INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

University Microfilms International
300 North Zeeb Road
Ann Arbor, Michigan 48106 USA
St. John's Road, Tyler's Green
High Wycombe, Bucks, England HP10 8HR

77-18,152

ZERVOS, Cristian Radu, 1949-
COLORED PETRI NETS: THEIR PROPERTIES AND APPLICATIONS.

The University of Michigan, Ph.D., 1977
Computer Science

Xerox University Microfilms, Ann Arbor, Michigan 48106

COLORED PETRI NETS:

THEIR PROPERTIES AND APPLICATIONS

by

Cristian Radu Zervos

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer, Information and Control Engineering)
in The University of Michigan
1977

Doctoral Committee:

Professor Keki B. Irani, Chairman
Associate Professor Daniel E. Atkins
Professor John F. Meyer
Assistant Professor William E. Riddle
Associate Professor William C. Rounds

ACKNOWLEDGEMENTS

I would like to thank the members of my doctoral committee, Professors Atkins, Meyer, Riddle and Rounds, for their interest and cooperation. I am especially grateful to my thesis advisor, Professor Keki B. Irani, for his guidance, encouragement and support not only throughout the course of this research but also throughout the entire period of my graduate studies.

Thanks are due to Karen Chapin, Cheryl Peck and Gail Mann who typed parts of the manuscript. I am especially indebted to Annette Sizemore who helped in many ways with the preparation of the final version of this report.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS ii

LIST OF FIGURES v

LIST OF APPENDICES x

LIST OF COMMON ABBREVIATIONS xi

CHAPTER

I. INTRODUCTION 1

1.1 Motivation 1
1.2 Overview of the Thesis 15

II. THE COLORED PETRI MODEL 18

2.1 Introduction and Preliminary Definitions 18
2.2 The Colored Petri Model 22
2.3 Execution Rules for the Colored Petri Model . . . 25
2.4 The C-Colored Petri Model 32
2.5 The EC-Colored Petri Model 40

III. THE REPRESENTATION CAPABILITIES OF THE EC-COLORED PETRI MODEL 54

3.1 Introductory Remarks 54
3.2 Composition Properties of the EC-CPM 58
3.3 Closure Properties of Λ_o(EC-CPM) Under Mappings . 86
3.4 The Extents of the Language Families Λ(EC-CPM) and Λ_o(EC-CPM). 98

IV. THE REPRESENTATION CAPABILITIES OF THE C-COLORED PETRI MODEL 103

4.1 Introduction 103
4.2 Closure under k-Limited Erasing of Λ(EPM) and Λ_o(EPM) 107
4.3 The Priority Petri Model (I) 125
4.4 The C-Colored Petri Model (I) 138
4.5 The Relationship Between the C-CPM and the PPM. . 157
4.6 Coordination Nets Revisited 183
4.7 A Result Concerning the Language Families Λ(C-CPM) and Λ_o(C-CPM). 204
4.8 The Relationship between the C-CPM and the EC-CPM 209

TABLE OF CONTENTS (Continued)

V. MODELLING WITH THE C-CPM AND THE EC-CPM 230

5.1 Modelling with the C-CPM 230
5.2 A Producer-Consumer Synchronization Problem with Dynamic Priority Hierarchy 231
5.3 A Process Coordination Problem with Conflict. . . 238
5.4 Modelling Reentrancy 245
5.5 Modelling with the EC-CPM 250
5.6 A Producer-Consumer Synchronization Problem with LIFO Structured Buffer, Revisited 251
5.7 Modelling of the Flow of Control in Recursive Subroutines 255
5.8 A Producer-Consumer Synchronization Problem with Generalized Queueing Mechanism 273

VI. CONCLUSIONS 279

6.1 Overview of the Research 279

APPENDICES 282

BIBLIOGRAPHY 315

LIST OF FIGURES

Figure		Page
1.1.1	Incorrect Modelling of a Reentrant Subroutine by the Complex GMC	4
1.1.2	"Copy" Method for the Modelling of Reentrant Subroutines	5
1.1.3	Meta-description of a Recursive Routine which Computes the Factorial Function	8
1.1.4	Execution of the Recursive Routine of Figure 1.1.3	9
2.1.1	Example of a Generalized Petri Net	20
2.4.1	Schematical Representation of a Producer-Consumer Synchronization Problem	33
2.4.2	C-CPM of the Producer-Consumer Synchronization Problem of Figure 2.4.1	33
2.4.3	Solution to the Readers-Writers Synchronization Problem Using Semaphores	37
2.4.4	C-CPM Representation of the Readers-Writers Synchronization Problem	38
2.5.1	Schematical Representation of a Producer-Consumer Synchronization Problem	41
2.5.2	EC-CPM Representation of the Producer-Consumer Synchronization Problem of Figure 2.5.1	41
2.5.3	Example of the Extension of a Color Set C	45
2.5.4	k-Accessor Classes Synchronization Problem	52
3.2.1	Sample Labelled EC-CPM	60
3.2.2	Sample Labelled EC-CPM	60
3.2.3	Juxtaposition of the Labelled EC-CPM's of Figure 3.2.1 and 3.2.2	61
3.2.4	The Introduction of a Control Place.	61
3.2.5	Construct for the Union Operation	66

Figure		Page
3.2.6	Construct for the Operation of Concatenation	70
3.2.7	Sample Transitions	72
3.2.8	The Composition of the Transitions t'_k and t''_r	73
3.2.9	Sample Labelled EC-CPM	73
3.2.10	Construct for the Intersection Operation	74
3.3.1	Sample Labelled EC-CPM Generating a Context-Free Language	89
3.4.1	Sample Labelled EC-CPM	100
4.1.1	Sample Extended Petri Net	106
4.2.1	Sample EPN(I)	109
4.2.2	Sample EPM(I)	117
4.2.3	One-Place Closed Subnets of the EPM(I) of Figure 4.2.2	117
4.2.4	EPM's Obtained from the One-Place Closed Subnets of Figure 4.2.3	118
4.2.5	EPM Obtained from the Sample EPM(I) of Figure 4.2.2	119
4.3.1	Sample PPN(I)	133
4.3.2	EPN(I) Obtained from the PPN(I) of Figure 4.3.1	134
4.3.3	Sample PPN	135
4.3.4	Sample PPN(I)	135
4.3.5	PPN(I) Obtained from the PPN of Figure 4.3.3	136
4.4.1	Sample C-CPN(I)	141
4.4.2	Typical Transition of a C-CPN(I)	144
4.4.3	Transition of a PPN(I) Obtained from the Transition of Figure 4.4.2	144
4.4.4	PPN(I) Obtained from the C-CPN(I) of Figure 4.4.1	147
4.4.5	Sample EPN	155

Figure		Page
4.4.6	C-CPN Obtained from the EPN of Figure 4.4.5	156
4.5.1	The Concatenation of Two Finite Lattices	158
4.5.2	The Set of Colors $C_I = (X_I, \leq_I)$	160
4.5.3	Simulation of the Enabling Color Selection Function of the Arc a^r_{jk}	163
4.5.4	Sample C-CPN.	166
4.5.5	Sample Replacement Transitions.	167
4.5.6	Sample Clean-Up Subnet	172
4.5.7	Simulation of the Firing of a Transition by CN'	173
4.5.8	Sample Detection Subnet	178
4.6.1	Sample COPM	184
4.6.2	Sample Coordination Petri Model	190
4.6.3	EPM(I) Obtained from the COPM of Figure 4.6.2	190
4.6.4	Sample Labelled EPM	191
4.6.5	Language Equivalent Labelled COPM	191
4.6.6	Replacement Transitions of $t_k \in T^z$	195
4.6.7	Replacement Transitions of $t_m \in T-T^z$	196
4.6.8	Replacement Transitions of "Stop" Transitions	200
4.7.1	Sample EPN(II)	205
4.8.1	Sample Transition of an EPM	211
4.8.2	Replacement Transition for the Transition of Figure 4.8.1	211
4.8.3	Relationship of Λ_o(C-CPM) and Λ(C-CPM) to Other Language Families	214
4.8.4	Sample Transition	217
4.8.5	Sample Transition	217

