tau}$ produced by a coordination structure until time τ is not a terminal simulation sequence, the structure (promptly) advances the simulation sequence (by producing a signal that advances the signal history).

Theorem 3 The coordination structures are attentive; that is, if after a coordination structure has produced a simulation sequence $\underline{X}$, termination of transitions is suspended, the simulation (promptly)

reaches a simulation sequence that is complete.

Theorem 4 Each simulation sequence of a coordination net C corresponds to some simulation sequence of the coordination structure S that implements the net. (This theorem is converse of theorem 1 above.)

Theorem 1 above merely states that the coordination structure that implements a coordination net does not produce any incorrect signal; this does not preclude the structure going into a deadlock and coming to a stop even when the simulation is not a terminal simulation. Theorem 2 states that the structures are active and theorem 3 states that the structures are attentive; structures having these two properties can be called deadlock free. Theorem 4 states that the set of simulation sequences for the structures does not exclude any simulation sequence of the net. Outline of the proofs of these theorems are presented before the proofs themselves in order to facilitate understanding of the proofs. The overview provided by these outlines should aid understanding of the role of various lemmas proved on the way.

5.6 Proof of Theorem 1

Theorem 1 can be restated as follows: If the coordination structure S that implements a coordination net C produces a signal history $\underline{H}^{\tau}$ through a sequence of successive time slices $\sigma = \tau(1)\ \tau(2) \ldots \tau(n) = \tau$ then for $1 \leq j \leq n$ $H(\underline{H}^{\tau(j)})$ is feasible initiation-

termination history of the net and is adjacent to the history $H(\underline{H}^{\tau(j-1)})$. (For definitions of feasible history and adjacency of histories see definitions 5.2 and 5.3.)

Outline of the Proof

1. The transitions which are initiated or terminated between adjacent time slices of a signal history are disjoint in that they do not have any input places in common.

2. The stone distribution (in coordination net C) corresponding to the feasible initiation-termination history $H(\underline{H}^{\tau(j-1)})$ has unclaimed stones in all the input places of transitions initiated between time slices $\tau(j-1)$ and $\tau(j)$.

3. $AP^{\tau(j-1)} \cup O(T'^{\tau(j-1)-\tau(j)})$ is admissible, where $T'^{\tau(j-1)-\tau(j)}$ is the set of transitions initiated between time slice $\tau(j-1)$ and $\tau(j)$ and the function $O(T')$ has as its value the output places of the transitions in the set T'.

4. An input transition is initiated only if it has received an input from the external world.

Steps 1, 2, 3 and 4 show that the occurrence set satisfies condition (i) in definition 5.3.

5. Transitions terminated between time slices $\tau(j-1)$ and $\tau(j)$ are active at time $\tau(j-1)$, and the terminations come only after the associated events, if any, in the external worlds have occurred. This satisfies condition (ii) of definition 5.3.

6. An input-condition is initiated only if it is inactive, and an input-condition is acknowledged only after the associated input transition has occurred.

7. An event is initiated only after the associated transition has initiated, and the transition is terminated only after the event has occurred.

Steps 6 and 7 show that the occurrence set satisfies conditions (iii), (iv), (v) and (vi) of definition 5.3.

It should be recalled that a stone cycle on a triple link consists of a cycle on the enable sublink and a cycle on the stone sublink, that is, it involves signals e.r, e.a, s.r and s.a on the link (figure 5.4). The link is said to be stone-active during the time interval between signals s.r and s.a, in this condition the stone sublink of the link is active.

Sublemma 5.1 The triple link at port 0 of a T-module is stone-active when the simple link at port 3 of the module is active; this includes the moment when the link is turned active and the moment the link is turned inactive. (A simple link is said to be active during the interval between a ready signal and the associated acknowledge signal.)

Proof: From the P-net specification of the T-module (figure 4.16) it can be seen that the signals 0e.r, 0e.a, 0s.r, 3.r, 3.a and 0s.a are ordered in this sequence. Since signals 3.r and 3.a fall in between signals 0s.r and 0s.a, the sublemma holds.

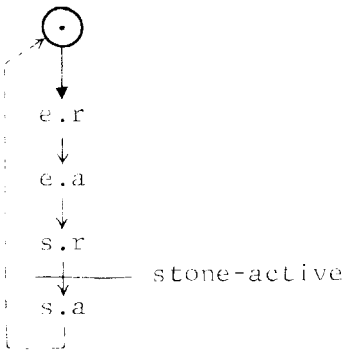


Figure 5.4 The Stone Cycle and the Stone-active Condition

Sublemma 5.2 If the emergent link of an IT-module is stone-active then the incident links are also stone-active.

Proof: The relationships among relevant signals are shown in figure 5.5. This figure shows that the sublemma holds.

■

Sublemma 5.3 Only one emergent link of an EP-module can be stone-active at a time, and when an emergent link is stone-active, the incident link is also stone-active.

Proof: The inter-relationship between relevant signals as obtained from the P-net specification of the EP-module is shown in figure 5.6. Because of the conflict at place p, only one of the emergent links can be stone-active at a time. Moreover, if any of the emergent links is stone-active then the incident link is also stone-active.

■

Lemma 5.1 The transitions which are initiated or terminated between adjacent time slices $\tau(j-1)$ and $\tau(j)$ are mutually disjoint from the transitions which continue to be active through time slices $\tau(j-1)$ and $\tau(j)$. (Two transitions are disjoint when they do not have any input places in common).

Proof: As the slices are adjacent, the initiations and terminations are coincident. That is, the initiations of transitions and terminations of other transitions occur at the same moment. From sublemmas 5.1 and 5.2, all triple links between the place structures and

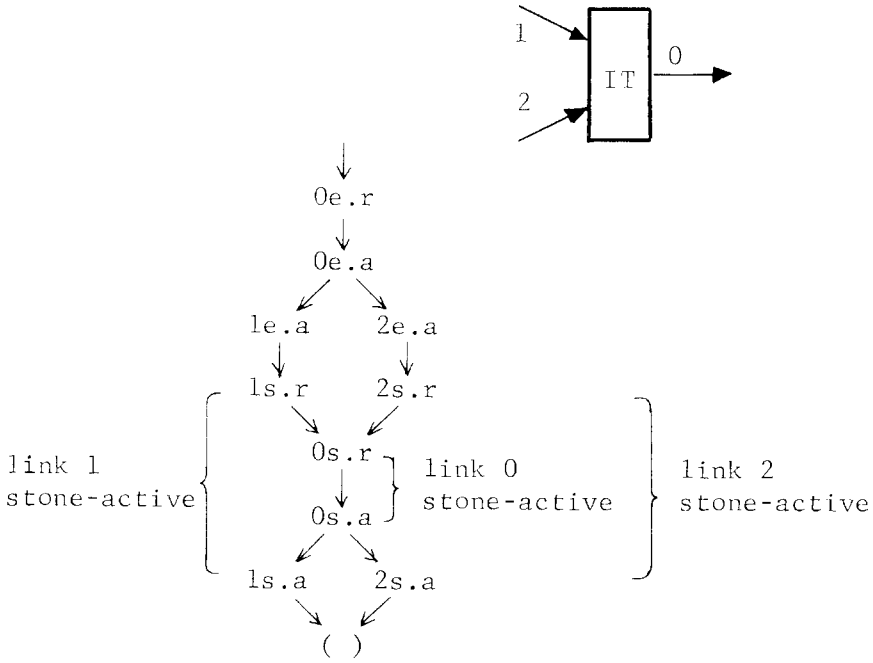


Figure 5.5 Stone-active Conditions and the IT-module

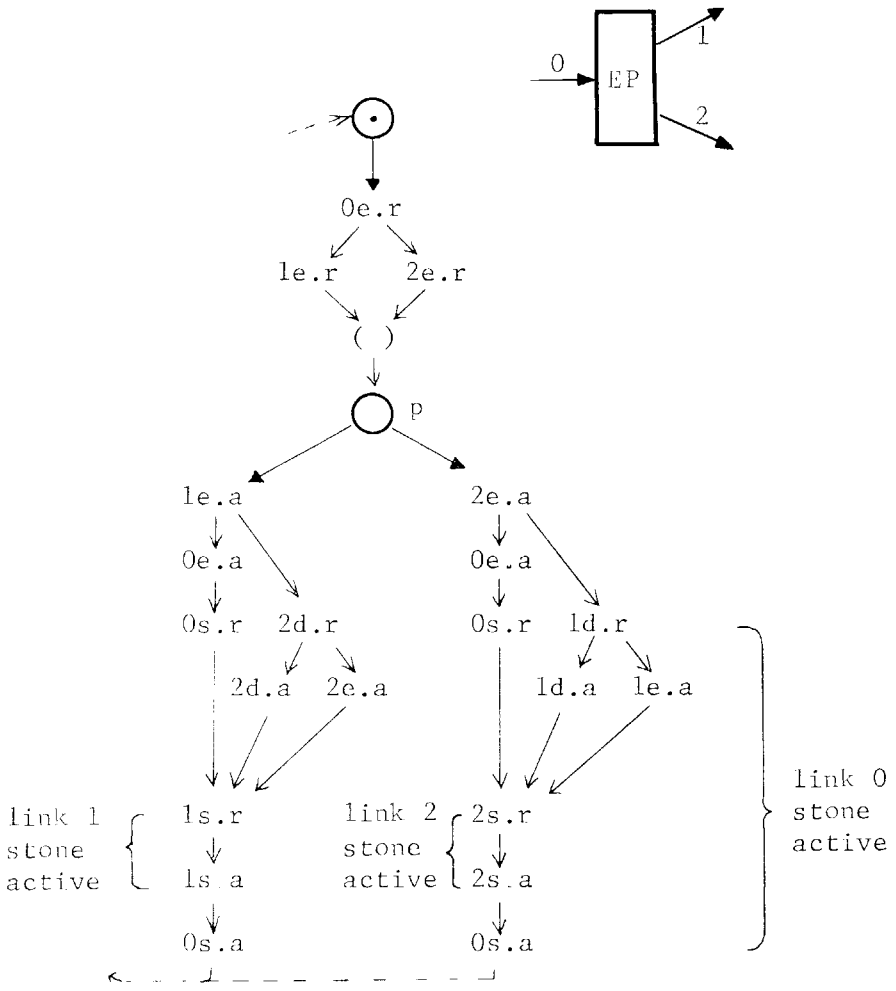


Figure 5.6 Stone-active Conditions and the EP-module

transitions which initiate or terminate between time slice $\tau(j-1)$ and $\tau(j)$ as well as transitions which continue to be active through the time slices $\tau(j-1)$ and $\tau(j)$ are stone-active at this moment. Therefore, if any of these transitions share an input place, at least two emergent links of an EP-module in the place structure will be stone active at this moment, but this is not possible as that contradicts sublemma 5.3 . Therefore these transitions do not have any input places in common, i.e., they are disjoint.

■

The following sublemmas show that the input places of the transitions initiated by the structure between time slices $\tau(j-1)$ and $\tau(j)$ have unclaimed stones in them.

Sublemma 5.4 One stone cycle on an emergent link of a place structure corresponds to one communication cycle on an incident link of the place structure, i.e., a definite communication cycle on an incident link is associated with a definite stone cycle on an emergent link.

Proof: The diagram of a place structure is shown in figure 4.28 . From the P-net specification of the modules it can be observed that: (i) One communication cycle on the emergent link of an IP-module corresponds to one communication cycle on an incident link and vice versa, (ii) one stone cycle on the emergent link of a P-module corresponds to one communication cycle on the incident link and vice versa, and (iii) one stone cycle on an emergent link of an EP-module

corresponds to one stone cycle on the incident link of the module and vice versa.

From these three observations it can be concluded that one stone cycle on an emergent link of a place structure corresponds to one communication cycle on an incident link.

■

The following sublemma is similar to that proved above except that it deals with the transition structure instead of the place structure.

Sublemma 5.5 The occurrence of a transition, that is an initiation and termination of the transition uses a stone cycle from each incident link of the transition structure, and starts a communication cycle on each of the emergent links of the structure.

Proof. The P-net specification of the T-module (figure 4.16) shows that the sequence 0e.r, 0e.a, 0s.r and 0s.a of signals is associated with the sequence 3.r, 3.a. That is, a stone cycle on link 0 is associated with a communication cycle on link 3. Furthermore one communication cycle on link 4 is associated with a communication cycle on link 3.

The P-net specification of the IT-module (figure 4.15) shows that a stone cycle on the emergent link is associated with a stone cycle on each of the incident links.

From the above paragraphs it can be seen that a communication cycle on link 3 of the T-module in the transition structure (figure 4.29) is associated with a stone cycle on each of the links incident on the structure.

The P-net of the ET-module shows that one communication cycle is undergone on each of the emergent links for each communication cycle on the incident link. From this fact and from the first paragraph above, it can be concluded that a communication cycle on link 3 of the T-module in the transition structure gives rise to a communication cycle on each of the emergent links. The sublemma is thus proved, for a communication cycle on link 3 of the T-module corresponds to an initiation and termination of the associated transition.

■

From the above sublemmas it can be seen that a transition in the coordination structure needs a stone cycle from each input place in the structure in order to occur, and a place structure gives rise to only one stone cycle for each communication cycle it receives from either some transition or from the initialization structure. Furthermore, in the process of termination a transition gives rise to a communication cycle on each output place of that transition. The place structure thus merely transmits such cycles from one transition to another, just as a place in the coordination net merely transmits a stone from one transition to another.

As the presence of a stone at a place in a net is important as regards which transition in the net can be initiated, a similar concept should be defined in connection with places (i.e. place structures) in the coordination structure. It has been said before that a transition, in terminating, gives rise to a cycle to the output places and the cycle given to a place is, so to say, used by a transition in the process of initiation and termination. From this arises the notion of a place being on. A place in a structure is said to be on during the time interval from the time a transition for which the place is an output place terminates to the time a transition for which the place is an input place terminates. Thus, for a place the condition of being "on" in the structure corresponds to the condition of "having a stone" in the net.

Lemma 5.2 If a structure produces a signal history $\underline{H}^T$, and $H(\underline{H}^T)$, the corresponding representative initiation-termination history, is feasible, then exactly those places which have stones in them at the end of a simulation of the net producing the initiation-termination history $H(\underline{H}^T)$ are on in the structure.

Proof: A place in the net has a stone at the end of the above simulation iff the number of stones placed in it (counting the initial stone) exceeds by one the number of stones removed from the place †. The number of stones placed in a place is equal to one (for the initial stone, if any) plus the number of terminations of transitions

for which the place is an output place, and the number of stones removed from a place is equal to the number of terminations of transitions for which the place is an input place.

In an anlogous way, a place in the structure is on iff the number of cycles given to it, counting the cycle given by the initialization structure (if any), is one more than the number of cycles used by transitions for which the place is an input place. From the construction of the coordination structure and the P-net specification of the initialization module, the initialization structure gives a cycle to a place structure corresponding to a place in the net iff the place in the net has an initial stone. The number of cycles given by the transitions is equal to the number of terminations of the transitions for which the place is an output place, and the number of cycles used by the transitions is equal to the number of terminations of transitions for which the place is an input place.

Since $\underline{H}^T$ and $H(\underline{H}^T)$ are isomorphic, it is clear from the above arguments that the places which are on in the structure correspond one-to-one with the places which have stones in the net.

■

---

† Note that, as the nets are assumed to be safe, a place is not occupied by more than one stone at a time.

Lemma 5.3 If a coordination structure produces a signal history $\underline{H}^{\tau}$ through a sequence of time slices $\sigma = \tau(1)\ \tau(2)\ \ldots\tau(j)$ and $H(\underline{H}^{\tau(j-1)})$ is feasible, then the input places of the transitions initiated between $\tau(j-1)$ and $\tau(j)$ have unclaimed stones at the end of $H(\underline{H}^{\tau(j-1)})$. That is, $I(T'^{\tau(j-1)-\tau(j)}) \subset B_u(C, H(\underline{H}^{\tau(j-1)}))$ where $T'^{\tau(j-1)-\tau(j)}$ is the set of transitions initiated between the time slices $\tau(j-1)$ and $\tau(j)$, and $I(T')$ represents the input places of the set of transitions T'.

Proof: First of all it will be shown that $I(T'^{\tau(j-1)-\tau(j)}) \subset B(C, H(\underline{H}^{\tau(j-1)}))$. The input places of transitions in $T'^{\tau(j-1)-\tau(j)}$ must be on at the end of $\tau(j-1)$, because these transitions cannot initiate unless the corresponding places are on and because between the end of time slice $\tau(j-1)$ and initiation of these transitions there can be no terminations of transitions that make these places on as $\tau(j-1)$ and $\tau(j)$ are successive time slices. Therefore, from lemma 5.2 these input places have stones, i.e.,

$$I(T'^{\tau(j-1)-\tau(j)}) \subset B(C, H(\underline{H}^{\tau(j-1)-\tau(j)})) \qquad \text{(i)}$$

From lemma 5.1, transitions which initiate between $\tau(j-1)$ and $\tau(j)$ are disjoint from the active transitions at $\tau(j-1)$. Therefore

$$I(AT^{\tau(j-1)}) \cap I(T'^{\tau(j-1)-\tau(j)}) = \Phi$$

which is the same as

$$B_c^{T(j-1)} \cap I(T'^{T(j-1)-T(j)}) = \Phi \qquad \text{(ii)}$$

From (i) and (ii)

$$I(T'^{T(j-1)-T(j)}) \subset B_u(C, H(\underline{H}^{T(j-1)}))$$

because $B_u^T = B^T - B_c^T$

■

The lemmas which follow are devoted to proving that the set of active places $AP^{T(j-1)} \cup O(T'^{T(j-1)-T(j)})$ is admissible, i.e., it does not violate the constraints.

The different conditions of double and mixed-triple links are explained in figures 5.7 and 5.8.

Sublemma 5.6 If the link at port 3 of the T-module is active, then the link at port 1 of the module is active.

Proof: The relationships between relevant signals are shown in figure 5.9 (the P-net specification for the T-module from which this is obtained is given in figure 4.16) which shows that link 1 is active when link 3 is active.

■

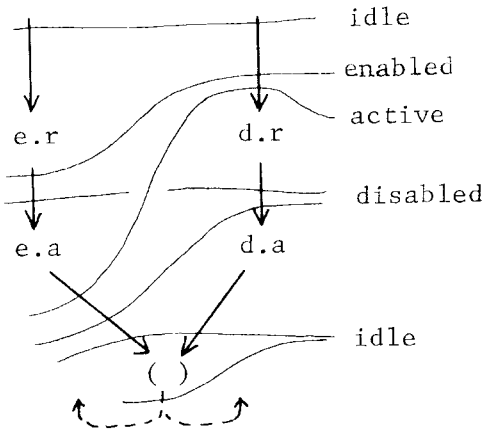


Figure 5.7 Conditions of the Double link

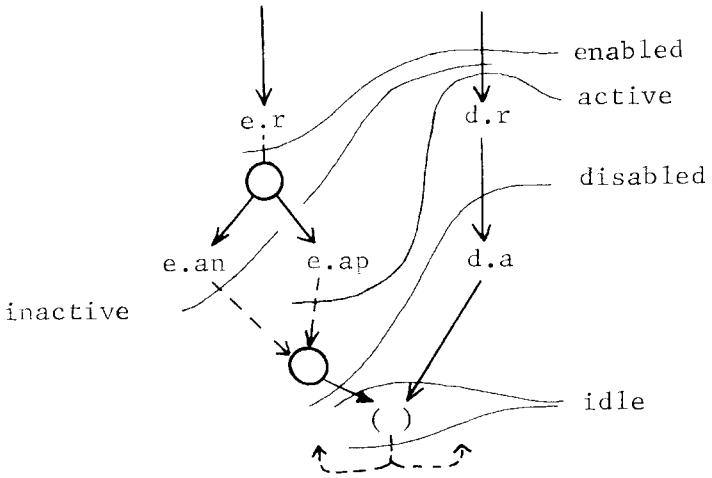


Figure 5.8 Conditions of the Mixed-tripple link

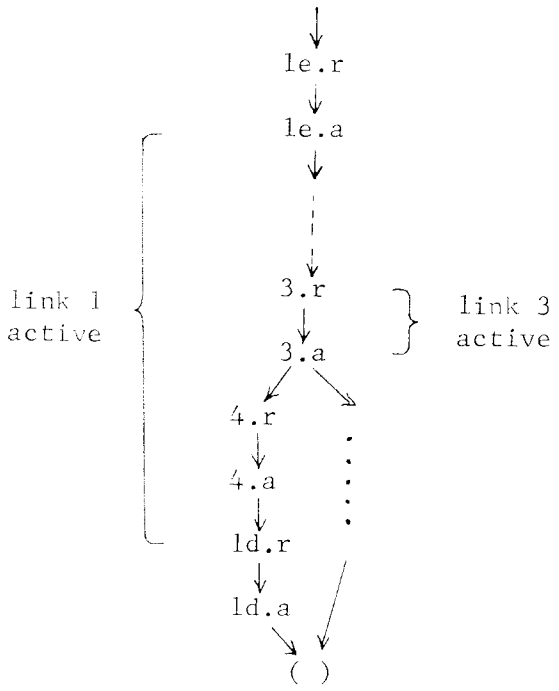


Figure 5.9 Relationships between active condition of the links at ports 1 and 3 of the T-module

A similar examination of the P-net specification of the IR, R and const modules shows that (i) the emergent link of the IR-module is active if any of the incident links is active, (ii) the emergent mixed-triple link of the R module is active if the incident link is active, and (iii) a mixed-triple link coming from a const-module is active if the corresponding incident link is active. Taken together these results can be stated as the following sublemma.

Sublemma 5.7 If link 1 of a T-module in the coordination structure is active, the entire mixed-triple link (in the constraint structure) associated with the T-module is active.

One of the important functions of the constraint module is to ensure that not all of the incident links of the module are active at the same time. It can be seen that this is in fact achieved as when an incident link is active, place p_k in the P-net specification of the module (shown in figure 4.23) has a stone and not all of the places $p_1,\ldots,p_n$ can have stones at the same time. The result that not all incident links of a constraint module can be active at the same time is stated below as a sublemma.

Sublemma 5.8 Not all incident links of a constraint-module can be active at the same time.

Sublemma 5.9 If the mixed-triple links associated with constraint equivalence classes $P_1,\ldots,P_k$ (of places) are active then the active place set given by $P_1 \cup \ldots \cup P_k$ is admissible i.e. $\mathcal{A}(P_1 \cup \ldots \cup P_k)$ is true.

Proof:

If $P_1 \cup \ldots \cup P_k$ were not admissible then there would be a subset $\{p_1,\ldots,p_i\}$ of $P_1 \cup \ldots \cup P_k$ that is a member of the constraint set Ct. Since $\{p_1,\ldots,p_i\}$ is a member of Ct, $\{P_1,\ldots,P_i\}$ is a member of RP(Ct) where $p_1,\ldots,p_i$ fall into the constraint equivalence classes $P_1,\ldots,P_i$ (from the definition of RP(Ct), section 3.2 of Chapter 3). From the construction of the constraint structure, since $\{P_1,\ldots,P_i\}$ is a member of RP(Ct), there is a constraint module in the constraint structure which involves just the links associated with these equivalence classes. Because of this constraint module the links associated with the equivalence classes $P_1,\ldots,P_i$ cannot be active at the same time (sublemma 5.8).
This is a contradiction as the links associated with $P_1,\ldots,P_i$ are active since the links associated with $P_1,\ldots,P_k$ are given to be active.

■

Sublemma 5.10 The links associated with the constraint equivalence classes into which the places in $AP^{\tau(j-1)}$ fall are active at the end of $\tau(j-1)$.

Proof:

$$AP^{\tau(j-1)} = B^{\tau(j-1)} \cup O(AT^{\tau(j-1)})$$

The links associated with the equivalence classes into which the places in $O(AT^{\tau(j-1)})$ fall are enabled because link 3 of each T-module associated with the transitions in $AT^{\tau(j-1)}$ is active. From sublemmas 5.6 and 5.7, when link 3 of a T-module is active, the link associated with the constraint equivalence class of places into which the output places of the transition fall is active (note that the net being homogeneous the output places of a transition fall into the same constraint equivalence class). (i)

The active condition of links emerging out of the places having stones (i.e. $B^{\tau(j-1)}$) is considered below.

From lemma 5.2 the places which have stones in the net are on in the structure. Examination of the precedence structure shows that when a place is on, either link 4 of the T-module of a transition structure or link 2 of the initialization module (as the case may be) that turned the place on is active. In the first case link 1 of the T-module is active since link 4 is active, and because link 1 is active,

the mixed-triple link associated with the transition is active in the constraint structure (sublemma 5.7). That is, the link associated with the constraint equivalence class of places into which the place falls is active. (Note that in a homogeneous net the output places of a transition belong to the same constraint equivalence class.) In the latter case, link 0 of the initialization module is active and therefore the mixed-triple link associated with the constraint equivalence class of places into which the place falls is active in the constraint structure.

One can say, therefore, that the links associated with the equivalence classes into which the places in $B^{\tau(j-1)}$ fall are active. (2)

From (1) and (2) the result in the sublemma follows.

■

Sublemma 5.11 The links associated with the constraint equivalence classes into which the places in $AP^{\tau(j-1)} \cup O(T'^{\tau(j-1)-\tau(j)})$ fall are active at the moment the transitions in $T'^{\tau(j-1)-\tau(j)}$ initiate (coincidentally).

Proof: From the P-net specification of the T-module it can be seen that link 1 of the module is active at the time link 3 is turned active. Therefore, from sublemma 5.7, the mixed-triple links in the constraint structure associated with these transitions are active, i.e. the links associated with the constraint equivalence classes into which the output places of the transitions in $T'^{\tau(j-1)-\tau(j)}$ fall are active.

Moreover the links associated with the constraint equivalence classes into which the places in $AP^{\tau(j-1)}$ fall, continue to be active until the effect of the termination of transitions in $T''^{\tau(j-1)-\tau(j)}$ is felt by them). The sublemma thus follows.

■

Lemma 5.4 If a structure produces a signal history $\underline{H}^{\tau}$ through time slices $\tau(1)$, $\cdots$, $\tau(j)$ and $H(\underline{H}^{\tau(j-1)})$ is feasible then the active place set $AP^{\tau(j-1)} \cup O(T'^{\tau(j-1)-\tau(j)})$ is admissible, i.e., $\mathcal{A}(AP^{\tau(j-1)} \cup O(T'^{\tau(j-1)-\tau(j)}))$ is true.

Proof. Let the places in $AP^{\tau(j-1)} \cup O(T'^{\tau(j-1)-\tau(j)})$ fall into the constraint equivalent classes P_1, $\cdots$, P_k. Then, according to sublemma 5.11, the links associated with these equivalence classes are active in the constraint structure. Therefore, from sublemma 5.9, the set of active places $P_1 \cup \cdots \cup P_k$ is admissible. Since $AP^{\tau(j-1)} \cup O(T'^{\tau(j-1)-\tau(j)}) \subseteq P_1 \cup \cdots \cup p_k$, and $p_1 \cup \ldots \cup p_k$ is admissible, the set of active places $AP^{\tau(j-1)} \cup O(T'^{\tau(j-1)-\tau(j)})$ is admissible.

■

Lemma 5.5 $T''^{\tau(j-1)-\tau(j)}$, the transitions terminated between adjacent time slices $\tau(j-1)$ and $\tau(j)$ are in $AT^{\tau(j-1)}$, and $\underline{ie}(T''^{\tau(j-1)-\tau(j)}) \subset IE_{\alpha}^{\tau(j-1)}$ is true (i.e. the events associated with the transitions have occurred).

Proof. Since the first signal on a simple link has to be a ready signal, and the ready and acknowledge signals alternate, a transition in the structure can be terminated only if it has been initiated and the associated event has occurred. --- (1)

As $H(\underline{H}^{\tau(j-1)})$ is isomorphic to $\underline{H}^{\tau(j-1)}$, the set of transitions active in the structure at $\tau(j-1)$ is the same as the set of transitions active in the net at $\tau(j-1)$. --- (2)

From (1) and (2) the transitions terminated (coincidentally) between time slices $\tau(j-1)$ and $\tau(j)$ are in $AT^{\tau(j-1)}$, and the events associated with them have occurred.

■

Lemma 5.6 The input-conditions which are initiated between adjacent time slices $\tau(j-1)$ and $\tau(j)$ are inactive at time $\tau(j-1)$, that is $C'^{\tau(j-1)-\tau(j)} \subset IC^{\tau(j-1)}$.

The proof of the lemma above is similar to the proof of lemma 5.5.

Lemma 5.7 The input-conditions which are terminated between adjacent time slices $\tau(j-1)$ and $\tau(j)$ are in $AC_{\beta}^{\tau(j-1)}$.

Proof. The termination of an input-condition in case of the coordination structure is marked by an acknowledge signal (on the associated incident link) which indicates that the associated input transition has terminated (and has registered (used) the input condition). Since the termination of the transition and the termination of input-condition are ordered that way, and the input-condition terminates between adjacent time slices $\tau(j-1)$ and $\tau(j)$, the termination of the transition must have occurred before $\tau(j-1)$. Now, in the coordination net, the corresponding termination of the transition removes the input-condition from the set AC_{α} and places it in the set AC_{β}. Therefore, the input-conditions which are (coincidentally) terminated between time slices $\tau(j-1)$ and $\tau(j)$ are in $AC_{\beta}^{\tau(j-1)}$.