Figure		Page
4.8.6	Sample Transition	217
4.8.7	Concatenation Subnet	219
4.8.8	Deletion Subnet	221
4.8.9	Generation of a Context-Free Language	222
4.8.10	Sample Transition	226
5.2.1	Producer-Consumer Synchronization System with Dynamic Priority Hierarchy	233
5.2.2	C-CPM of the Producer-Consumer Synchronization System of Figure 5.2.1	233
5.2.3	Buffer Contents for the Producer-Consumer Synchronization System of Figure 5.2.1	236
5.2.4	PPM of the Producer-Consumer Synchronization System of Figure 5.2.1	237
5.3.1	Process Coordination System with Conflict	240
5.3.2	C-CPM Representation of the Coordination System of Figure 5.3.1	241
5.3.3	Coordination Policy for the Process Coordination System of Figure 5.3.1	243
5.3.4	PPM of the Process Coordination System of Figure 5.3.1 (only one process P_i is connected to each buffer B_i, i = 1,2,3)	244
5.4.1	Modelling of a Sample Reentrant Subroutine	247
5.4.2	Nested Reentrant Subroutines	248
5.4.3	Modelling of Nested Reentrant Subroutines	249
5.6.1	Transfer of Requests for I/O Operations in a Multiprogrammed Computing System	254
5.7.1	Sample Flow Diagram	258
5.7.2	Representation of Subroutine Calls	261
5.7.3	Sample Recursive Subroutine	264

Figure		Page
5.7.4	Uninterpreted Flow Diagram of the Recursive Subroutine of Figure 5.7.3	265
5.7.5	Sample Labelled EC-CPM Generating a Context-Free Language	270
5.7.6	Sample Transition of a Labelled EC-CPM	270
5.7.7	Substitution Labelled EC-CPM	272
5.8.1	EC-CPM of a Producer-Consumer Synchronization System with Generalized Queueing Mechanism	275
D.1	Sample Transition	303
D.2	Sample Transitions	304
D.3	Operation of a Switch	306
D.4	The Firing of a Transition without a Switch as Input Place	307

LIST OF APPENDICES

		Page
APPENDIX A:	Bag Notations and Definitions	282
APPENDIX B:	Tape and Time Complexity of the Language Families Λ(EC-CPM) and Λ_o(EC-CPM)	285
APPENDIX C:	Multi-Counter E-Acceptors	297
APPENDIX D:	The Petri Net Model with Switches Disjunctive Logic and Token Absorbers	308

LIST OF COMMON ABBREVIATIONS

$\|S\|$	Cardinality of the set S
$\mathcal{D}(B)$	Domain of the bag B
$\#(b,B)$	Number of occurrences of the element b in the bag B
$< S >^*$	Bag closure of the set S
Z	Set of integers
Z^o	Set of nonnegative integers
Z^+	Set of positive integers
T	Set of transitions
P	Set of places
I	Input incidence function
O	Output incidence function
I_k	Bag of input places of the transition t_k
O_k	Bag of output places of the transition t_k
M^i	Marking
$M^i(p_j)$	Number of tokens in the place p_j in the marking M^i (marking of the place p_j)
$C = (X,\leq)$	Set of colors
$U(C)$	Extension of the finite color set C
CM^i	Color marking
UM^i	Color marking of an EC-CPM
C^i_j	Color bag of the place p_j in the color marking CM^i
U^i_j	Color bag of the place p_j of an EC-CPM in the color marking UM^i
θ^i_{jk}	Bag of enabling colors of transition t_k from the input place p_j in the color marking CM^i (UM^i)
θ^i_k	Bag of enabling colors of transition t_k in the color marking CM^i (UM^i)
E^i	Set of transitions enabled in the color marking CM^i (UM^i)
$\|\gamma\|$	Length of the firing sequence γ
$S(CM^o)$ $(S(UM^o))$	Set of firing sequences
$T(CM^o,CM^f)$ $(T(UM^o,UM^f))$	Set of terminal firing sequences
$R(CM^o)$ $(R(UM^o))$	Set of color markings reachable from CM^o (UM^o)
$E(CM^o)$ $(E(UM^o))$	Set of active color markings reachable from CM^o (UM^o)
$CM^i_1 \nabla CM^r_2$ $(UM^i_1 \nabla UM^r_2)$	Direct sum color marking of CM^i_1 and CM^r_2 (UM^i_1 and UM^r_2)

$R_1(CM_1^o) \nabla R_2(CM_2^o)$ $(R_1(UM_1^o) \nabla R_2(UM_2^o))$	Set of all reachable direct sum color markings

CHAPTER I

INTRODUCTION

Section 1.1 MOTIVATION

Over the past fifteen years a considerable research effort has been focused by computer scientists on the problems of concurrent computing. Many different aspects of concurrent computing, ranging from the design and implementation of hardware structures capable of performing computations in parallel to the control and coordination of such computing systems, have been under study. One of the effects of this research work has been the development of several distinct abstract models of concurrent systems. The general goal of the development of formal models is to obtain a clear, simple but nonetheless valid abstract representation of systems exhibiting concurrency, based on which certain analysis or synthesis questions regarding the modelled real-life systems may eventually be easier to answer. Recent works ([PETE-73], [AGAR-75]) have shown that despite the fact that the various models of concurrent systems use different approaches in their way of representing the processes which can be encountered in a concurrent processing environment, they have a great deal in common. It is apparent that earlier models were concerned mainly with the representation and analysis of the sequencing of the various concurrent and sequential computational steps of a program, in other words with the study of the control flow in parallel programs. More recent studies have considered the modelling of collections of programs which interact with each other in a competitive as well as cooperative manner while executing simultaneously. Gostelow ([GOST-71]) and Cerf ([CERF-72]) have attempted to model well-known programming structures such as subroutines, reentrant subroutines and recursive routines using the UCLA Graph Model of Computation (GMC). The GMC is formed by a bilogic directed graph in which the vertices represent computational steps and the directed arcs which connect the vertices to each other represent precedence relations existing between the various computational steps. In [GOST-71] and [CERF-72] the GMC is extended to the so-called "Complex GMC" by the introduction of complex arcs, that is directed arcs

which can have multiple source vertices and multiple destination vertices. A certain precedence condition is said to hold if there are tokens present on the corresponding arc of the Complex GMC. Each node is associated with a fixed input and output logic which may be of the AND (*) or OR (+) type. A node with AND input logic may begin its execution if all its input arcs have tokens present on them. If the input logic of a node is of the OR type, it may start its execution if any of its input arcs contains tokens. A node is executed by removing tokens from its input arcs according to the vertex input logic and by placing tokens on its output arcs according to its output logic. If a node has AND type output logic then all its output arcs receive tokens at the termination of the vertex execution. On the other hand, if the output logic is of the OR type then any one of the vertex output arcs can receive tokens upon vertex execution termination. The number of tokens removed and/or placed by a node on a complex arc is given by the degree of the respective arc.

Gostelow and Cerf were mainly interested in the analysis of the control flow in parallel programs and therefore their studies use the Complex GMC as an uninterpreted model of parallel computation.

In [GOST-71] it is shown that the introduction of complex arcs in the GMC permits the correct modelling of subroutine calls in the context of parallel programs, which would not have been possible if only simple arcs were used. Complex arcs are credited with the capability of correctly solving the problem of return of control from a subroutine model which may be called from several distinct call vertices, without violating the assumption that the model is uninterpreted. Both Gostelow and Cerf, however, recognize that reentrant subprograms cannot be represented by the Complex GMC as neatly as ordinary subroutines.

In Gostelow's view "reentrancy is that property of programs which allows a single program to be entered and to execute as a program, while other independent entries and executions of the same program are in progress." Reentrancy is thus a special property of programs which not every program can have, namely that more than one independent control flow streams operate simultaneously within the same program. Since the Complex GMC is a model of control flow independent of inter-

pretation it should be able to represent the distinct control flow streams acting in a reentrant subroutine only through a proper arrangement of the arcs and vertices of the model, without relying on extra information provided by some specific interpretation of the vertex functions. The Complex GMC, as it is currently defined, cannot accomplish this task. Figure 1.1.1 exhibits an attempt to model a reentrant subroutine using the Complex GMC, as given in [CERF-72]. The reentrant subroutine formed by vertices 4 and 5 can be called concurrently from vertices 2 and 3, respectively. If a token appears on arc D, it is absorbed by vertex 5 which in turn places one token on arc R. Since both arcs A and B may contain tokens simultaneously, the token on arc R can be removed by either vertex 6 or 7. The choice of the vertex which absorbs the token on arc R is completely arbitrary in this case, according to the execution rules of the GMC. Therefore, vertex 5 may fail to return control precisely to the calling routine which corresponds to the control flow stream which has just terminated the execution (one of the two simultaneous executions) of the reentrant subroutine. The problem stems from the fact that the occurrence of a token on arc D does not provide vertex 5 with enough information regarding which of the two control flow streams operating concurrently in the reentrant subroutine does that token represent; that is to which call of the reentrant subroutine (from vertex 2 or 3) does the respective token belong.

This difficulty in modelling reentrant control structures is not particular only to Complex GMC's. In fact, based on the results obtained in [PETE-73], [AGAR-75], [MILL-74], where the modelling power of the significant models of parallel computation such as Petri Nets, the Complex GMC, Parallel Program Schema (Finite State), Program Graph Model, Vector Addition Systems, etc. was compared, it appears that the modelling difficulty mentioned above extends to all these formal models as well.

Gostelow suggests the following solution to the problem of representing reentrant programs. Reentrant subroutines are a priori identified and distinct copies are made for each instance of reentrancy, that is each call vertex which can initiate a reentrant execution of the sub-

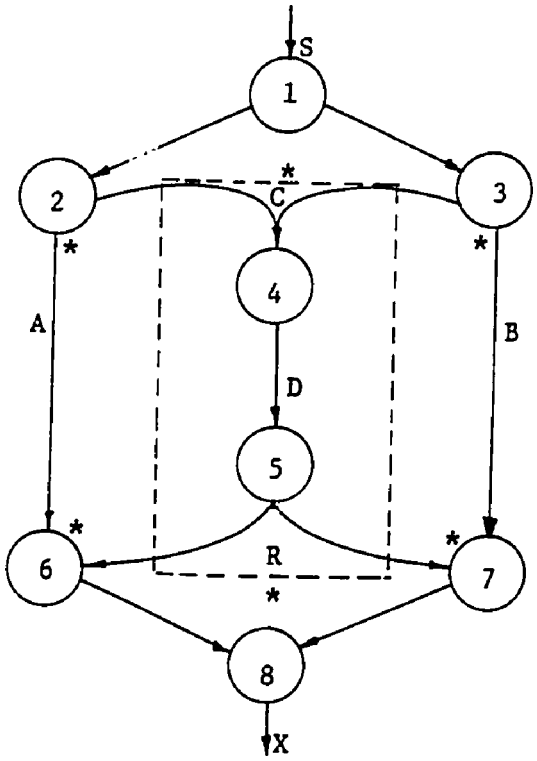


Figure 1.1.1

Incorrect Modelling of a Reentrant Subroutine by the Complex GMC

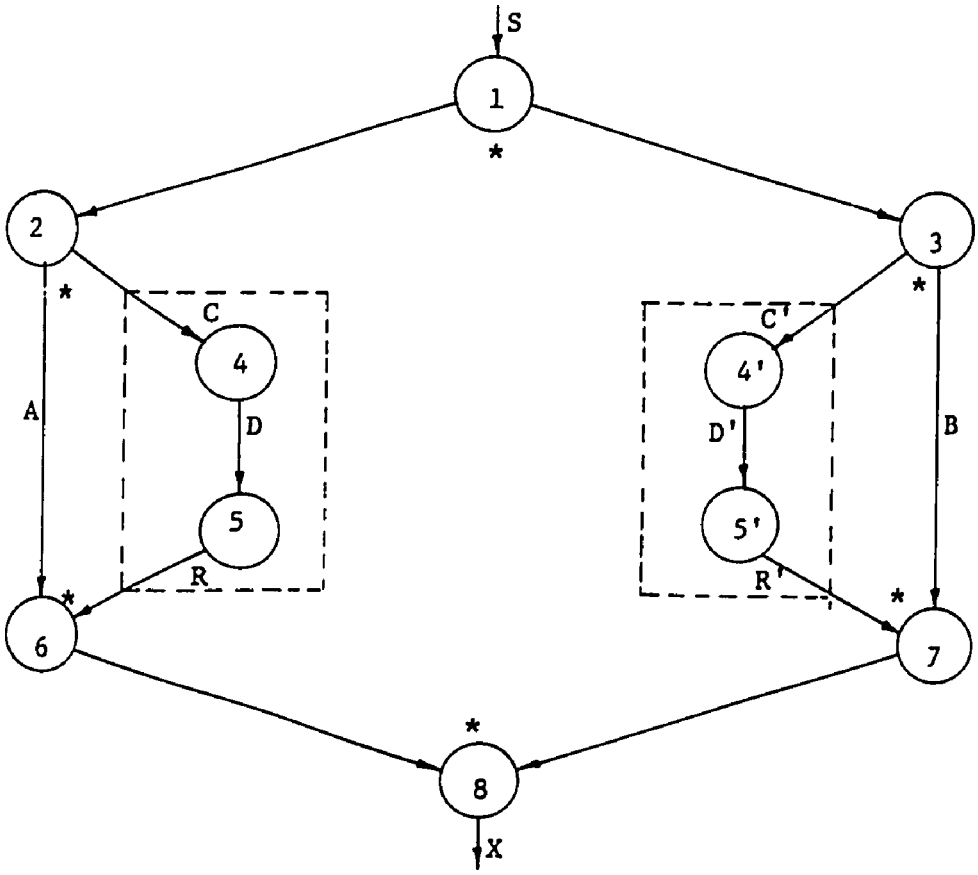


Figure 1.1.2

"Copy" Method for the Modelling of Reentrant Subroutines

routine in question is linked to a separate, distinct copy of the graph model of that subroutine. Since the number of call vertices is bounded, the resulting graph model will remain finite, as required. Figure 1.1.2 exhibits the method given above as applied to the reentrant subroutine of Figure 1.1.1.

The main drawback of this solution is that it increases the size of the respective graph model at an undesirable rate, especially if subroutine calls are nested. Analysis of such a model would be at the best difficult, especially if one is interested in modelling real-life collections of programs such as reentrant portions of compilers or reentrant operating system routines.

On the other hand, models of concurrent systems such as the Petri Net Model for example, do not have to be viewed only as formal representations of programs, but more generally, as formal counterparts of arbitrary asynchronous, concurrent structures. Petri Nets were originally defined to represent relationships between events and conditions. Instances of events are represented in the model by transitions and instances of conditions are represented by places. Graphically, a Petri Net is formed by a bipartite directed graph in which transitions are represented by bars and places by circles. Directed arcs connect transitions to places and places to transitions. Tokens are placed in the circles to represent the holding of the corresponding condition. As mentioned above, Petri Nets can be used to represent formally coordination structures of asynchronous concurrent systems. If we are seeking a physical implementation of a certain coordination structure modelled theoretically, such as the hardware realization of Petri Nets suggested in [PATI-70], then the number of places and transitions used in the model is of primary importance. Such general coordination structures may also contain reentrant substructures and Gostelow's method of representing reentrancy becomes inappropriate.

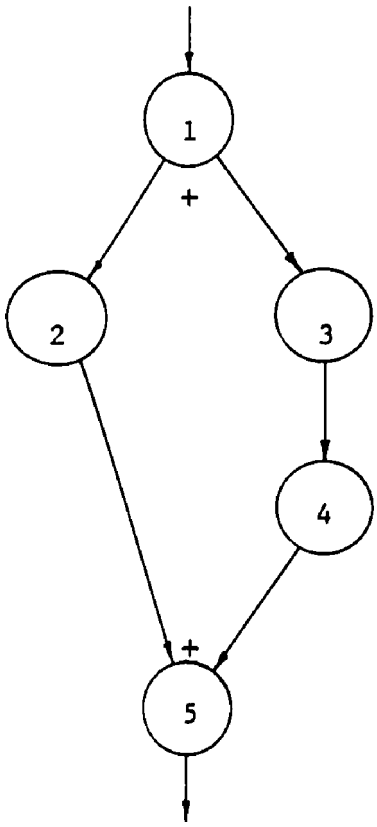
Another potential drawback is the fact that no interrelationship exists between the subroutine copies which have replaced the instances of reentrant calls. Let us consider a class of process coordination structures which inherently contain reentrancy, namely the "pipelined" coordination structures. In connection with pipelined synchronization

structures we would like to be able to model the interaction between the different control flow streams operating simultaneously within the collection of processes which form the pipeline. In particular, one would be interested in assigning different priorities to distinct control flow streams with respect to the activation of processes at certain points in the pipeline structure. This goal seems to be impossible to achieve through Gostelow's method of representing reentrant control structures.

As an alternate way of representing reentrancy which Gostelow and Cerf suggest is to allow colored tokens to identify which control stream it represents. Upon entry to a subroutine, the control token is painted with a color which identifies the call vertex and upon returning from the subroutine the token changes color again in the reverse order. This method would eliminate the need of distinct copies for distinct reentrant calls to a subroutine. Both Gostelow and Cerf, however, do not pursue this method further as it would imply changes to the original definition of their model.

As pointed out in [PETE-73], another assumption which limits the theoretical modelling power of most models of parallel computation is the assumption of a static system structure. This excludes all systems that can only be modelled by dynamic control structures, such as programs which use recursion, from the class of real systems which can be formally modelled.