■

Lemma 5.8 $E'^{\tau(j-1)-\tau(j)} \subset IE_{\beta}^{\tau(j-1)}$ and $E''^{\tau(j-1)-\tau(j)} \subset AE^{\tau}$.

Proof: This lemma holds because the links emerging from the coordination structure produce a ready signal at the head of the link (that is in the external world) only in response to a ready signal received at the tail, because they produce an acknowledge signal at the tail only in response to an acknowledge signal received at the head, and because the external world acknowledges the ready signal only after the associated event has occurred.

■

Theorem 1 Each simulation sequence $\underline{X}$ of the coordination structure S that implements a coordination net C corresponds to some simulation sequence of the net.

Proof. The theorem can be restated as follows: If the coordination structure S that implements a coordination net C produces a signal history $\underline{H}^{\tau}$ through a sequence of successive time slices $\sigma = \tau(1)\ \tau(2)\ \ldots\ \tau(n) = \tau$ then for $1 \leq j \leq n$ $H(\underline{H}^{\tau(j)})$ is feasible initiation-termination history of the net and is adjacent to the history $H(\underline{H}^{\tau(j-1)})$.

Basis

If σ is null this is true trivially.

Induction

The theorem is assumed to hold for the length of sequence σ less than or equal to $j - 1$, and is shown to hold for σ of length j.

To prove the induction step holds what needs to be shown is that $H(\underline{H}^{T(j)})$ is feasible and adjacent to $H(\underline{H}^{T(j-1)})$. As $H(\underline{H}^{T(j-1)})$ is feasible (by hypothesis) it is only necessary to show that $H(\underline{H}^{T(j)})$ is adjacent to $H(\underline{H}^{T(j-1)})$.

$H(\underline{H}^{T(j)})$ is adjacent to $H(\underline{H}^{T(j-1)})$ because from lemmas 5.1 to 5.8, $H(\underline{H}^{T(j)})$ meets the requirements for being adjacent to $H(\underline{H}^{T(j-1)})$ specified in the definition of adjacency (see definition 5.3).

The theorem thus follows.

■

5.7 Proof of Theorem 2

Theorem 2 states that coordination structures are active, that is, if the signal history produced by a coordination structure up to time slice τ is not a terminal history, then the structure (promptly) produces some signal which advances the signal history (as well as the time slice). In other words the structure continues to produce signals until a terminal signal history is produced, at which point the structure suspends its activity until arrival of a signal (a ready corresponding to an input-condition or an acknowledgement of an event) that makes the resulting signal history non-terminal. The steps in the proof of this theorem are as follows.

Outline of the Proof

1. A transition that is permitted to proceed by the conflict structure disables the conflicting transitions and initiations.

2. A transition that sends a request to the conflict structure for permission to proceed either gets permission promptly or is promptly disabled by some conflicting transition. The adjective 'promptly' is used to emphasize the fact that the action takes place without undue delays; 'promptness' allows the delays of the kind encountered in transmission of signals, but does not allow indefinite delays as might be encountered if the action has to await arrival of some input signals. Moreover, if a transition that does not conflict with any of the transitions which already have permission from the conflict structure, sends a request to the conflict structure, then the conflict structure gives permission

to at least one such transition promptly.

3. Upon receiving permission from the constraint structure, a transition sends a request to the conflict structure.

4. From 1, 2 and 3 it can be concluded that a transition that gets permission from the constraint structure either initiates (promptly) or is promptly disabled by a conflicting transition.

5. A ready transition, i.e. a transition whose input places are in B_u^τ and whose input-condition is in AC_α^τ, sends a request to the constraint structure for permission unless it is disabled by a conflicting transition in the meantime.

6. If there are any ready transitions whose output places can be admitted to the set of active places without violating the constraints, then the constraint structure gives permission to at least one of them.

7. An active transition is promptly terminated following termination of the associated event.

8. An active input-condition is promptly terminated following termination of the associated transition.

From the steps above it can be concluded that if there are any ready transitions that can be initiated without violating the constraints then at least one of them is initiated promptly, if there is any transition that can be terminated then it is terminated promptly, and if there is any input-condition that can be terminated then it is terminated

promptly. Thus if the signal history is not a terminal history, the structure advances the history promptly by producing a signal (corresponding to initiation of a transition, termination of a transition or termination of an input-condition.

To show that step 1 in the proof of the theorem holds it should be shown that: (i) In giving permission to a transition the conflict structure denies permission to conflicting transitions by not acknowledging enable signals from them until disable signals are received. (ii) The transition which gets permission from the conflict structure promptly claims the stones at the input places of the transition and causes disable signals to be sent to the conflicting transitions (if any). (iii) When a disable signal is sent to a transition from a place, the portion of the coordination structure associated with that transition promptly returns to the condition it would be in if the transition had not received an enable signal from the place. (iv) When stone signals are sent to the transition from all input places of the transition, the transition structure initializes the part of the conflict structure associated with the transition and initiates the transition by sending a ready signal on link 3 of the associated T-module. These results are proved in a number of lemmas given below.

The following sublemma states that only one of a set of conflicting transitions is permitted to proceed by the conflict structure

Sublemma 5.12 Among the incident links of the conflict structure exactly one of the links associated with conflicting transitions can be active (Figure 5.8) at any given time.

Proof. From the P-net specification of a conflict module (Figure 4.24) it can be seen that the net associated with an incident link must pick up the stone at place p_0 before activating the link, and that the stone is not returned until the link is disabled. As a result, only one of the incident links of a conflict module can be active at any given time.

--- (1)

Moreover, the emerging link associated with the incident link is activated before the incident link is activated, and is disabled only after the incident link is disabled. Therefore, an active incident link of the conflict structure is active throughout the conflict structure.

--- (2)

By virtue of the construction of the conflict structure, links associated with conflicting transitions have at least one conflict module in common. Therefore, from (1) and (2), only one link among the links associated with conflicting transitions is active at any given time.

■

The next sublemma states that the transition that gets permission from the conflict structure (promptly) claims the stones at its input places. It should be recalled that to claim the stone at a place (i.e. the stone equivalent in the place structure) the transition must be the first one to acknowledge the enable signal sent to it by the place structure. This is indeed the case because in giving permission to the transition, the conflict structure blocks conflicting transitions; the conflict structure does not acknowledge enable (request) signals from conflicting transitions until they are disabled by the place structures.

Sublemma 5.13 A transition which is permitted to proceed by the conflict structure promptly claims the stones at its input places and causes the places to send disable signals to conflicting transitions. When the conflicting transitions are disabled, as indicated by acknowledge signals for the enable and disable signals, the places send stone signals to the transition that claimed the stones.

Proof. From the P-net specification of the T,IT and EP modules and the arrangement of these modules in the precedence structure the following can be observed:

1. Upon receiving permission (acknowledgement for the enable signal) on link 2, the T-module (Figure 4.16) acknowledges the enable signal on link 0, and the IT-module (Figure 4.15) in turn acknowledges

enable signals on the incident links. These acknowledge signals reach the EP-modules corresponding to the input places of the transition. At each input place, this transition is guaranteed to be the first one to acknowledge the enable signal because the other transitions are blockaded in the conflict structure, i.e., the conflict structure does not acknowledge the enable signals from them until they are disabled.

2. When the stone at a place is claimed by a transition, the EP-module acknowledges the enable signal on the incident link and sends disable signals to the other transitions.

When the acknowledge signal from the EP-module is received, the P-module immediately sends a stone signal to it. When the other (conflicting) transitions acknowledge the disable signals (and also the enable signals), the EP-module sends a stone-signal to the transition which claimed the stone.

From these observations the statement of sublemma follows.

■

The next three sublemmas deal with the influence of a disable signal on the structure associated with a transition. The disable signal turns the communication cycle which was started with the enable signal into a void cycle. That is to say that the disable signal nullifies the influence of the enable signal that was sent to the transition. The modules have been so specified that on receiving a disable signal on an incident link they initialize to the condition they would be in if the enable signal had not been received, but before returning to this condition they send a disable signal to the structures connected to the emergent links, if enable signals have been sent to these emergent links,

so that they too are disabled in a similar manner. The modules signal completion of the initialization by acknowledging the enable and disable signals.

Sublemma 5.14 In response to a disable signal on an incident link the conflict structure initializes its substructure associated with that link and acknowledges the enable and disable signals.

Proof. From the P-net specification of a conflict module (Figure 4.24) it can be observed that the response of the module to a disable signal on an incident link is as follows:

(i) If the module has not yet sent an enable signal on the corresponding emergent link, it immediately initializes itself and acknowledges the enable and disable signals on the incident link.

(ii) In case the module has sent an enable signal on the corresponding emergent link, it sends a disable signal on the emergent link, and when these signals (the enable and disable signals) are acknowledged, it initializes itself and acknowledges the enable and disable signals on the incident link.

(iii) The sink module used in terminating a double link acknowledges the enable and disable signals without any delay and returns to its initial condition.

From these observations the sublemma follows.

■

The following sublemma is similar to the sublemma above, but deals with the constraint structure instead.

Sublemma 5.15 In response to a disable signal on an incident link the constraint structure initializes its substructure associated with the link to the condition it would be in if the associated enable signal had not been received, and signals completion of the initialization by acknowledging the enable and disable signals.

Proof. From the P-net specification of IR, R and const modules and their arrangement in the constraint structure, one can observe that:

1. The response of the IR-module to a disable signal is as follows:

   a) If the module has not acted on the enable signal or if at that time the other incident link is connected to the emergent link, the module immediately initializes its portion associated with the link (on which the disable signal was received) and acknowledges the enable and disable signals.

b) If the other incident link is idle and the module has sent an enable signal on the emergent link, then upon receiving the disable signal the module sends a disable signal on the emergent link to nullify the effect of that enable signal. When it receives an acknowledgement for the disable signal and for the enable signal (if it has not been received already) it initializes itself and acknowledges the disable signal (and also the enable signal unless that has been acknowledged already) on the incident link.

2. The response of the R-module is as follows:

a) If the communication on the emergent link of the module is in a completed state when the disable signal is received, the module immediately returns to its initial condition and acknowledges the enable and disable signals on the incident link.

b) If the communication on the emergent link of the module is incomplete, the module sends a disable signal on the emergent link (if it has not been sent already). When, as a result of the disable signal, the communication is completed, the module returns to its initial condition and acknowledges the enable and disable signals on the incident link.

3. The response of the constraint module to a disable signal on an incident link is as follows: (Recall that a communication cycle on the mixed-triple link consists of a communication cycle on the disable sublink and a communication cycle on either the enable or the check sublink.)

case 1 A communication cycle involving a cycle on the check sublink.

a) If the module receives the disable signal before it has acted on the check signal, (a ready signal on the check sublink) the module immediately initializes its portion associated with the link and acknowledges the check and disable signals (figure 4.23)

b) If the module has acted on the check signal (by sending a check signal on the corresponding emergent link), the module sends a disable signal on the emergent link to nullify the check signal. When, as a result of the disable signal, the communication cycle on the emergent link is completed, the module initializes the portion associated with the link and acknowledges the check and disable signals.

case 2 A communication cycle involving a cycle on the enable sublink.

The response of the module in this case is similar to that in case 1 above except in this case an enable signal is involved instead of a check signal.

The T-module, on receiving a disable signal on link 0, sends disable signals to the constraint structure and the conflict structure. A disable signal is sent to the conflict structure only if an enable signal was sent to it already. From sublemmas 5.14 and 5.15 the disable signals initialize these structures so that they are returned to the condition they would be in if the enable signals had not been sent to them. The enable and disable signals are then acknowledged to indicate that the initialization is complete. When the acknowledge signals are received, the T-module acknowledges the enable and disable signals on link 0.

2. From the P-net specification of the IT-module (figure 4.15) the following can be observed:

If the IT-module is waiting for an enable signal on the other incident link (from another place structure), the module immediately acknowledges the enable and disable signal from the place under consideration. In this case no communication is involved on the emergent link. On the other hand if the module has already sent an enable signal on the emergent link, the module sends a disable signal on the emergent link to nullify the enable signal. From part 1 of this proof, the disable signal nullifies the enable signal and both of them are acknowledged. Upon receiving these signals, the IT-module returns to the condition it would be in if no signals were received on the incident link under consideration and acknowledges the enable and disable signals.

From 1 and 2 above the sublemma follows.

■

Proof: From sublemma 5.13 a transition which is permitted by the conflict structure promptly claims the stones at its input places and causes the places to send disable signals to the conflicting transitions. From sublemma 5.16 these disable signals promptly disable the conflicting transitions by initializing the part of the structure associated with them to the condition it would be in if the enable signals notifying the presence of stones at the input places were not sent to them. The disabled transitions signal completion of the disabling process by acknowledging the enable and disable signals. When these signals are received, the places send stone signals to the transition under consideration (sublemma 5.13), and upon receipt of stone signals from these places the transition structure promptly initiates the transition by sending out a ready signal on link 3 of the associated T-module (sublemma 5.17).

■

Lemma 5.10 If there are any enabled incident links of a conflict structure which do not individually conflict with any of the incident links which are active, then at least one of them becomes active. (See figure 5.7 for the definition of the active condition of a double link.)

In other words, if transitions that do not conflict with any of the transitions which are in the process of initiating, send requests to the conflict structure, then at least one of them is given permission to initiate.

Proof. The proof is by induction on the number of conflict modules in the conflict structure. Let n denote the number of conflict modules in the conflict structure.

Basis ($n = 0$): The case when there is no conflict module in the conflict structure. In this case all incident links are terminated by sink modules and none of the incident links conflict with any other links. Since a sink module immediately acknowledges the enable signal, the result of the lemma holds.

Induction: The lemma is assumed to hold for $n = r$ and is shown to hold for $n = r + 1$.

Consider the conflict structure as consisting of two parts: i) the conflict module nearest to the incident links and ii) the remaining portion of the conflict structure having r conflict modules.

From the P-net specification of the conflict module (figure 4.24) it can be observed that if some incident links of the module are enabled while none of them is in the active condition then the module enables the emergent link corresponding to one of the enabled incident links (if it has not done so already), and when the emergent link is activated the module activates the corresponding incident link. Thus at least one of the enabled links under consideration will be enabled past the first conflict module in the structure, i.e., one of the links incident on the remaining structure which does not conflict with any of the active links will be enabled. Since the remaining

conflict structure has only r conflict modules, from the hypothesis it follows that one of the links incident of this part of the conflict structure will be activated. If this link bypasses the first conflict module then it is one of the incident links of the structure, while if it passes through the first conflict module then the module activates the corresponding incident link.