Recursive programs raise even more complex modelling problems than reentrant programs. In [GOST-71] and [CERF-72] the following solution was adopted in connection with the Complex GMC. Recursive subroutines are represented by a "meta-description," that is whenever a call vertex to a recursive subroutine is executed, the processing of the graph model is stopped, the current state of the model is saved and the structure of the graph is redefined: a copy of the recursive routine is substituted for its dummy stand-in vertex. After the substitution of the subroutine copy is made, the execution of the graph model is restarted from the saved state. Figure 1.1.3 exhibits the meta-description of a recursive routine and Figure 1.1.4 displays the same routine after two levels of recursion have occurred. This example was originally




```
1:  if n=0 go to 2 else go to 3;
2:  F: = 1;
3:  F: = F(n-1);
4:  F: = n*F;
5:  return;
```

Figure 1.1.3

Meta-description of a Recursive Routine which Computes the Factorial Function

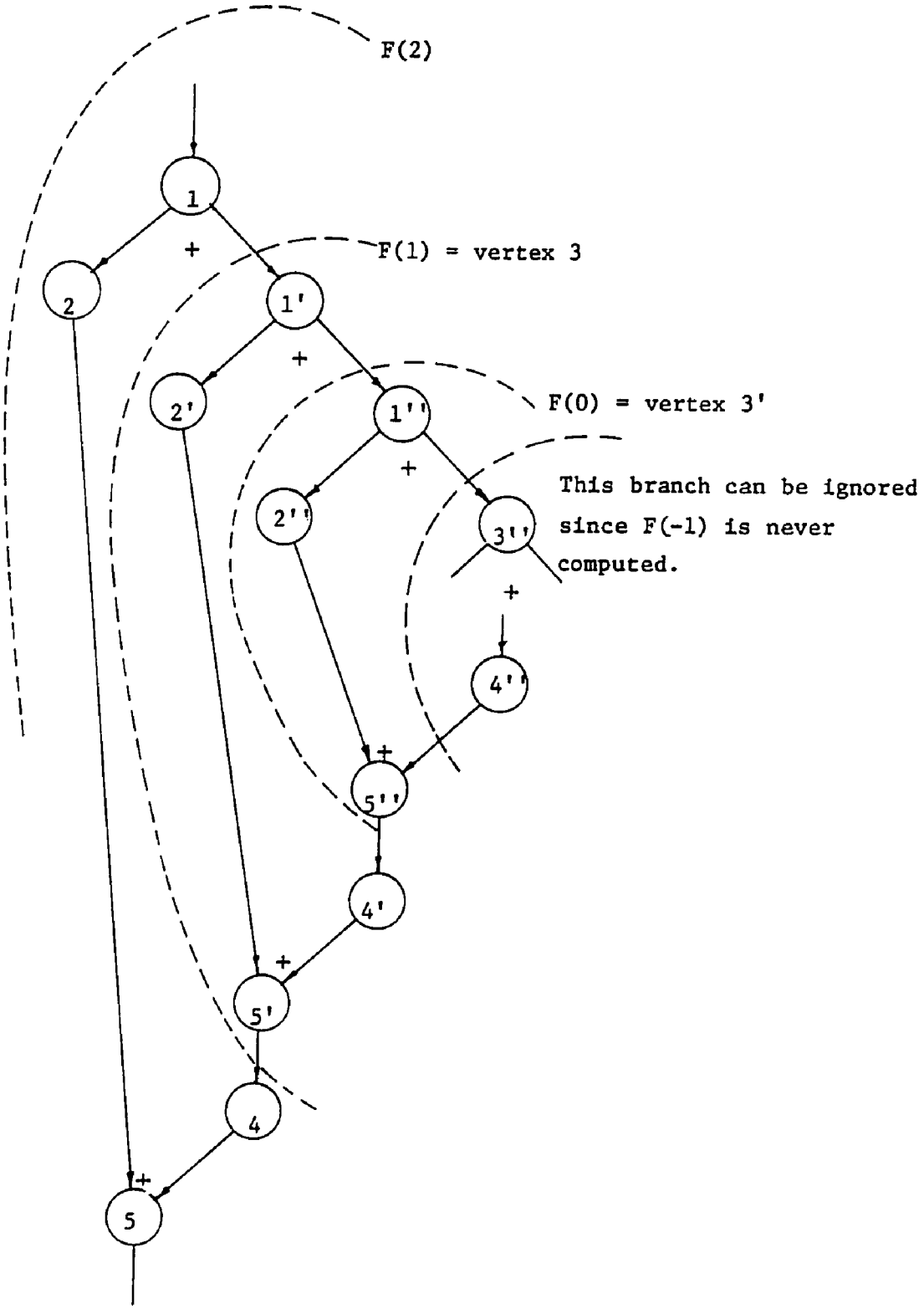


Figure 1.1.4

Execution of the Recursive Routine of Figure 1.1.3

presented in [CERF-72].

This method of handling recursive routines in the context of the Complex GMC is similar to the method used for reentrant programs: recursive routines are a priori detected and copies are substituted for each distinct call to a recursive routine. The main difference is that it is not possible to state a priori how many levels of recursion will occur during the execution of the model and thus how many copies of the recursive subroutine must be made. Therefore the structure of the graph model must be redefined dynamically, during the execution of the model. This rule certainly violates the assumption that the Complex GMC is an uninterpreted model of control flow in programs since vertices with special functions are used for all call vertices to the recursive subroutines, namely dummy vertices which merely stand-in for copies of the corresponding recursive subroutine. Secondly, the user is required to follow the execution of the graph model and to replace the dummy stand-in vertices of recursive subroutines by copies of the called subroutine whenever the execution of the graph model reaches such a call vertex. This meta-description of recursive subroutines can neither be correctly represented merely by a proper arrangement of the arcs and vertices of the graph model nor can it be supported by the current execution rules of the Complex GMC. As Cerf simply remarks "... there is no way with the current GMC to express recursion ... As a guess, the solution to this problem will require a major quantum change in the nature of the GMC." The analysis of recursive programs is then particularized in [CERF-72] where it is proven by induction that the Complex GMC meta-description of certain recursive functions is properly terminating (PT).

The difficulty in the representation of recursion discussed above in connection with the Complex GMC is not specific only to this model of parallel computation but seems to extend to all the other formal models mentioned earlier whose representation power is equivalent to or less than that of the Complex GMC. The drawbacks which we have exhibited relative to the "copy" method used for the representation of reentrant subroutines in the Complex GMC apply as well to the meta-description method suggested for the representation of recursion. One

should notice that the graph model of a recursive program increases in size at a very fast rate. The dynamic restructuring of the model required by the meta-description method makes a physical implementation such as that suggested in [PATI-70] of a recursive coordination structure represented formally by a Petri Net impossible. In fact, even more powerful versions of the Petri Net Model such as the Extended Petri Model ([AGAR-75]) or the Priority Petri Model ([HACK-75]) fail to offer a natural representation of reentrancy and recursion. This statement will be reexamined in Chapter V. As an alternate method, Gostelow suggests again the use of colored tokens in the representation of recursion but does not pursue this idea further.

So far we have discussed three types of control structures, namely the reentrant, recursive and pipelined control structures, and we have tried to emphasize the reasons for which the existing formal models of concurrent systems have difficulty in representing these forms of control. We have also hinted on an intuitive basis that the use of colors in the representation of flow of control may have a beneficial influence on the representation capabilities of formal models of concurrent systems. It appears that none of the existing models of concurrent systems which we have mentioned earlier can handle in their original definition "colored" representations of control flow. There are however in the literature pertinent to the field of formal models of parallel systems a few reports on attempts to use colors in the modelling of concurrent control structures.

The first formal model known to have indirectly used "colors" in the representation of control flow is the E-Net Model developed in [NUTT-72] and [NOE-73]. The E-Net Model is a modified version of the original Petri Net Model, created with the goal of having an interpreted model of parallel computer systems. E-Nets were meant to be used as aids in development of simulations and planning of measurements for examining characteristics of parallel computing systems such as throughput, turnaround time and utilization of resources. An E-Net can be constructed by using transitions from five primitive transition classes and places. Tokens may move along directed arcs in the net, being transferred by transitions from their input places to their

output places. It is interesting to mention that a transition cannot fire if its output places contain tokens. In this way it is ensured that any place contains at the most one token at any time during the execution of the net. It is also interesting to mention that the tokens used in an E-Net can be of two types, namely simple tokens such as those used in the ordinary Petri Net Model (their presence in a place of the net merely implies the holding of a certain condition) and attribute tokens. The attribute tokens could be regarded as "colored tokens" since each attribute token is associated with a vector representing a set of attributes. As an attribute token flows through the net, its attribute values may change as well as the attribute set itself. A special type of places, called resolution places, may reference the attributes of attribute tokens present in the various places of the net in order to evaluate certain predicates associated with the respective resolution places. The E-Net Model represents a formalism very similar to a simulation language such as GPSS for example, oriented, of course, towards the modelling of parallel computing systems. The attribute tokens resemble closely to the dynamic entities of the GPSS models, the so-called transactions.

Another model of parallel systems which makes explicit use of colored tokens is the CL-Net Model presented in [SONN-75]. There the CL-Nets were introduced as a modified version of the Petri Nets, offering a more compact representation of systems with concurrency and an increased representation power. The tokens present in a CL-Net carry one of the colors of a finite set of colors associated with the net. A partial ordering relation is defined on the set of colors such that the set of colors forms a lattice. Each transition of a CL-Net is also assigned a fixed color. Based on the colors of the tokens present in the input places of transitions and on the colors of the transitions, a hierarchy of firing priorities is established for the transitions of the net. This priority rule is used in solving conflicts among transitions. A transition has also the capacity of deciding the color of the tokens it will place in its output places upon firing. In this way the tokens can be repainted while flowing through the net.

The CL-Nets were introduced as an auxiliary tool, meant to be used

as a compact representation of certain parallel algorithms presented in [SONN-75]. Therefore, no in-depth analysis of the CL-Net Model was attempted there. We shall consider again the CL-Nets together with an interesting remark made in [SONN-75] about their representation power in Section 1.2.

In [DENN-75], based on the concept of data flow, a programming language for parallel processing environments is defined. The language is equivalent in expressive power to a block structured language with internal procedure variables and permits the concurrent execution of noninterfering program parts. The sequencing of the program steps is determined based solely on the availability of the input data for the particular computations.

A data flow program is graphically represented as a bipartite directed graph where the two types of nodes are called links and actors. Actors are connected to links and links are connected to actors by means of directed arcs. The arcs of a data flow program are regarded as channels through which tokens flow from each actor to other actors by way of the links.

Values are represented by a "heap", a finite, acyclic, directed graph having one or more root nodes, such that each node of the heap can be reached via some directed path from one of the roots. The nodes of the heap are of three types:

i. - elementary nodes - these nodes have no emanating branches and contain a value of the type boolean (true or false), integer, real or string.

ii. - structured nodes - these nodes contain pointers to the nodes placed at the ends of the branches emanating from the respective structured node and the values associated with the "successor nodes."

iii. - text nodes - these nodes contain the text of some data flow procedure as value.

The tokens flowing through a data flow program are "indirectly" colored in the sense that each token carries a pointer to some node of the heap structure. The actors perform operations on the values associated with the nodes of the heap pointed to by the tokens present on their input arcs. The links have only one input arc and possibly

several output arcs. Their purpose is to distribute identical tokens to actors. During the execution of a data flow program, nodes are appended and/or deleted from the heap.

A special type of actors, called "apply actors," operate on text-nodes. Each time such an actor is fired, the data flow procedure pointed to by the token which has caused the apply actor to fire begins execution. Since many concurrent activations of the same data flow procedure may exist due to concurrent or recursive procedure applications, colored tokens are used in order to avoid the possibility of confusing the tokens present in a data flow procedure on account of two or more distinct activations of that procedure. That is, tokens present in some data flow procedure carry besides the pointer to the heap structure mentioned earlier, a color tag which uniquely identifies the procedure activation the token is associated with.

The architecture of a parallel computing system capable of executing programs written in the data flow language is defined in [RUMB-75].

The data flow language defined in [DENN-75] can be viewed as an interpreted model of parallel computation. For a general study of flow of control in parallel programs, however, the data flow language model has to be used rather as an uninterpreted model. The specific interpretation of the actor nodes, the heap structure and consequently, the pointers to the heap structure carried by tokens will not be included in an uninterpreted data flow program scheme. It appears that the control coordination problems which may occur in such an uninterpreted data flow program scheme are fairly simple. This fact is also emphasized by a restriction imposed in the definition of the data flow language on the firing rules of the nodes, namely that any node can fire if and only if its output arcs do not contain tokens. In this way no more than one token can exist at any time on any arc of the model. Complications may arise, however, if procedure calls are modelled in an uninterpreted data flow program scheme because the procedures are stored as text-nodes in the heap structure.

Another potential problem with the data flow language regarded as a model of parallel computation is the fact that it can be only used to represent parallel programs in which the sequencing of the computational

steps is determined only by the data flow, that is by the availability of the input operands of the respective computations. If one is interested to model coordination problems of asynchronous concurrent processes in general, then the data flow language model becomes inappropriate. From this point of view it becomes apparent that the CL-Nets of [SONN-75] represent a model of systems exhibiting concurrency whose definition is very close to the original definition of the Petri Net Model. In this way, the CL-Net Model maintains the generality of the ordinary Petri Nets, namely the capability to represent asynchronous concurrent systems in general. On the other hand, the modifications introduced in the execution rules of the CL-Nets with respect to the original execution rules of the Petri Nets are minimal, and thus the CL-Net Model also has the advantage of a relatively simple representation of the modelled systems.

Section 1.2 OVERVIEW OF THE THESIS

In Section 1.1 we have explained how and why the idea of using colors to represent distinct control flow streams operating simultaneously in concurrent systems was developed. We have tried to emphasize the reasons why the use of colors would enable a simpler and more natural representation of reentrancy, recursivity and pipe-lining of control in asynchronous concurrent structures and which would also eliminate some of the drawbacks of the other solutions to these problems known so far. Consequently, we have defined a model of concurrent systems having the capability of handling "colored" representations of control flow. We could have proceeded in this respect in two different ways, that is we could have defined a completely new model or we could have tried to extend the definition of an already existing model of parallel systems in order to make it capable of handling "colored" representations of control flow streams. We have chosen the latter method in order to have a basis for comparison between our model and other established models of parallel systems. Thus, we have selected the Petri Net Model as the formal model underlying our extension. The reasons for this specific selection were:

i) Petri Nets had been defined initially with a broad scope in mind, namely to represent formally coordination schemes of asynchronous

concurrent processes rather than specifically parallel programs. Nonetheless, parallel programs themselves can be thought of as coordinated collections of asynchronous concurrent processes where the processes are the various computational steps of the program. In this way we hope to increase the generality of an already quite general formal model.

ii) More analytical work has been performed on the Petri Net Model than on other formal models of concurrent systems.

We have modified the structure and the execution rules of the Petri Nets in order to be able to handle colored tokens. From this point of view our model is a generalization of the Colored Petri Nets introduced in [SONN-75]. While extending the definition of the ordinary Petri Net Model we have tried not to complicate our model beyond a reasonable level. After all, any formal model of concurrent systems is useful only as long as it offers a simple, natural and easy-to-understand abstract representation of the real-life systems.

Further our objective was to analyze formally the modelling power of the model we have defined and to find a measure of its representation capabilities, as accurately as possible. In more specific terms we would like to get some insight into what is gained, as far as the representation capabilities of the model are concerned, by introducing colored tokens in the ordinary Petri Net Model.

We have mentioned earlier that our model is a generalization of the model defined by Sonnenburg in [SONN-75]. There, Petri Nets augmented with colored tokens (called CL-Nets) were used mainly as a more compact representation of parallel algorithms than that offered by ordinary Petri Nets. No further analytical analysis of the model was pursued. However, it was shown that the class of concurrent systems representable by the CL-Nets includes the class of concurrent systems representable by ordinary Petri Nets (with uncolored tokens) as a proper subset. This assertion was proven by showing that CL-Nets can correctly model a producer-consumer problem mentioned earlier in [KOSA-73]. Also in [KOSA-73] it was proven that ordinary Petri Nets cannot represent correctly the same problem. It is interesting to mention that the respective producer-consumer synchronization problem does

not involve in any way reentrancy or recursivity of control flow. This indicates that the use of colored tokens positively affects the representation capabilities of the Petri Nets with respect to the modelling of synchronization schemes in areas other than those envisaged in Section 1.1 (i.e., reentrancy, recursivity, pipe-lining, etc.), and thus encourages one to attempt a more in-depth analysis of the representation power of the model.

A general framework for the modelling of systems exhibiting concurrency, called the Colored Petri Model, is defined in Chapter II. The Colored Petri Model (CPM) has emerged from the association of the Generalized Petri Nets with sets of colors. Starting with real-life synchronization situations, two classes of the CPM having simple structures, namely the C-Colored Petri Model (C-CPM) and the EC-Colored Petri Model (EC-CPM), are singled-out. The C-CPM and the EC-CPM are defined such that they offer a compact and natural representation for various coordination structures of practical significance.