The lemma thus follows.

■

Lemma 5.11 A transition which is permitted to proceed by the constraint structure promptly sends a request to the conflict structure for its permission.

Proof. From the P-net specification of the T-module (figure 4.16) it can be observed that when the constraint structure gives permission to the transition (to proceed) by acknowledging the enable signal on link 1, the T-module immediately sends an enable signal to the conflict structure on link 2. The enable signal sent to the conflict structure from the T-module represents a request on part of the associated transition for permission to initiate.

■

Lemma 5.12 A transition which gets permission from the constraint structure either initiates promptly or is promptly disabled by a conflicting transition.

Proof. From lemma 5.11 the transition which is given permission by the constraint structure promptly sends a request to the conflict structure. From lemma 5.10, unless some conflicting transition is given permission by the conflict structure, it is given permission by the conflict structure too. If the transition is given permission, it initiates after promptly disabling the conflicting transition (lemma 5.9). On the other hand if a conflicting transition is given permission, the conflicting transition promptly disables the transition under consideration (lemma 5.9).

■

Lemma 5.13 A ready transition, that is a transition whose input places are in B_u^T and whose input-condition is in CA_α^T, sends a request to the constraint structure for permission to proceed unless it is disabled by a conflicting transition in the meantime.

Proof. From the P-net specification of the T, ET, IP, P, EP and IT modules the following can be observed:

1. As soon as the transition associated with a T-module is terminated, i.e. an acknowledge signal is received on link 3, the module sends an enable signal on link 4.

2. As soon as the ET-module receives an enable signal, it sends enable signals on the emergent links.

3. The IP-module sends an enable signal on the emergent link for each signal received on an incident link. If the module is not serving some other link when an enable signal is received on an incident link, the module immediately sends an enable signal on the emergent link, otherwise the transmission of the signal is temporarily delayed until the module is released by that link. (see observation 8)

4. The P-module sends an enable signal on the emergent link as soon as it receives an enable signal (a ready signal) on the incident link.

5. The EP-module sends enable signals on the emergent links as soon as it receives an enable signal on the incident link.

6. Upon receiving enable signals on both incident links, the IT-module sends an enable signal on the emergent link unless in the meantime one of the enable signals on the incident links is nullified by a disable signal.

7. Upon receipt of an enable signal on link 0, the T-module promptly sends a request to the constraint structure for permission to proceed.

8. From lemma 5.2 the places which have stones in the coordination net are <u>on</u> in the coordination structure. The

places in $B(B^0,H^T)$ are in the on condition. Being in the on condition means that some transition for which this place is an output place has begun the process of sending the enable signal to the place (following the termination of the transition). The enable signal sent to the place corresponds to a stone. Coordination nets being safe a place can be on on account of at most one transition at a time. Therefore in the observation 3 the IP-module is called upon to serve only one incident link at a time.

From the observation above, all input places of a transition enabled for the stone distribution $B(B^0,H^T)$ send enable signals to the IT-module of the transition. If the transition is not an input transition, these enable signals account for all incident links of the IT-module. If the transition is an input transition, this condition is reached when the associated input-condition sends an enable signal. The IT-module sends an enable signal to the T-module unless one of the enable signals is nullified (by a disable signal) in the meantime. The T-module then promptly sends a request to the constraint structure for permission to proceed. ■

A portion of the constraint structure consisting of a sequence of constraint modules together with the mixed-triple links which pass through them and the sink modules which terminate the links is called a partial constraint structure; the portion of the constraint structure below the R-modules is a partial constraint structure and so is the

portion below any constraint module. Earlier in this chapter (sublemma 5.9) it was shown that the constraint structure does not activate an enabled incident link if it cannot be admitted without violating the constraints. This is true because a partial constraint structure does not activate an enabled incident link if it cannot be admitted without violating the constraints associated with it. The following sublemma shows that if there are any enabled incident links which can be admitted without violating the constraints then the partial constraint structure activates at least one enabled incident link. (For the different conditions of a mixed-triple link see figure 5.8.)

Sublemma 5.18 If any of the enabled incident links of a partial constraint structure can be activated without violating the constraints, then at least one of the enabled links is activated (promptly); the others are (promptly) put into either an active or an inactive condition.

Proof. The proof is by induction on the number of constraint modules in the partial constraint structure. Before taking up the proof, the following properties of the constraint structure should be observed from the P-net specification of the module (figure 4.23). It should be recalled that a constraint module is said to be saturated when it has admitted all but one of the links incident on it. A constraint module is said to have admitted a link as soon as it allows the enable signal on the link to pass through it; the link is said to be in the admitted condition until it is either disabled or turned inactive.

1. If the constraint module is not saturated and some of its incident links are enabled, the module admits at least one of them by enabling the corresponding emergent link. If the emergent link is subsequently activated, (i.e. the enable signal is acknowledged positively) the module activates the corresponding incident link, and if it is put into inactive condition, the module makes the corresponding incident link inactive.

2. If the constraint module is saturated, it immediately makes the enabled incident link inactive, that is, the constraint module immediately returns a negative acknowledgement for the enable signal.

From these observation it can be concluded that a constraint module promptly either admits an enabled incident link or make it inactive.

Now, to continue to the proof, let the number of constraint modules in the partial constraint structure be denoted by n.

Basis ($n = 0$): In this case all incident links of the structure are terminated by sink modules. Since a sink module immediately returns a positive acknowledge signal for an enable signal sent to it, the sublemma is valid for the basis.

Induction The sublemma is assumed to hold for $n = r$ and is shown to hold for $n = r + 1$. Consider the partial constraint structure in two parts: i) the first constraint module from the top and ii) the remaining substructure below the first constraint module.

Consider the progress of an enabled link which can be activated without violating the constraints. The progress need by observed only up to the moment when some incident link, not necessarily the same as the one under consideration, is activated.

If the link under consideration bypasses the first constraint module, it is directly incident on the the sub-structure. In this case from the hyposthesis of the induction step, at least one of the incident links of the sub-structure is activated (the sub-structure has only r constraint modules). If the link which is activated bypasses the first module, it is itself an incident link of the partial structure under consideration. On the other hand, if the link passes through the first module, the module activates the corresponding incident link.

If the enabled link whose progress is being considered passes through the first module, the remaining links of the first module could not all be active, but it could be that they have been admitted by the module. The links which have been admitted but not activated will promptly be made either active or inactive by the sub-structure. If any link is activated, the corresponding incident link is activated. On the other hand if any of the links is rejected, the link under consideration will be admitted. From the hypothesis, then, at least one of the incident links of the sub-structure will be activated and this will lead to activation of the corresponding incident link of the structure under consideration.

Sublemma 5.19 If an enabled incident link of a partial constraint structure inclusive of the R-module cannot be activated without violating the constraints, that is it cannot be added to the set of links already active without violating the constraints, then the progress of the link comes to a halt at some (saturated) constraint module which blocks a check signal on the link.

Proof. If a link cannot be added to the set of links already active without violating the constraints, then by virtue of the construction these exists a constraint module through which this link passes, all of whose other links of are already active. The module is therefore saturated and from the P-net specification of the module, it blocks the check signal on the link. Therefore, if the R-module sends a check signal, it is blockaded at one such module. What needs to be considered is the possibility that the R-module has already sent an enable signal on behalf of the link. In this case the saturated module will return a negative acknowledge signal in response to the enable signal and the R-module will then send a check signal which will get blocked at the saturated constraint module.

■

Sublemma 5.20 If any of the enabled incident links of a partial constraint structure inclusive of the R-modules can be activated without violating the constraints, then at least one of them is activated.

Proof. If none of them is activated then the following must hold.

1. The progress of the links which cannot be activated without violating the constraints will soon be halted at some constraint modules as the check signals on those links are blocked at these modules.

2. When the progress of all links which cannot be activated without violating the constraint is blocked, only those links which can be activated without violating the constraints can be enabled (past the R-module). If there is any such enabled link then, from sublemma 5.18 at least one of them is activated. If none is already enabled then at least one of them will be, as the check signal sent out on it by the associated R-module will reach the sink module unobstructed and will be acknowledged positively, whereupon the R-module will enable the link.

■

Lemma 5.14 If any of the enabled incident links of the constraint structure can be individually activated without violating the constraints, then at least one of them is activated.

Proof. Consider the progress of an enabled incident link which can be activated without violating the constraints.

1. If the emergent link of the IR-module on which the link is incident is already active, (on account of the other incident links), the IR-module immediately activates the incident link under consideration. If the emergent link is idle, the module enables the emergent

link and waits for it to be activated. If the emergent link has been enabled already on account of the other incident link, the module just waits for it to be activated. In any case, it follows that the emergent link is enabled if it is not active already.

2. Now consider the partial constraint structure inclusive of the R-module. Since the link under consideration (which is enabled by the IR-module) can be activated without violating the constraints, it follows, from sublemma 5.20, that at least one of the incident links of this partial constraint structure is activated, and shortly thereafter the corresponding incident link (or links) of the constraint structure is activated. ■

Lemma 5.15 Following time slice τ, if there are any ready transitions which can be individually initiated, that is the output places of the individual transitions can be added to the set of active places without violating the constraints, then the constraint structure promptly gives permission to at least one of them.

Proof.

1. If some incident links of the constraint structure associated with transitions other than those which are either active or have terminated but whose output stones have not been removed by other transitions, are active then they correspond to ready transitions which have been given permission but have not had time to initiate, i.e. the constraint structure has already permitted some ready transition. If none of the ready transitions have been given permission the following holds:

2. From lemma 5.13, the ready transitions send requests to the constraint structure unless they are disabled, but none of them will be disabled until at least one of them gets permission from the constraint structure and proceeds to initiate. Therefore, some ready transitions which can be individually initiated without violating the constraints promptly send request to the constraint structure. It remains to be shown that at least one of them is given permission by the constraint structure:

3. A transition disables its link to the constraint structure after its output stones have been used up by some other transitions (the initialization structure does the same too). Therefore, in due time only (i) the incident link corresponding to those transitions that have occurred but whose output stones are still in output places and (ii) the links from the initialization structure corresponding to initial stones which are still there, are active. The link associated with a ready transition whose output places can be added to the set of active places without violating the constraints can be active, together with those links already active, without violating the constraints. Therefore from lemma 5.14 the constraint structure activates the link associated with at least one ready transition and, in so doing, gives it permission.

Lemma 5.16 If following time slice τ there are any ready transitions which can be individually initiated, that is, the output places of the individual transitions can be added to the set of active places without

violating the constraints then the coordination structure initiates at at least one of them.

Proof. This lemma follows directly from lemmas 5.15 and 5.12 . Lemma 5.15 says that at least one of these transitions is given permission by the constraint structure and lemma 5.12 says that if some transitions are given permission by the constraint structure then at least one of them initiates.

■

Lemma 5.17 If a transition t is in AT^{τ} and ie(t) is in IE^{τ}_{α}, then the transition is promptly terminated.

Proof. That transition t is in AT^{τ} means that the transition is active, i.e. it has not been initiated, and that ie(t) is in IE^{τ}_{α} means that the associated event has terminated. For the coordination structure this means that an acknowledge signal has been returned to the transition structure (on link 3 of the T-module) but the signal has not reached the structure. Since signals sent on links at one end reach the other end promptly, the acknowledge signal promptly reaches the transition structure and the transition is terminated.

The lemma thus follows.

■

Lemma 5.18 If there is any input-condition in AC_β^τ then it is promptly terminated.

Proof. An input condition is placed in the set AC_β when the associated transition has terminated (following its initiation). The P-net specification of the T-module (figure 4.16) shows that the T-module associated with the transition acknowledges stone signal on link 0 as soon as the transition is terminated. This acknowledge signal reaches the P-module associated with the input condition. The P-module then immediately returns an acknowledge signal on the incident link corresponding to that input condition i.e. the input-condition is terminated.

Thus the lemma follows.

■

Theorem 2 The coordination structures are active; that is, if the simulation sequence $\underline{X}^\tau$ produced by a coordination structure until time τ is not a terminal simulation sequence, the structure (promptly) advances the simulation sequence (by producing a signal that advances the signal history).

Proof. The proof of this theorem follows directly from lemma 5.16, 5.17 and 5.18 for when the history is not terminal then the conditions of at least one of these lemmas are met and according to the lemmas the structure produces a signal which advances the signal history.

■

5.8 Proof of Theorem 3

Theorem 3 The coordination structures are attentive; that is, if after a coordination structure has produced a simulation sequence X, termination of transitions is suspended, the simulation (promptly) reaches a simulation sequence that is complete.

Proof. Consider the simulation of the net that is represented by the simulation of the structure (that implements the net). As soon as termination of transitions is suspended, the stone distribution in the net is frozen as only termination of transition alters stone distribution. Therefore, no new transitions can be enabled, and a transition which cannot be initiated without violating the constraints continues to be that way. There being only a finite number of transitions in the net, the number of enabled transition is also finite. Thus after termination of transitions is suspended only a finite number of transition which are already enabled and whose initiation would not violate the constraints can be initiated. Now, from lemma 5.16 if there are any enabled transition whose input-condition are active, then the structure continues to initiate transitions until none such transition if left. Furthermore, from lemma 5.18, if there are any active input conditions and the transitions associated with them are already terminated (using them), then those input-conditions are promptly terminated by the structure.

Thus the simulation of the structure promptly reaches a simulation sequence which cannot be advanced by initiating a transition or terminating an input-condition.

The theorem thus follows.

■

5.9 Proof of Theorem 4

Theorem 4 Each simulation sequence of a coordination net C corresponds to some simulation sequence of the coordination structure S that implements the net.

In the following discussion, s refers to a signal of the kind r_{t_i} which represents a ready signal associated with transition t_i, and $\underline{H}^{\tau}\cdot s$ refers to the signal history that results when signal s is added to the signal history $\underline{H}^{\tau}$. Similarly $\underline{H}^{\tau}\cdot s_1 s_2 \ldots s_n$ denotes the signal history obtained by adding signals $s_1, s_2, \ldots, s_n$ to history $\underline{H}^{\tau}$ in that sequence.

Sublemma 5.21 If $\underline{H}^{\tau(j)}$ is a signal history produced by a coordination net up to time slice $\tau(j)$, and there exists a signal s such that $H(\underline{H}^{\tau(j)}\cdot s)$ is feasible then $H^{\tau(j)}\cdot s$ is a possible signal history of the structure at the successive time slice $\tau(j+1)$.