In Chapter III we analyze in detail the representation capabilities of the EC-CPM. For this purpose, the family of computation sequence sets and the family of terminal computation sequence sets of the EC-CPM are defined and their closure properties under various composition operations and mappings as well as their extents are studied.

Chapter IV presents a study of the representation power of the C-CPM. There, the C-CPM is compared with various other formal models of concurrent systems. Chapter IV is concluded with the comparison between the representation power of the EC-CPM and that of the C-CPM.

Several modelling examples were designed in order to support the necessity and appropriateness of the concepts of dynamic priority hierarchy, conflict and queueing of control flow streams incorporated in the C-CPM and EC-CPM, respectively. These modelling examples are exhibited in Chapter V.

Finally, Chapter VI presents a few conclusions which can be drawn from this research regarding "colored" representations of control flow in asynchronous concurrent structures.

CHAPTER II

THE COLORED PETRI MODEL

Section 2.1 INTRODUCTION AND PRELIMINARY DEFINITIONS

In this chapter we shall formally define the Colored Petri Model (CPM). We shall also define two classes of the CPM, named the C-Colored Petri Model (C-CPM) and the EC-Colored Petri Model (EC-CPM), respectively. These two classes of the CPM are obtained by imposing certain restrictions on the definitions pertinent to the CPM. Before defining the CPM, however, we shall first give a few basic definitions and notations.

Definition 2.1.1

A Generalized Petri Net (GPN) is a system $N = (T, P, I, O)$ where:

1. T is a finite set of transitions; $T = \{t_1, \ldots, t_n\}$.
2. P is a finite set of places; $P = \{p_{n+1}, \ldots, p_{n+m}\}$

where $n = |T|$, $m = |P|$ and $T \cap P = \phi$.

3. I is an input incidence function; $I: P \times T \rightarrow Z^o$

where $Z^o = \{0, 1, 2, \ldots\}$ is the set of nonnegative integers.

4. O is an output incidence function; $O: T \times P \rightarrow Z^o$.

A Generalized Petri Net is represented graphically by a bipartite directed graph where:

1. Transitions are represented by bars.
2. Places are represented by circles.
3. Circles and bars are connected by directed arcs. The set of arcs associated with the sets T and P is denoted by A. Let $t_k \in T$ be a transition and let $p_j \in P$ be a place of the GPN $N = (T, P, I, O)$. For every such pair $(p_j, t_k) \in P \times T$, there are exactly $I(p_j, t_k)$ arcs directed from p_j to t_k. If $I(p_j, t_k) > 0$, p_j is called an input place of transition t_k. The arcs connecting p_j to t_k are denoted by a^i_{jk}, $i = 1, \ldots, I(p_j, t_k)$ and each arc a^i_{jk} is called an input arc of transition t_k. Similarly, for every pair $(t_k, p_j) \in T \times P$, there are exactly $O(t_k, p_j)$ arcs directed from

t_k to p_j. If $O(t_k, p_j) > 0$, p_j is called an output place of transition t_k. The arcs corresponding to such a pair (t_k, p_j) are denoted by a^i_{kj}, $i = 1, \ldots, O(t_k, p_j)$ and each arc a^i_{kj} is called an output arc of transition t_k.

Definition 2.1.2

If $t_k \in T$, the bag of input places of transition t_k is defined to be:

$$I_k = < p_j^s \mid p_j \in P \text{ and } I(p_j, t_k) = s > 0 >$$

(for bag notations and definitions, see Appendix A)

Definition 2.1.3

If $t_k \in T$, the bag of output places of transition t_k is defined to be:

$$O_k = < p_j^s \mid p_j \in P \text{ and } O(t_k, p_j) = s > 0 >$$

Note: The definition of a GPN $N = (T, P, I, O)$ does not impose $I_k \cap O_k = \phi$, for any $t_k \in T$. Thus, a place $p_j \in P$ may be both an input as well as an output place for some transition $t_k \in T$. Also, $I(p_j, t_k)$ may be different from $O(t_k, p_j)$, i.e. the number of input arcs from p_j to t_k is not necessarily equal to the number of output arcs from t_k to p_j.

Definition 2.1.4

A marking of a GPN $N = (T, P, I, O)$ is a total, single-valued mapping from P into Z^o. An arbitrary marking of N is denoted M^i, where the superscript i distinguishes it from other possible, distinct markings of N. For each $p_j \in P$, $M^i(p_j)$ indicates the number of tokens present in place p_j in the marking M^i. Graphically, a token is represented by drawing a dot in the corresponding place.

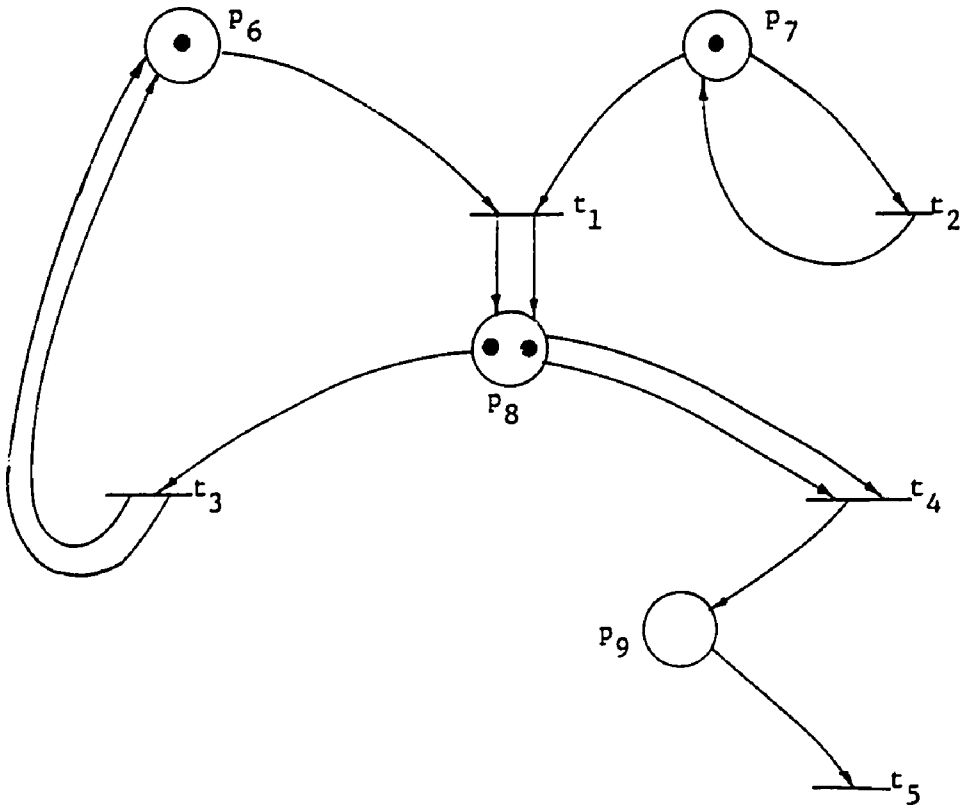
Figure 2.1.1 gives an example of a GPN.

Definition 2.1.5

A set of colors is defined to be a lattice $C = (X, \leq)$ where:

1. X is a set.
2. $\leq$ is a partial ordering relation defined on the set X.

An element of the set X is called a color.



$N = (T, P, I, O)$

$T = \{t_1, t_2, t_3, t_4, t_5\}$; $P = \{p_6, p_7, p_8, p_9\}$

I	t_1	t_2	t_3	t_4	t_5
p_6	1	0	0	0	0
p_7	1	1	0	0	0
p_8	0	0	1	2	0
p_9	0	0	0	0	1

O	p_6	p_7	p_8	p_9
t_1	0	0	2	0
t_2	0	1	0	0
t_3	2	0	0	0
t_4	0	0	0	1
t_5	0	0	0	0

P	M^i
p_6	1
p_7	1
p_8	2
p_9	0

$I_1 = < p_6 p_7 >$ $I_2 = < p_7 >$ $I_3 = < p_8 >$ $I_4 = < p_8^2 >$ $I_5 = < p_9 >$

$O_1 = < p_8^2 >$ $O_2 = < p_7 >$ $O_3 = < p_6^2 >$ $O_4 = < p_9 >$ $O_5 = \emptyset$

Figure 2.1.1
Example of a Generalized Petri Net

In the following definitions we shall assume that each token present in a place of a GPN N carries a color from a unique color set associated with N. The implications of this fact and the way in which colors from the color set are assigned to tokens will become apparent later on.