Proof. Consider initiation of transitions in the coordination structure. Which of the transitions that are ready at time slice $\tau(j)$ but which conflict over input places or cannot be all initiated because of the restrictions on account of constraints, initiate depends on the outcome of a race among them in the constraint structure and in the conflict structure; those transitions which do not conflict and which are not restricted by the constraints proceed independently. Since delays in transmission of signals can be arbitrary, both in the constraint structure and in the conflict structure, any of the competing transitions could win the race. Thus a transition which is ready at time slice $\tau(j)$ could be the next transition to initiate. Similarly, an input-condition which can be terminated following time slice $\tau(j)$ could be the next input-condition to terminate, an event which can be initiated following time slice $\tau(j)$ could be the next event to be initiated, and a transition which can be terminated following time slice $\tau(j)$ could be the next transition to be terminated. Initiations of input-conditions and terminations of events is controlled by the external world. Thus depending on the external world, a given input-condition could be the next input-condition to be initiated, and similarly a given event could be the next event to be terminated.

As the signals mentioned above occur independently and delays in transmission of signals are arbitrary, any of them could be the (next) signal that advences the time slice $\tau(j)$ to $\tau(j + 1)$. Thus the sublemma holds.

■

Lemma 5.19 If $\underline{H}^{\tau(j)}$ is a signal history produced by a coordination net up to time slice $\tau(j)$, and there exixts a set of signals $s_1, s_2, \ldots, s_n$ (an occurrence set) such that $H(\underline{H}^{\tau(j)} \cdot s_1 s_2 \ldots s_n)$ is feasible and adjacent to $H(\underline{H}^{\tau(j)})$, then it is possible for $\underline{H}^{\tau(j)} \cdot s_1 s_2 \ldots s_n$ to be the signal history of the structure at the successor time slice $\tau(j+1)$.

Proof. Since $H(\underline{H}^{\tau(j)} \cdot s_1 s_2 \ldots s_n)$ is feasible and adjacent to $H(\underline{H}^{\tau(j)})$, signals $s_1, s, \ldots, s_n$ correspond to distinct transitions, input-conditions and events. Moreover, they could occur in any sequence as occurrence of any of them does not affect the feasibility of any of the others (see definition 5.3). From the above fact and sublemma 5.21, it follows that these signals could be the next n signals that proceed to occur, and since they are independent and travel on links with arbitrary delays, they could occur at the same time.

The lemma thus follows.

■

Theorem 4 Each simulation sequence of a coordination net C corresponds to some simulation sequence of the coordination structure S that implements the net.

This theorem is a direct consequence of lemma 5.19 and the definitions of feasible history and simulation sequence of coordination nets (definitions 5.2 and 5.5).

CHAPTER 6

CONCURRENT MANAGEMENT OF SHARED RESOURCES

6.1 Introduction

An important application of the scheme for the coordination of events is in the management of computing resources in a parallel computer system where the shared computing resources are kept in a pool from which loans can be made to the users upon demand. Proper management of such resources involves (i) ensuring that conflicting assignments are not made, (ii) overly generous assignments are not made (i.e. more than the amount requested is not assigned), (iii) the resources are not kept idle unnecessarily, and (iv) that the system is free of any hang-ups on account of mismanagement of the resources.

In sequential computer systems, some of these requirements are naturally met, so much so that in many instances resource management is considered synonymous with resource allocation even though resource management involves more than just resource allocation. In parallel computer systems in which resources are assigned to the user concurrently, these requirements are not met automatically and the role of parts other than the allocator becomes important. A resource management system has three parts: an allocator, a selector and a connector.

The function of the allocator is to make decisions regarding the quantity of resources of various types to be allocated to the users. Such decisions are made either when a request for the use of some resources is received or when some resources are released by some users. An allocator does not distinguish between the individual members of a pool of resources of one type: the allocator allocates resources without indicating which particular resource units are to be assigned. This task is that of the selector. The selector, after assigning particular resources to the user, instructs the connector to make the assigned resources physically available to the user.

This chapter indicates the nature of the coordinations involved in concurrent management of shared resources. The coordination nets for the parts of the resource management systems can be considered to be software specifications of the systems and the corresponding coordination structures can be considered as their hardware implementations. If the speed of the resource management system is not critical and the coordination nets representing the system can be programmed in a language implemented on the computer then it is not necessary to consider the hardware implementation of the resource management system. Resource management systems implemented in the hardware are called arbiters [14]. The over all structure of an arbiter for concurrent management of resources is shown in figure 6.1

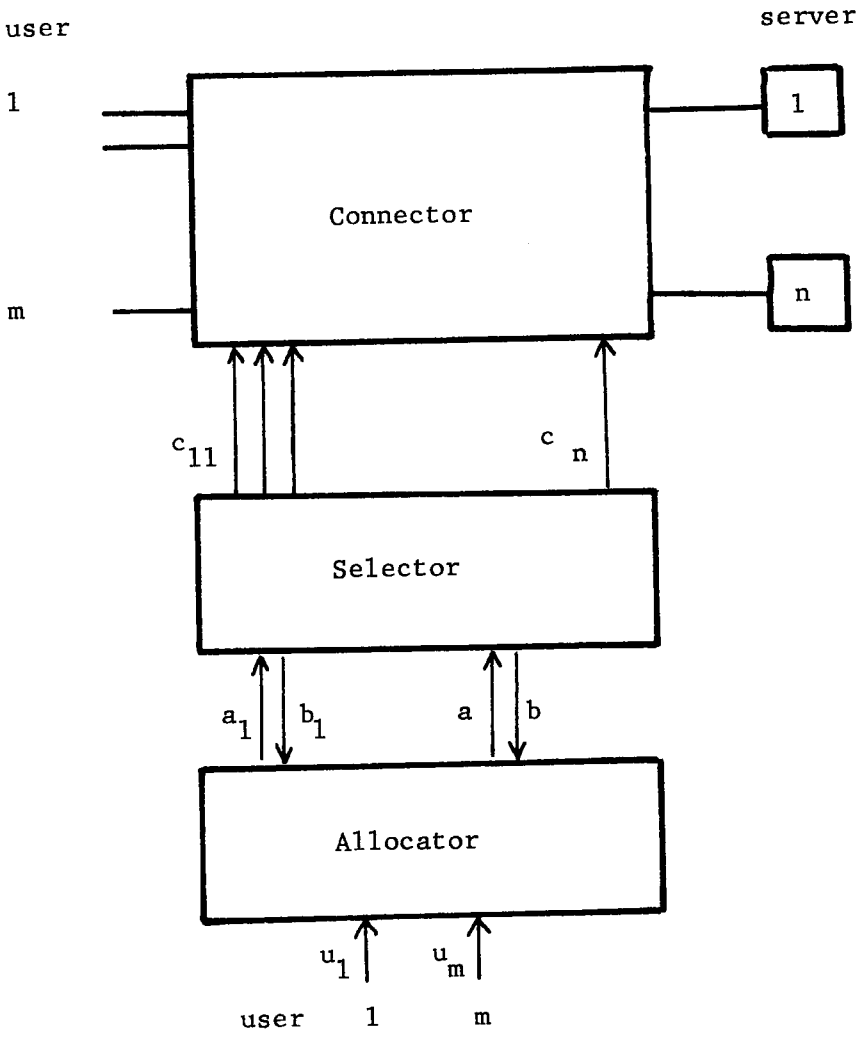


Figure 6.1 Arbiter for Concurrent Management of Resources

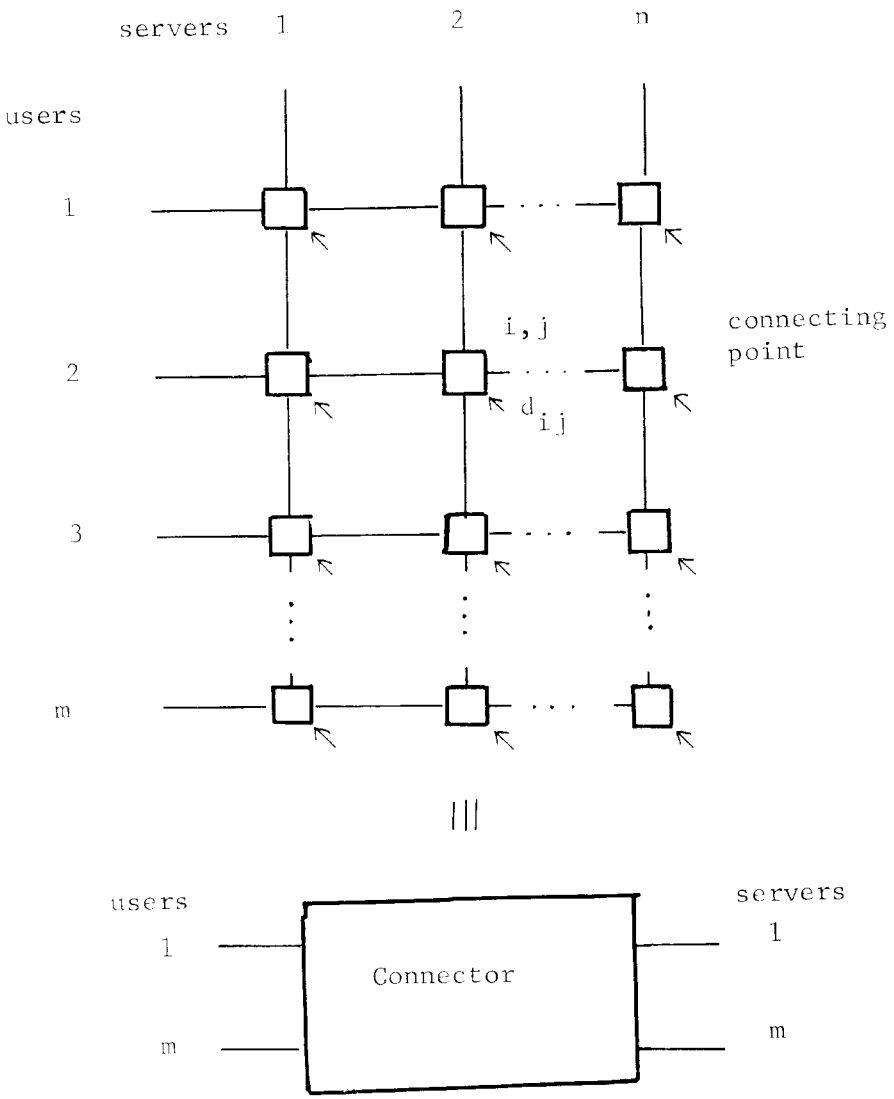
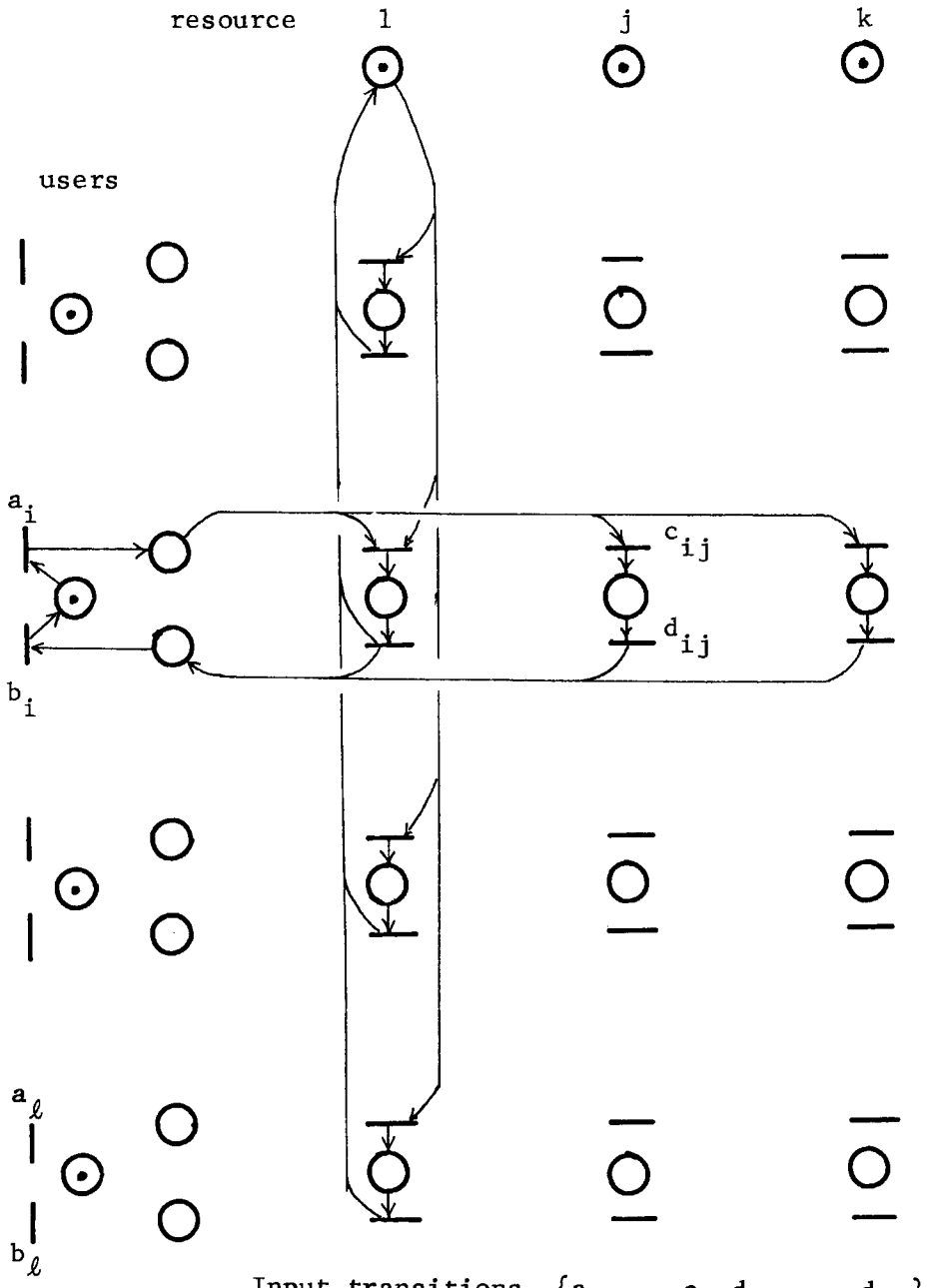


Figure 6.2 The Connector



Input transitions $\{a_1,\ldots,a_\ell;d_{11},\ldots,d_{\ell k}\}$

Output transitions $\{b_1,\ldots,b_\ell,c_{11},\ldots,c_{\ell k}\}$

Figure 6.3 A Coordination net for the Selector

labelled 'a' indicate that the corresponding user has been granted a unit of resource; when more than one resource is allocated, more than one input transition is associated with the user. The output transition labelled 'c_{ij}' indicates that the resource unit j is assigned to user i. The input transition labelled 'd_{ij}' indicates that user i has done with the use of the assigned resource, and the output transitions labelled 'b' indicate that the assigned resources have been released.