Definition 2.1.6

Let N = (T, P, I, O) be a GPN and let C = (X,≤) be the set of colors associated with N. Given some marking M^i of N, the color bag of a place $p_j \in P$ is defined to be:

$$C_j^i = < c_r^{n_r} \mid c_r \in X \text{ and } c_r \text{ is the color of } n_r \text{ tokens present in } p_j \text{ in the marking } M^i >.$$

By convention, if $M^i(p_j) = 0$, then $C_j^i = \phi$, the empty bag. Note also that following relation must hold:

$$\sum_{c_r \in \mathcal{D}(C_j^i)} \#(c_r, C_j^i) = \sum_{c_r \in \mathcal{D}(C_j^i)} n_r = M^i(p_j)$$

where $\mathcal{D}(C_j^i)$ denotes as usual, the domain of the bag C_j^i.

Definition 2.1.7

Let N = (T, P, I, O) be a GPN and let C = (X,≤) be the set of colors associated with N. Given some marking M^i of N, the corresponding color marking of N is a total, single-valued mapping CM^i: $P \rightarrow < X >^*$ where $< X >^*$ denotes the bag closure of the set X. For any place $p_j \in P$, $CM^i(p_j) = C_j^i$.

Definition 2.1.8

Let $N_1 = (T_1, P_1, I_1, O_1)$ and $N_2 = (T_2, P_2, I_2, O_2)$ be two GPN's, and let $C_1 = (X_1,\leq_1)$ and $C_2 = (X_2,\leq_2)$ be the sets of colors associated with N_1 and N_2, respectively. Let us also assume that $P_1 \cap P_2 = \phi$ (this condition can always be achieved through appropriate renaming). Given some color markings CM_1^i and CM_2^r of N_1 and N_2, respectively, we shall define the direct sum color marking of CM_1^i and CM_2^r to be

the total, single-valued mapping $CM_1^i \nabla CM_2^r: P_1 \cup P_2 \rightarrow < X_1 \cup X_2 >^*$ such that for any place p_j

$$CM_1^i \nabla CM_2^r (p_j) = \begin{cases} CM_1^i(p_j) & \text{if } p_j \in P_1 \\ CM_2^r(p_j) & \text{if } p_j \in P_2 \end{cases}$$

The notion of direct sum color marking will be used in the study of the composition properties of the EC-CPM, to be presented later on. It can easily be extended to more than two GPN's.

Section 2.2 THE COLORED PETRI MODEL

Based on the definitions given in the previous section, we can proceed now to define the Colored Petri Model.

Definition 2.2.1

A Colored Petri Net (CPN) is a system CN = (N, C, F, R, Q) where:

1. N = (T, P, I, O) is a Generalized Petri Net. Let A be the set of arcs associated with the sets T and P.

2. C = (X,≤) is the set of colors associated with N.

3. F: A → f is a total, single-valued mapping. The range f of the mapping F is a set of functions:

$$f = \{f_{jk}^i \mid f_{jk}^i = F(a_{jk}^i) \text{ for some arc } a_{jk}^i \in A \text{ and}$$

$$f_{jk}^i: \mathscr{P}(X) \rightarrow X \text{ if } a_{jk}^i \text{ is an input arc or}$$

$$f_{jk}^i: \mathscr{P}(X) \times \mathscr{P}(X) \rightarrow X \text{ if } a_{jk}^i \text{ is an output arc}\}$$

Here $\mathscr{P}(X)$ denotes the power set of the set X.

4. $R \in \{R_L, R_{LE}, R_E, R_{GE}, R_G\}$ where

$R_L = \{(c,c') \mid c \in X, c' \in X \text{ and } c < c'\}$

$R_{LE} = \{(c,c') \mid c \in X, c' \in X \text{ and } c \leq c'\}$

$R_E = R_{LE} - R_L = \{(c,c) \mid c \in X\}$ (the identity relation on the set X)

$R_{GE} = R_{LE}^{-1} = \{(c,c') \mid c \in X, c' \in X \text{ and } c \geq c'\}$

$R_G = R_L^{-1} = \{(c,c') \mid c \in X, c' \in X \text{ and } c > c'\}$.

5. Q: T → q is a total, single-valued mapping. The range q

of the mapping Q is a set of functions:

$$q = \{q_k \mid q_k = Q(t_k) \text{ for some } t_k \in T \text{ and } q_k: \mathcal{P}(X) \rightarrow X\} .$$

In other words, the mapping F associates with each arc $a^i_{jk} \in A$ a function f^i_{jk}. If a^i_{jk} is an input arc of some transition $t_k \in T$, the corresponding function f^i_{jk} is called the color threshold function of the input arc a^i_{jk}. If a^i_{jk} is, on the contrary, an output arc of some transition $t_j \in T$, the function f^i_{jk} is called the output color function of the output arc a^i_{jk}. R is a relation defined on the set X, which is derived from the partial ordering $\leq$ which determines the lattice $C = (X, \leq)$. The definitions which follow will explain the role of the functions f^i_{jk} and of the relation R. The mapping Q associates with each transition $t_k \in T$ a function q_k called the priority function of the transition t_k. Further explanations regarding the functions q_k will be given in Section 2.3.

Let CN = (N, C, F, R, Q) be a CPN and let us assume that the GPN N = (T, P, I, O) is in some color marking CM^i. Consider some transition $t_k \in T$ and suppose $p_j \in \mathcal{D}(I_k)$. Consequently, there are m input arcs $a^1_{jk}, \ldots, a^m_{jk}$ from p_j to t_k, where $m = I(p_j, t_k)$. Then, for each input arc a^r_{jk}, $1 \leq r \leq m$, $f^r_{jk}(\mathcal{D}(C^i_j))$ is a color $c_r \in X$.

Definition 2.2.2

A color c of a token present in place p_j in the color marking CM^i is said to be the enabling color of transition t_k on input arc a^r_{jk} if $(c, c_r) \in R$, and there exists no other token of color c', $c \neq c'$, present in p_j in the color marking CM^i, such that $(c, c') \in R$ and $(c', c_r) \in R$.

Note: The ordering relation $\leq$ defined on the colors of the set X is only a partial ordering. Thus, depending on the relation R and on the particular color marking CM^i, there may exist tokens of colors c and c', in the color bag C^i_j such that $(c, c') \notin R$, $(c', c) \notin R$ and $(c, c_r) \in R$, $(c', c_r) \in R$. In other words, c and c' are incomparable under the relation R and both colors are eligible to become the enabling color of transition t_k on the input arc a^r_{jk}, according to Definition 2.2.2. For a situation like this where

there exists a set of several tokens of distinct colors, all eligible to become the enabling color on some input arc a^r_{jk}, the general rule will be that the enabling color is selected arbitrarily from the set of candidate colors. We have to emphasize that after the selection of the enabling color for some input arc a^r_{jk} is made, the selected color is the unique enabling color for the respective arc, regardless of how many colors were initially eligible to become its enabling color. We should also mention that once the enabling color for some arc a^r_{jk} is selected, we shall not differentiate among the tokens of that color present in p_j at that time. Thus, we select an enabling color for the arc a^r_{jk} and not a particular token of that color. It should be noticed that if for some color bag C^i_j, $f^r_{jk}(\mathscr{D}(C^i_j)) = c_r$, then $c_r \in X$, but it is not necessary that $c_r \in \mathscr{D}(C^i_j)$. Also, if there exists no color $c \in \mathscr{D}(C^i_j)$ such that $(c,c_r) \in R$, then simply there exists no enabling color for a^r_{jk} in the color bag C^i_j. Thus, the function f^r_{jk} actually sets a color threshold for the selection of the enabling color of a^r_{jk} in any color marking CM^i of the CPN CN. The arc a^r_{jk} is said to claim a token of color c if c is the enabling color of a^r_{jk}.

Definition 2.2.3

The bag θ^i_{jk} of enabling colors of t_k from the input place p_j is defined for some color bag C^i_j as follows:

$$\theta^i_{jk} = \langle c \mid c \text{ is the enabling color of } a^r_{jk},\ 1 \le r \le I(p_j, t_k) \rangle$$

For each color $c_s \in X$, $n_s = \#(c_s, \theta^i_{jk})$ indicates the number of input arcs from p_j to t_k which claim tokens of the same color, c_s, from C^i_j.