A coordination structure for implementing the coordination can be derived from the coordination net by the method presented in chapter 4.

The interconnection of a selector with the other structures in the arbiter is shown in figure 6.1 . The links incident on the selector that come from the allocator correspond to input transitions labelled 'a', the emergent links going to the allocator correspond to output transitions labelled 'b', the emergent links going to the connector correspond to output transitions labelled 'c', and the incident links from the connector correspond to input transitions labelled 'd'.

From the coordination net for the selector it can be seen that the selector does not assign a particular unit of resource to more than one user (as a resource unit is assumed not to be usable by more than one user at a time). Moreover the selector does not assign more than the requested amount of resource. Thus the selector meets the requirements of proper resource management.

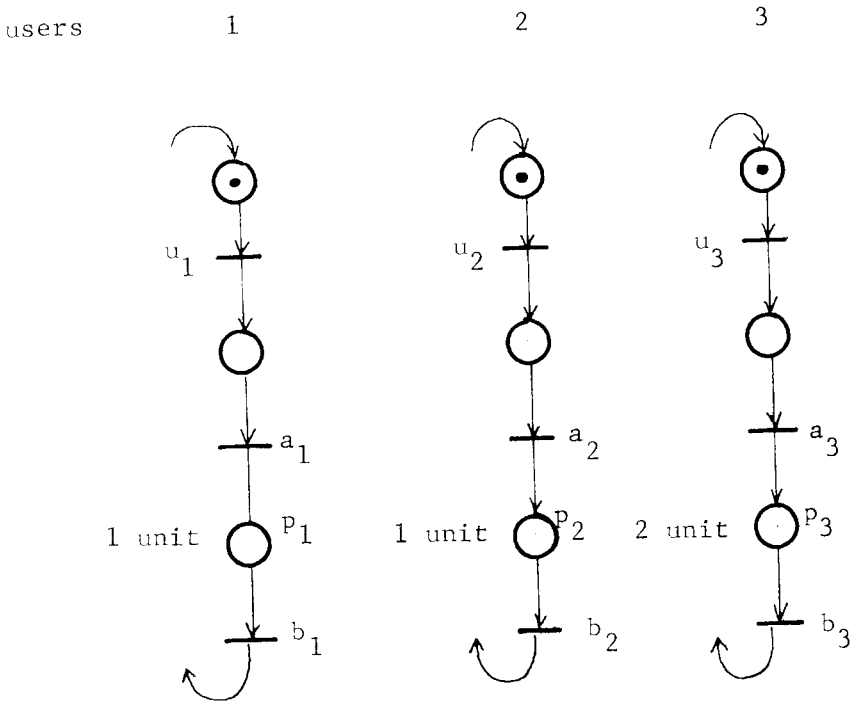
6.4 The Allocator

The task of the allocator is to allocate various amounts of resources to the users. In so doing the allocator must not allocate more than the available resources, and must not make allocations which lead to hang-ups. These requirements are part of the requirements of proper resource management.

An allocator is the part of the arbiter in which the allocation strategy of the resource management system is implemented. As allocation strategies vary widely depending on the nature of the users and the resources, instead of presenting a specific allocator, different aspects of coordination in allocators are discussed below with the aid of examples.

Consider a situation where there are two identical resource units and three users named 1, 2 and 3. User 1 and user 2 each need one unit of resource but user 3 needs two units of resource. Furthermore user 3 does not care for the resource unless he can get both the units at the same time. The users use the resources for a finite time and return them to the pool thereafter. The users are permitted to repeat their request as many times as they wish.

A coordination net defining an allocator for the above situation is shown in figure 6.4 . The constraint set in the coordination net ensures that events which would cause over-allocation are prevented from occurring.



Capacity 2 units

$$Ct = \{\{p_1, p_3\}, \{p_2, p_3\}\}$$

Figure 6.4 A Coordination net for an Allocator

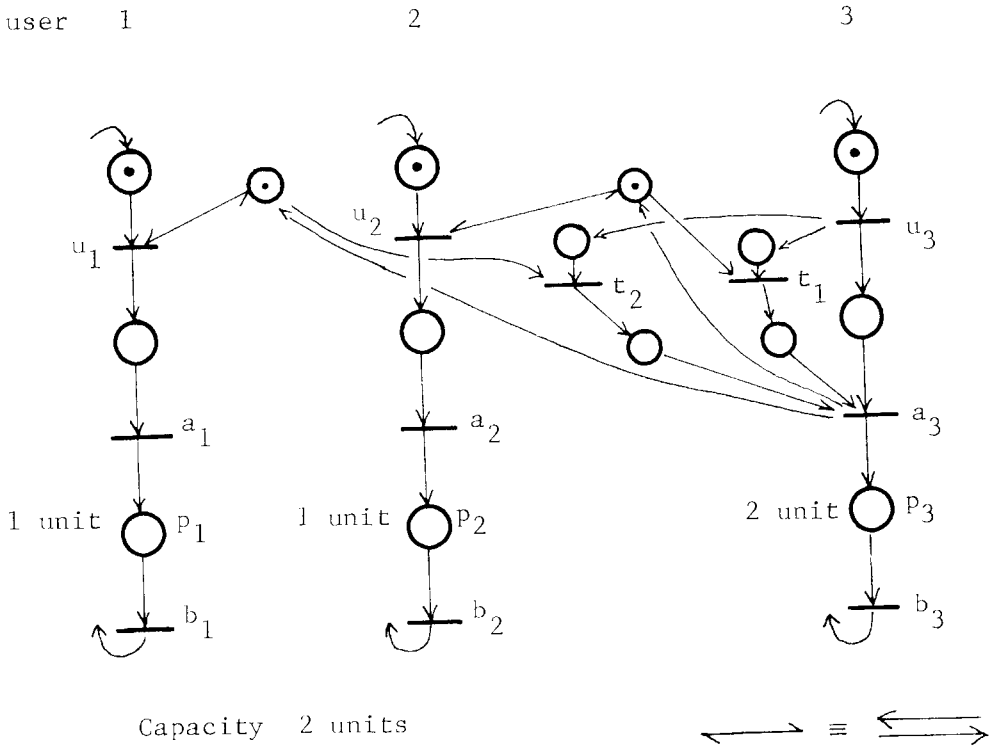
6.5 Fairness in Allocation

It may be recognized that the allocation strategy embodied in the above coordination net is unfair to user 3, for if users 1 and 2 cycle indefinitely, user 3 may not get any service. This defect of the allocator above can be remedied by adding some additional constraints as shown in figure 6.5 . In this coordination net the transitions t_1 and t_2 ensure fair treatment to all users.

6.6 Deadly Embrace and Safety

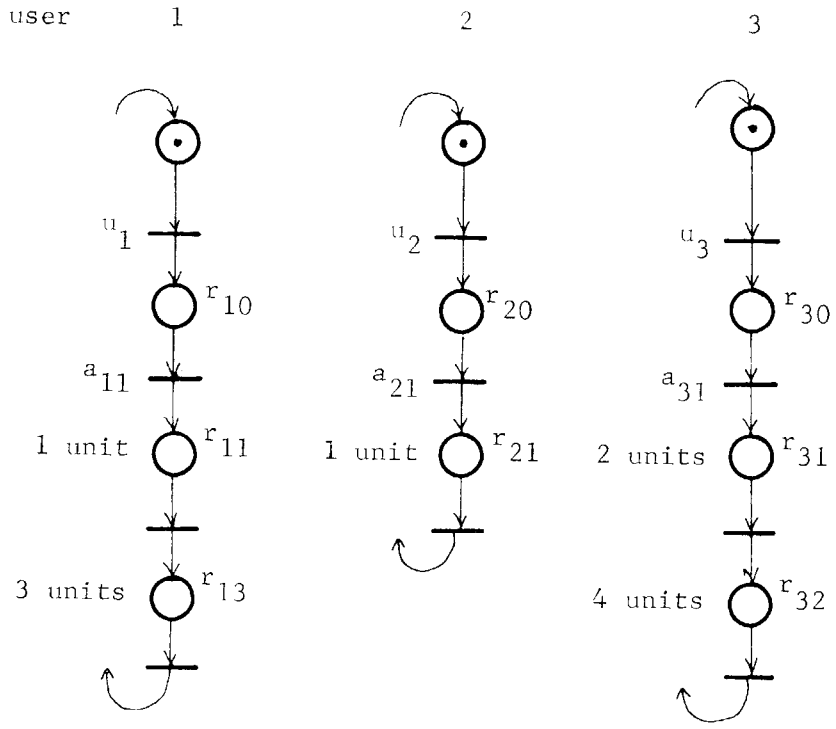
In addition to ensuring fairness, an allocator must not perform any allocation that leads to a hang-up. Figure 6.6 shows an allocator which permits hang-ups. This allocator handles four units of resource, and in this case the users, instead of requiring the resources at one time, require the resources in steps. The resources allocated at a step are not returned until end of the last step. For example, user 1 requres one unit of resource in the first step and two more units of resources in the second step, and the units allocated in either step one or step two are not returned until the job ends. Similarly user 3 requires two units of resources at first and two additional units of resources later before completing the job.

In this case consider a situation in which user 1 is in step one, user 2 is idle and user 3 requests resources for its first step. At this moment there being three free resource-units, the allocator may allocate two units of resource to user 3 to initiate the first



$$Ct = \{\{p_1, p_3\}, \{p_2, p_3\}\}$$

Figure 6.5 A Fair Allocator



$Ct = \{\{r_{12}, r_{31}\}, \{r_{21}, r_{32}\}, \{r_{11}, r_{32}\}\}$

Resource Capacity 4 units

Figure 6.6 An Allocator which has a Deadly Embrace

step. This allocation though feasible is unwise, for it results in a situation where neither user 1 nor user 3 can complete his job on account of a lack of adequate resources and neither user can return the allocated resources to let the other user complete his job. Hang-ups of these kind are referred to as deadly embraces. For proper operation an allocator should be free of any deadly embraces.

An allocation is said to be a safe allocation if there exists some subsequent allocations which permit all users to complete their jobs. Strategies for safe allocations are studied by Haberman [17], Shoshani, Coffman [18] and Hebalkar [19]. Haberman assumes a simple model where the users do not return allocated resources until the job is completely done (as in the above example), and assumes that the capabilities of the resources do not overlap. Shoshani and Coffman extend Haberman's model to allow the users to return some of the allocated resources at various points in the completion of the job. Hebalkar in his forthcoming thesis deals with multiple types of resources whose capabilities may overlap. The author would like to acknowledge Hebalkar's work as the source of the following treatment of safe allocations.

6.7 Coordination for Safety

The conditions (i.e. the stone distributions) in the coordination net representing an allocator represent the states of the allocator. Each of these states implies some allocation of resources. A table of these states with the state transitions (in the sense of

switching circuit theory) indicates how the allocation state could change. Since concurrent activities are involved, the often made assumption of a single variable changing at a time does not hold, and therefore the table includes transitions corresponding to multiple changes. A variable is associated with each user and its values correspond to the steps in the activity of the user with regard to his resource requirements.

What is required of the allocator is that it should not enter into certain undesirable states. The undesirable states are of two types: i) those which correspond to allocations which are not feasible because of the limited amount of resources and ii) those which are feasible but are unsafe because they lead to deadly embraces. The infeasible states can be determined easily as the resource requirement of each allocation state is known from the resource requirements of the users at various stages of their activity. In the case of the allocator of figure 6.6, states 012, 201, 102, 202, 211, 112 and 212 are infeasible as they require more than four units of resource but the system has only four units of resource at its disposal.

A state is safe if there exists an allocation path that involves only feasible states from that state to the idle state (i.e. 000). The test for safeness may involve a search through the state table (or the corresponding state diagram). Bounds on the lengths of such tests are studied in the works mentioned earlier.

It is interesting to note that even though multiple variable changes are permitted, an abridged state diagram giving only the transitions corresponding to single variable changes is sufficient to determine safeness of the states. Moreover there is no need to show transitions from the infeasible states. An abridged diagram for the allocator of figure 6.6 is shown in figure 6.7 . From this diagram it can be seen that states 101 and 111 are unsafe.

By ruling out infeasible and unsafe states as attainable states the behavior of the allocator is restricted to that of a strongly connected finite state machine whose states are the feasible states that are safe. The application of the scheme for the coordination of events to allocators can now be seen. Coordination of events provides a means for enforcing such a restriction on allocators. In the abstract world such restrictions are obtained through the coordination nets and in the physical world the restrictions are enforced through the coordination structures.

It may be recalled that states in the state diagrams for the allocator correspond to conditions in the coordination net for the allocator. The undesirable states can be ruled out by ruling out the corresponding conditions i.e. by including the conditions in the constraint set of the net. In the case of the allocator considered above, the constraint set $Ct = \{\{r_{11}, r_{31}\}, \{r_{12}, r_{31}\}, \{r_{11}, r_{32}\}, \{r_{21}, r_{32}\}\}$ enforces the necessary restrictions.

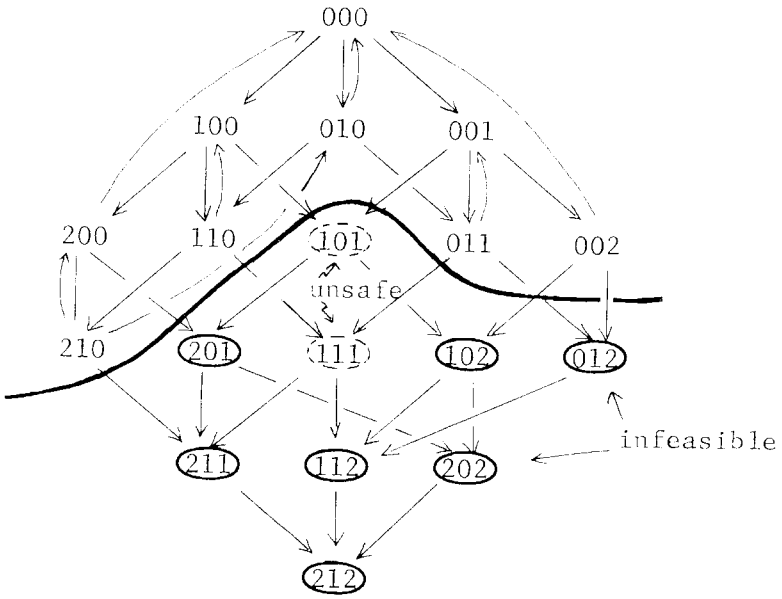


Figure 6.7 Restricting the Allocator to the Feasible States which are Safe

6.8 Multiple Types of Resources

The resources which are managed by the resource management system need not be of the same type. When more than one type of resource is involved, there are two cases to be considered, viz i) the functional capabilities of the resources of different types do not overlap and ii) the functional capabilities of the resources of different type do overlap, so that in some instances resources of one type can be used in place of resources of another type. In the first case the types of coordination nets used for systems with resources of a single type are adequate; the only change required is that the resource requirements of the different steps now involve more than one type of resource, and the infeasible states of the allocator are the states whose requirements for any type of resource exceeds the available resources of that type. From the knowledge of the infeasible states, the unsafe states can be found in the same way as in the case of resources of a single type.

The second case involves choices in the allocations. These choices are used by the allocator to improve the utilization of resources and services to users. Choices can be represented in the coordination nets by means of conflicts as in the case of the allocator described below.

Consider a situation in which there are two types of resource; one unit of resource type A and two units of resource of type B.

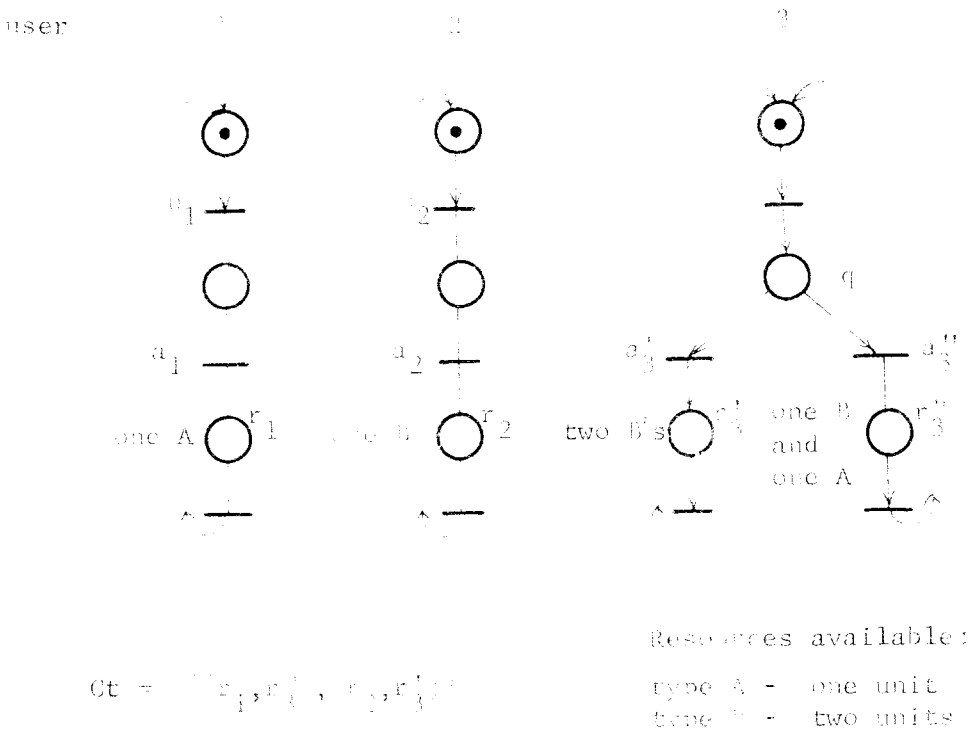


Figure 5.5 An Allocator with Alternatives

The system has three users; user 1 needs one unit of resource of type **A**, user 2 needs one unit of type B and user 3 needs either two units of type B or one unit of type A together with one unit of type B.

To use the resources fully, the allocator chooses between the alternatives of allocating either two units of type B or one unit of type A and one of type B to user 3 depending on the available free resources. A coordination net for the above allocator is shown in figure 6.8 . In the net, transitions a_3' and a_3'' which conflict with each other over place q represent the above mentioned alternatives. When place q gets a stone, the allocator makes the choice of allocation by choosing one of these two transitions. The choice is based on which of the resource requirements is met first; when the requirements for both alternatives are met at the same time (i.e. when one unit of type A and two units of type B are free to be allocated), the choice is made arbitrarily.

The treatment of safeness in this case is not much different from the earlier cases. The states of the allocator correspond to the condition in the net. States which include alternative conditions are ruled out as they cannot arise. The infeasible states can be found from the resource requirements of the states, and once the infeasible states are known, the unsafe states can be found just as in the previous cases.

CHAPTER 7

CONCLUSIONS

7.1 Introduction

The subject of this thesis is intimately connected with the following topics: i) languages and schemata for the representation of systems and the interaction of one system with another, ii) systematic design and implementation of modular hardware structures, and iii) theories relating to information in computing systems and behavior of systems. A discussion of these topics will bring out the areas of future research interest.

7.2 Languages and Representation of Systems

Conventional programming languages represent a special class of the nets of the kind discussed in this thesis. This class of nets have only one stone at any given time; the stone corresponds to the locus of control in the program. The locus of control wanders around in the program as the program is executed. A single locus of control is not adequate for representing systems, for systems frequently have several loci of control - even in the simplest case there is one locus of control for each independently operating unit in the system. Clear evidence of this fact is the provision of an I/O interrupt facility in computers. Multiple locus of control implies concurrency. Languages for specifying systems must therefore be equiped to represent concurrency.

The concept of concurrency in systems goes far beyond using parallel actions to attain greater speed. Concurrency is a manifestation of independence. In the extreme case there is complete independence: then the activities of the systems are completely uncoordinated. The more interesting case is, however, one in which activities are both in part independent and in part dependent. This dependence of otherwise independent activities is what is called interaction. Interaction is quite fundamental to systems as the only way a system affects the external world, man and other systems, is through interaction, and interaction implies concurrency (i.e. the presence of more than a single locus of control). The fundamental nature of concurrency is more real than arguments of speed and efficiency suggest; it is intimately connected with the semantics and understanding of systems.

Contemporary programming languages have developed from attempts to characterize functions and have ignored concurrency. Consequently they are not adequate for characterizing interaction of one system with others. It is no wonder, then, that these programming languages have not made substantial headway in being languages for system specification, for there is more to a system than functional transformation; in particular, there is the interaction of the system with the external world. For the same reasons these languages have failed to be languages for hardware specification. Hardware consists of an interconnection of functional units and the functional nature of the units is simple but the interaction among them is not.

7.3 Is Functionality an Appropriate Characterization of Systems?

A determinate system is one whose input-output relation is a many to one mapping, that is, the relation is a function. Still another way of stating this is that a determinate system gives the same output for repeated execution of a given program for given data, and therefore a determinate computation is also referred to as a reproducible computation. There is no doubt that reproducibility of computations is important. For it is very difficult for a system programmer to debug a faulty system whose activity is not reproducible. While there is no question that determinate systems are important, to say that only determinate systems are useful systems and that the systems which are not determinate are useless is a mistake. Consider an information retrieval system which has names of students from all schools. The system is a question/answer system. A news reporter is interested in talking to a student from M.I.T., and makes the following request of the system, "Give me the name of a student from M.I.T." . It happens that the system is implemented using a drum that continuously rotates and on which the names of the students together with the school they belong to are written. Upon receiving the request from the reporter, the system picks the first name of a student from M.I.T. that comes under the reading head and gives it to the reporter. It is clear that the system is not determinate (i.e. reproducible) as there is more than one student at M.I.T., and the reporter may get the name of some other student should

he make the same request once again. But the system has answered his request to his complete satisfaction and he should have no complaints about it. One could say that even though the input-output relation of this system is not a functional relationship, its behavior can be still described by a function whose one parameter is the position of the drum at the time the request is made. Implicit in this, however, is the assumption that one can have a complete information about the state of the universe.

Attempts by people to push the concept of a function to such limits show the immense faith they have in the notion that functions are a formal counterpart to the notion of algorithm, and they consider systems to be nothing more than manifestations of algorithms. Church made this thesis in the early days of investigations into these problems and ever since everyone has regarded it as true. It should be remembered that Church's thesis is only a claim; the claim cannot be proved, it must be accepted or rejected on the basis of experience [20]. Functions are not adequate for representing systems, they are at best adequate for representing systems without interaction. Interaction involves concurrency, and functions are not an adequate characterization of concurrent systems. Therefore the study of characterizations for concurrent systems is important.

There is still another important aspect to this, viz that of dealing with systems with only partial information about them, as in the case of the information retrieval system considered earlier in which the state of the system was only partly known. A theory that handles

systems with only partial information about them must be developed for a better understanding of the nature of systems.

7.4 Hardware Systems

Next consider the design of hardware systems. Even though theories for systematic implementation of hardware such as conventional switching circuit theory, have been studied at great length, when it comes to actual design of hardware, the theories are cast aside and the design is worked out on the basis of the emperical art of designing hardware systems, indicating that the theories do not meet the needs of the designer. There is no doubt that these theories have contributed a great deal to the understanding of the nature and the limitations of the structures they describe, but treatment of structures in these theories runs into combinatorial complexity of a magnitude quite unsuited to the designers who have their human limitations. There is a need for systematic theory of hardware design procedure which avoids such complexity and provides a method better matched to human ability. It is from these considerations that the modular and hierarchical approach to the design of machines is important. It was pointed out earlier that interaction among hardware units plays a significant role and that understanding of hardware systems requires understanding of the interaction. To enhance understanding of hardware systems the interaction among units must be made transparent. Hardware should be suitably structured as to explicate and simplify the interaction, especially if a formal and systematic treatment of hardware of practical importance is to be worked out.

Moreover, advances in hardware technology also call for simplification of interaction and modularity of hardware as the difficulty is no longer so much in putting devices onto a semiconductor chip as in interconnecting one chip with other chips.

One problem that has beset the hardware designer is that a system which is correctly put together on paper does not necessarily work correctly when the components are put together in real life. Thus a newly designed hardware system must undergo debugging to detect flaws which cannot be detected from the paper design. The modular asynchronous approach to design of coordination structures presented in this thesis is a step towards a design procedure for hardware in which the hardware runs correctly in the real world if it is designed correctly on paper.

Hardware systems hardly ever have a single locus of control. Because each component in hardware acts independently subject to its interaction with other components, hardware systems are naturally concurrent, and in fact concurrency is often exploited in hardware to great advantage. Since hardware systems are concurrent, the languages for specifying hardware should be able to represent concurrency naturally if hardware specification in them is to be natural. This insight into the requirement of languages for hardware design should be useful in guiding advances in this direction. Coordination nets and modular asynchronous coordination structures represent a step in this direction.

Another important requirement of hardware design is that one should be able to construct a larger system by interconnecting smaller systems. Therefore it is of particular interest to know what behavior will emerge when a number of systems are interconnected. In general, behavior of both determinate and non-determinate system is important. Many a time, however, one wishes to construct a larger determinate (functional) system by interconnecting a number of smaller determinate systems. The ideal case would be that in which any interconnection of a number of determinate systems resulted in a determinate system. However, this is not true in general, i.e., an interconnection of a number of determinate systems need not result in a determinate system. Some recent work of the author [21] states constraints on interconnecting links that are both necessary and sufficient for the interconnection of a number of determinate systems to be determinate. This is a step towards understanding the influence of interaction on the behavior of a system of interconnected systems.

7.5 Related Theories

In order to be able to treat interaction of systems in greater depth, a theory for system behavior is needed. This theory should be to computer systems what Linear System Theory is to feedback systems. In particular the theory should provide effective means for deriving the behavior of a system of interconnected systems from the behavior of individual systems. Since not all useful systems are determinate

systems, the theory must concern itself with non-determinate systems as well. Moreover, as behavior of computer systems (an important class of systems that would be handled by the theory) involves flow of information into and out of the systems, the theory would be intimately connected with a theory of information that deals with logic and the flow of information. A theory of information, a theory of system behavior and schemata for representation of systems are all related, and an advance in any of these is an advance in the others. In this respect the work of Holt and his assoicates [8,22] is significant.

The work on concurrent systems should find application in modelling computer systems and also in other areas. For example better models for business systems could be constructed using these ideas as business systems are inherently concurrent systems involving interactions among several otherwise independent activities. Similarly biological systems like nervous systems could be better modelled as they too are concurrent systems. At a later stage, it should be possible to analyse such systems and suggest the amount and the kind of coordination necessary to ensure their proper functioning. The coordination nets presented in this thesis are not probablistic in nature, but they can be easily extended to probablistic models by associating probabilities with transitions just as a Markov model is obtained by associating probability measures with transitions in finite state machines.

APPENDIX I

Hardware Implementation of Modules

This thesis does not deal with hardware implementation of the modules, but a discussion of this topic is desirable. There are two approaches to design of modules: i) synchronous design, and ii) asynchronous design. In the first type of design, even though a module interacts with other modules in an asynchronous manner, it is synchronous inside. In this case, the module samples its inputs at regular intervals and, depending on the outcome of sampling, produces outputs using a combinational circuit. This type of implementation is used in the design of some parts of present day computers, but modules so designed are slow and lack elegance. In the second type of design, a module is designed as an asynchronous circuit. The conventional switching circuit theory is not of much use in the design of such asynchronous circuits, but the author believes that an elegant method for designing such modules in terms of elementary circuits, called micro-modules, can be worked out. This approach to the design of modules is introduced below.

Links and Wires

The basic links (see section 4.4) can be constructed from wires as shown in figure 8.1 . Ready wires are used to send ready signals only, and acknowledge wires to send acknowledge signals only. A change

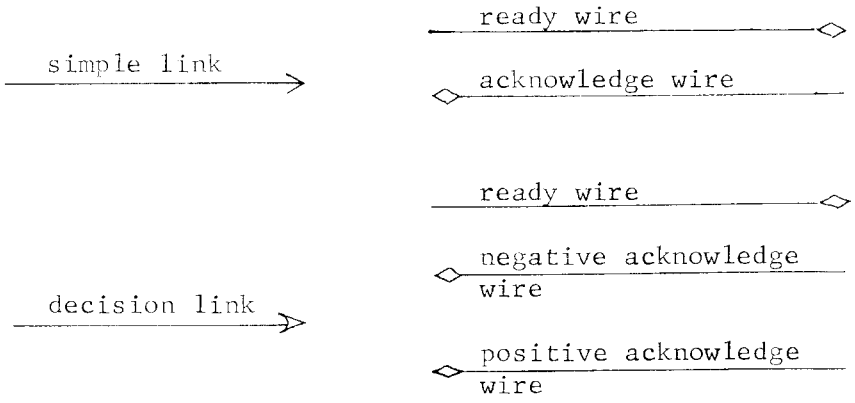


Figure 8.1 Links and Wires

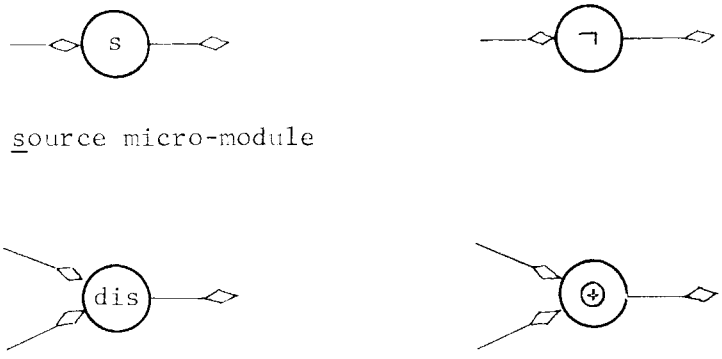


Figure 8.2 The source and disjunct Micro-modules

in the level of a wire, as opposed to the level itself, represents a signal. A signal on a ready wire is called a ready signal and a signal on an acknowledge wire an acknowledge signal; ready and acknowledge signals are of the same kind - the role a signal plays in communication over a link determines whether it is a ready signal or an acknowledge signal.

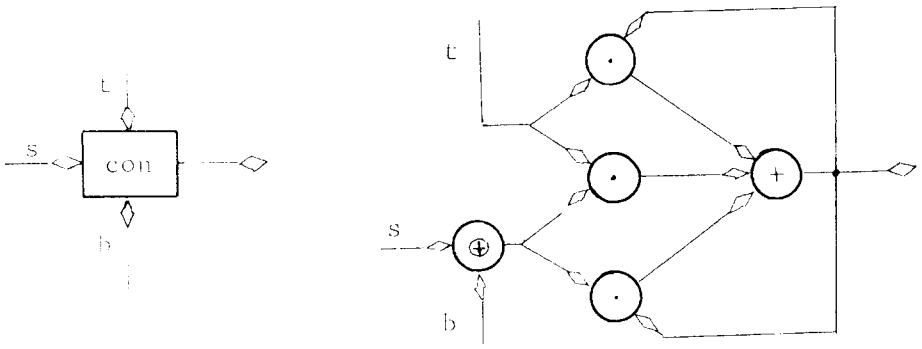
Micro-modules

Micro-modules are elementary logic circuits for constructing bigger modules. In this discussion four micro-modules are introduced (figures 8.2 and 8.3). The operation of these micro-modules is explained below.

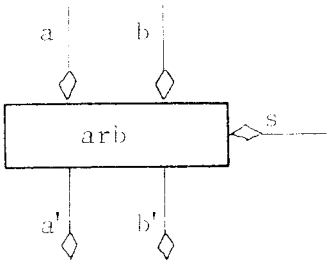
A NOT gate is called a source micro-module because following initialization of the circuit (a momentary forced grounding of all wires), the output wire of the NOT gate goes from 0 to 1 , and thus sends a signal.

An Exclusive-OR gate is called a disjunct micro-module as it transmits a signal on the output wire for each signal received on either of the input wires (in use of this micro-module it is necessary to ensure that both input wires do not receive signals coincidentally).

The circuit shown in figure 8.2c is called a conjunct micro-module. Signals are sent to this micro-module in pairs - a signal on wire s and on wire t (in either order or coincidentally) or on s and b (s, t and b stand for signal, transmit and block). In case of the first pair, a signal is sent on the output wire when both signals are received, and in case of the second pair the micro-module just absorbs the signals.



a) conjunct micro-module



b) arbiter micro-module

Figure 8.3 The conjunt and arbiter Micro-modules

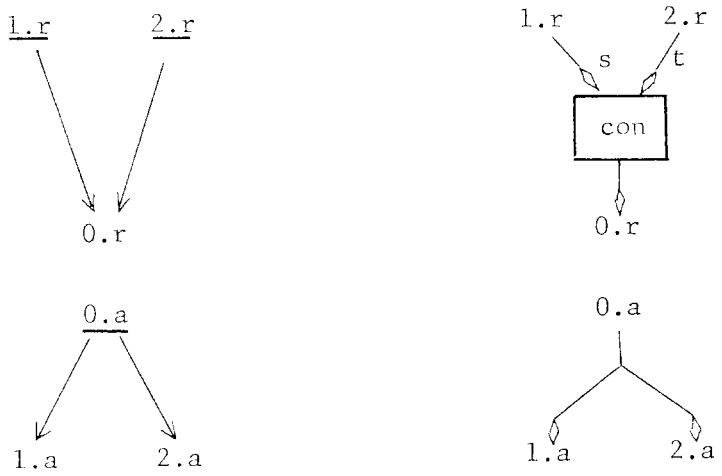
Figure 8.3 shows the schematic of the arbiter micro-module. This micro-module is an elementary two input arbiter. The micro-module can be viewed as a gate in the path of the wires a-a' and b-b' . Signals may be sent to the micro-module on both input wires a and b coincidentally. The micro-module permits one signal from input a or from b (but not both) to pass through it for each signal received on the third input wire s . Which one of the two signals is permitted to pass through the module is determined by the module on a first come first serve basis except when the signals are coincident, in which case the choice is arbitrary. Even though arbiters have been designed and are used in present day computers, they are slow and inefficient. Improved design of arbiters is currently in progress [23].

Construction of Asynchronous Modules from Micro-modules

Examples showing the use of micro-modules in construction of asynchronous modules are presented in figures 8.4 and 8.5 .

Figure 8.4a shows a P-net for the junction module obtained from the P-net of figure 4.25 by removing certain redundancies taking into account the relationship among signals on the links. A micro-modular circuit for the junction modules obtained from the P-net for the module is shown in figures 8.4b and 8.4c .

Figure 8.5a shows a P-net for the IP-module obtained from the P-net specification of the module (figure 4.13). The micro-modular circuit for the IP-module obtained from the P-net for the module is shown in figure 4.5b .



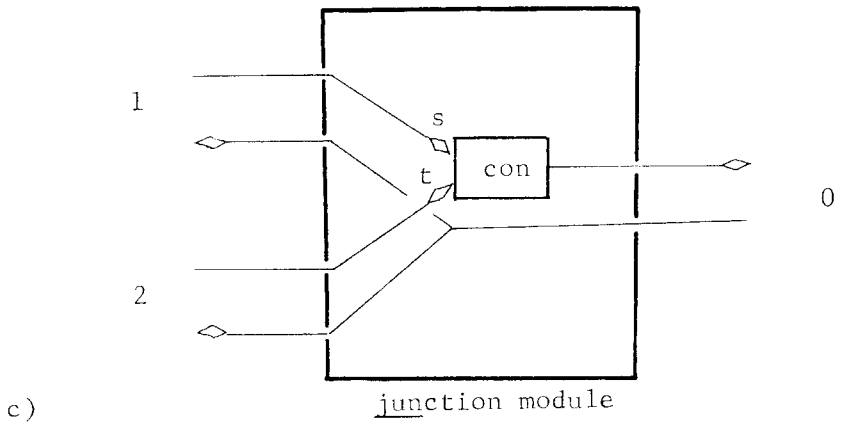


Figure 8.4 Implementation of the junction Module

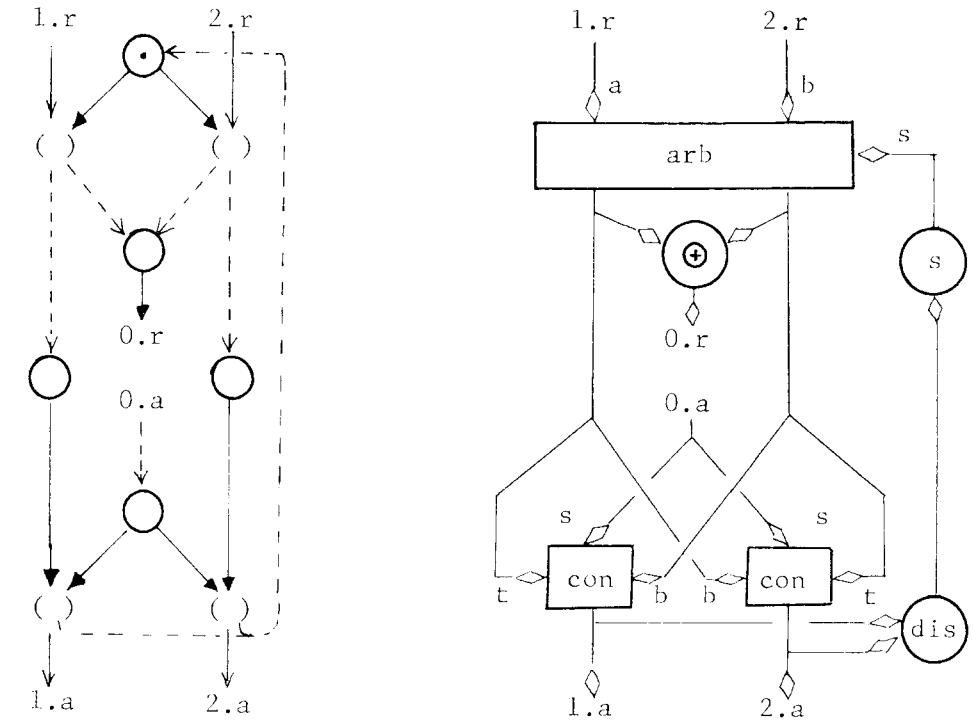


Figure 8.5 Implementation of the IP-module

BIBLIOGRAPHY

1. Corbato, F. J. and Vyssotsky, V. A., "Introduction and Overview of the Multics System," FJCC 1965.
2. Dijkstra, E. W., "Cooperating Sequential Processes," Math. Dept. Technological University, Eindhoven (1965).
3. Dennis, J. B. and Van Horn, E. C., "Programing Semantics for Multiprogrammed Computations," Comm. ACM 9, March 1966, pp 143-155.
4. Karp, R. M. and Miller, R. E., "Parallel Program Schemata: A Mathematical Model for Parallel Computations," IEEE Conference Record of the Eighth Annual Symposium on Switching and Automata Theory, October, 1967, pp 55-61.
5. Slutz, D. R., "A Flow Graph Schemata Model of Parallel Computation," Doctoral Thesis, M.I.T., September 1968.
6. Luconi, F. L., "Asynchronous Computational Structures," Doctoral Thesis, M.I.T., January, 1968.
7. Petri, C. A., "Kommunikation mit Automaten," Schriften des Reinisch-West falischen Inst. Instrumentelle Math., and der Universitat Bonn, Nr 2., Bonn, 1962.
8. Holt, A. W., Shapiro, R. M., Saint, H. and Warshall, S., "Final Report for the Information System Theory Project," Applied Data Research, Inc., February 1968, ADR Ref. no. 6608
9. Muller, David E., "Asynchronous Logic and Application to Information Processing," Switching Theory in Space Technology, Stanford University Press, 1963, pp 289-293.

10. Muller, David E. and Bartky, W. S., "A Theory of Asynchronous Circuits," Proceedings of an International Symposium on the Theory of Switching, The Annals of the Computation Laboratory of Harvard University, Volume 29, pp 204-243, Harvard University Press, Cambridge, Mass., 1959.

11. Clark, Wesley A., et. al., "Macromodules," - a set of papers, AFIPS Conference Proceedings (SJCC), Volume 30, pp 335-401, 1967.

12. Shapiro, Robert M. and Saint, H., "The Representation of Algorithms," Applied Data Research, Inc., RADC-TR-69-313, Volume II, Final Technical Report, September 1969.

13. Patil, Suhas S., "Macro-modular Design of Asynchronous Circuits," Project MAC, Computation Structures Group Memo No. 41, May 1969.

14. Patil, Suhas S., "n-server m-user Arbiters," Project MAC, Computation Structures Group Memo No. 42, May 1969.

15. Patil, Suhas S., "A Micro-modular Implementation of the Control Modules of Basic Macro-modular Circuits," Project MAC, Computation Structures Group Memo No. 43, October 1969.

16. Benes, V. E., Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, 1965.

17. Habermann, A. N., "On the Harmonious Cooperation of Abstract Machines," Ph.D. Dissertation, Technological University of Eindhoven, 1967.

18. Shoshani, A. and Coffman, E. G., "Sequencing Tasks in Multi-process, Multiple Resource Systems to Avoid Deadlocks," Technical Report 78, Computer Science Laboratory, Princeton University, June 1969.

19. Hebalkar, Prakash G., "Deadlock-free Sharing of Resources in Asynchronous Systems," forthcoming Ph.D. Dissertation, Department of Electrical Engineering, M.I.T.

20. Rogers, Hartley, Theory of Recursive Functions and Effective Computability, McGraw-Hill Book Company, 1967, page 20.

21. Patil, Suhas S., "Communication Requirement for Determinacy of a System of Determinate Systems," Project MAC, Computation Structures Group Memo No. 44, November 1969.

22. Holt, Anatol W. and Commoner, Fred, "Information Flow in State Machines,"" Applied Data Research, Inc., March 1970.

23. Stollac, Jolles, "Implementation of the Elementary Arbiter Module," forthcoming S.B. Thesis, Department of Electrical Engineering, M.I.T.

INDEX OF NOTATION

SUBJECT INDEX

BIOGRAPHICAL NOTE

Suhas Shrikrishna Patil was born in Jamshedpur, India on June 23, 1944. He graduated from Mrs. K. M. P. M. High School, Jamshedpur in 1959, and from St. Xavier's College, Calcutta with Intermediate Science in 1961. He received a Bachelor of Technology (Hons) degree from Indian Institute of Technology in 1965. He was the president of the Electronics and Electrical Communication Engineering Society at Indian Institute of Technology, Kharagpur from 1964 to 1965. While ar I.I.T., he was the chairman of the gardening committee of R.K. Hall of Residence, received awards for electronic modelling and for the best undergraduate project in 1965.

He began studies at Massachusetts Institute of Technology in 1965, where he received the degree of Master of Science in 1967. He became a member of the staff of the Electrical Engineering Department in 1965 as a Teaching Assistant in 1965, and as an Instructor in 1968. He has been associated with Project MAC since 1966. During summer of 1967 he worked at T. J. Watson Research Center, I.B.M..

He has joined the faculty of Massachusetts Institute of Technology as Assistant Professor of Electrical Engineering.

This empty page was substituted for a blank page in the original document.

CS-TR Scanning Project
Document Control Form

Date : 2/8/96

Report # LCS-TR-72

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 240 (244 IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☒ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): _____

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description : Page Number:

IMAGE MAP: (1-240) UNT'ED TITLE, BLANK, ABSTRACK, BLANK, ACKNOWLEDGEMENT, DEDICATION, FUNDING AGENT, BLANK, TABLE OF CONTENTS, 6-7, UNT'ED BLANK, 9-234, UNT BIB NOTES, UNT BLANK (241-244) SCANCONTROL, TARGET'S(3)

Scanning Agent Signoff:
Date Received: 2/8/96 Date Scanned: 2/9/96 Date Returned: 2/15/96

Scanning Agent Signature: Michael N. Cobb

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**