Definition 2.2.4

The bag of enabling colors of transition t_k in the color marking CM^i is:

$$\theta^i_k = \bigcup_{p_j \in \mathscr{D}(I_k)} \theta^i_{jk}$$

We can now explain the role of the output color functions. Let

us assume that $p_s \in \mathcal{D}(O_k)$ where t_k is an arbitrary transition of T. Suppose also that $O(t_k, p_s) = n$. Consequently, there are n output arcs $a_{ks}^1, \ldots, a_{ks}^n$ directed from t_k to p_s. For each such output arc a_{ks}^r, the corresponding output color function f_{ks}^r determines the color of the token placed by the arc a_{ks}^r in the place p_s when t_k fires (an exact definition of the "firing of a transition" will be given in the following section). Thus, the color of the token produced by t_k on the output arc a_{ks}^r is $f_{ks}^r(\mathcal{D}(\theta_k^i), \mathcal{D}(C_s^{i-}))$. The color of the output token is determined by the domain $\mathcal{D}(\theta_k^i)$ of the bag of enabling colors of transition t_k in the color marking CM^i and by the domain of the color bag C_s^{i-} of the output place p_s, where $C_s^{i-} = C_s^i$ if $p_s \notin \mathcal{D}(I_k)$ and $C_s^{i-} = C_s^i - \theta_{sk}^i$ if $p_s \in \mathcal{D}(I_k)$. The occurrence of the color bag C_s^{i-} will be further explained in the next section.

Definition 2.2.5

A Colored Petri Model (CPM) is a system $\mathcal{P} = (CN, CM^0)$ where:

1. $CN = (N, C, F, R, Q)$ is a Colored Petri Net.
2. CM^0 is the initial color marking of CN.

Section 2.3 EXECUTION RULES FOR THE COLORED PETRI MODEL

So far we have defined the structure of the CPM. We can present now the dynamic behavior of the CPM which we shall call the "execution of the CPM."

Definition 2.3.1

Let $CN = (N, C, F, R, Q)$ be a CPN in some color marking CM^i. A transition t_k of N is said to be enabled in the color marking CM^i if the following condition holds:

$$\theta_{jk}^i \subseteq C_j^i \text{ for every } p_j \in \mathcal{D}(I_k)$$

where θ_{jk}^i denotes the bag of enabling colors of transition t_k from input place p_j, as defined earlier.

In other words, a transition t_k is enabled in some color marking CM^i if each input arc a_{jk}^r, for all places $p_j \in \mathcal{D}(I_k)$, can claim

a distinct enabling token from the respective color bag C_j^i. Two tokens of the same color, present in some color bag C_j^i, are considered to be distinct from this point of view.

A transition t_k enabled in some color marking CM^i may be selected to fire. The rule under which this selection is made will be given later in this section. We shall first describe the effect produced by the firing of a transition.

Definition 2.3.2

A transition t_k enabled in some color marking CM^i fires by performing the following operations:

1. For each color $c_r \in \mathcal{D}(\theta_{jk}^i)$, n_r tokens of color c_r are removed from the corresponding input place p_j, where $n_r = \#(c_r, \theta_{jk}^i)$. This operation is performed for all $p_j \in \mathcal{D}(I_k)$ and results in a new, "intermediate" color marking CM^{i-} in which each place $p_j \in P$ is associated with some color bag C_j^{i-}, where:

$$C_j^{i-} = \begin{cases} C_j^i & \text{if } p_j \notin \mathcal{D}(I_k) \\ C_j^i - \theta_{jk}^i & \text{if } p_j \in \mathcal{D}(I_k) \end{cases}$$

2. For each output arc $a_{ks}^r \in A$, for all output places $p_s \in \mathcal{D}(O_k)$, a token of color $f_{ks}^r(\mathcal{D}(\theta_k^i), \mathcal{D}(C_s^{i-}))$ is placed in the corresponding output place p_s.

Before we proceed with the definition of the execution rule of the CPM, we shall first make a few observations on the "conflicts" which may occur in the process of selecting the transition to be fired.

Let CN = (N, C, F, R, Q) be a CPN in some color marking CM^i. We shall denote by E^i the set of transitions enabled in the color marking CM^i. Let us also denote for each place $p_j \in P$ the set of transitions for which p_j is an input place by $T(p_j)$:

$$T(p_j) = \{t_k \mid t_k \in T \text{ and } I(p_j, t_k) > 0\}.$$

Definition 2.3.3

There exists a conflict at the place p_j for the color c_r in the color marking CM^i if and only if:

$$\sum_{t_k \in E^i \cap T(p_j)} \#(c_r, \theta^i_{jk}) > \#(c_r, C^i_j)$$

In other words, we shall say that a conflict has occurred at the place p_j for the color c_r if the number of tokens of color c_r claimed from p_j by the transitions enabled in the color marking CM^i is larger than the number of tokens of this color contained in the color bag C^i_j. We note that in some color marking CM^i, there may exist conflicts at the same place p_j for several distinct colors.

Two transitions $t_k \in T$ and $t_s \in T$ are said to <u>conflict</u> in the color marking CM^i if the following conditions hold:

1. $t_k \in E^i$ and $t_s \in E^i$
2. There exists at least one place $p_j \in \mathcal{D}(I_k) \cap \mathcal{D}(I_s)$ such that there is a conflict at p_j for some color c_r, where $c_r \in \mathcal{D}(\theta^i_{jk}) \cap \mathcal{D}(\theta^i_{js})$.

If two transitions t_k and t_s conflict, this relation between t_k and t_s is denoted by $t_k \; o \; t_s$. For convenience, we shall assume that $t_k \; o \; t_k$ for every $t_k \in E^i$.

We can extend the conflict relation defined above as follows. Let CONF be the relation on E^i such tat for any $t_k \in E^i$ and $t_m \in E^i$, $(t_k, t_m) \in$ CONF if there exists a sequence of transitions $t_k = t_{r1}, t_{r2}, \ldots, t_{rn} = t_m$ and $t_{rj} \; o \; t_{rj+1}$ for $1 \leq j \leq n-1$.

<u>Lemma 2.3.1</u>

CONF is an equivalence relation on E^i

<u>Proof:</u> We have to show that CONF is a reflexive, symmetric and transitive relation on E^i.

i. By the definition of CONF, for every $t_k \in E^i$, $(t_k, t_k) \in$ CONF. Thus, CONF is reflexive.

ii. We shall first remark that by Definition 2.3.3, if $t_k \; o \; t_m$ in CM^i, then $t_m \; o \; t_k$ in CM^i as well. Let us assume now that $(t_k, t_m) \in$ CONF. Consequently, there exists a sequence of transitions $t_k = t_{r1}, t_{r2}, \ldots, t_{rn} = t_m$ such that $t_{rj} \; o \; t_{rj+1}$ for $1 \leq j \leq n-1$. Since $t_{rj+1} \; o \; t_{rj}$ also, for $1 \leq j \leq n-1$, it follows that $(t_m, t_k) \in$ CONF. Therefore, CONF is symmetric.

iii. Let us assume that $(t_k, t_m) \in CONF$ and $(t_m, t_s) \in CONF$. Then, there exist the transition sequences

$t_k = t_{r1}, t_{r2}, \ldots, t_{rn} = t_m$ and $t_m = t_{rn}, t_{rn+1}, \ldots, t_{rq} = t_s$ such that $t_{rj} \circ t_{rj+1}$, for $1 \leq j \leq q-1$. Consequently, $(t_k, t_s) \in CONF$. Thus, CONF is transitive.

□

An equivalence class of the relation CONF is called a conflict cluster. The quotient set of E^i modulo CONF, i.e. $E^i/CONF$, is called the set of conflict clusters of the color marking CM^i and is denoted by CF^i.

Lemma 2.3.1 immediately leads to the following fact:

Lemma 2.3.2

The set CF^i of conflict clusters of some color marking CM^i is a partition of the set E^i of transitions enabled in CM^i.

Let $CF^i = \{CT_1^i, \ldots, CT_m^i\}$ where CT_r^i, $1 \leq r \leq m$, denotes the conflict clusters present in CM^i.

Definition 2.3.4

Given a conflict cluster CT_r^i, a conflict subcluster is a maximal subset SCT_{rk}^i of CT_r^i such that for any two transitions $t_m \in SCT_{rk}^i$ and $t_s \in SCT_{rk}^i$, $t_m \circ t_s$.

In other words, the transitions which form a conflict subcluster are all pairwise in conflict in the color marking CM^i.

Definition 2.3.5

A transition t_k is said to cover another transition t_m in the color marking CM^i if the following conditions hold:

1. $t_k \in E^i$ and $t_m \in E^i$.
2. $q_k(\mathcal{D}(\theta_k^i)) > q_m(\mathcal{D}(\theta_m^i))$

If t_k covers t_m in the color marking CM^i, we denote this relation between t_k and t_m by t_k/t_m. Equivalently, we shall say that t_k has higher priority than t_m in the color marking CM^i if t_k/t_m.

Definition 2.3.6

The set of majorants of a conflict subcluster SCT_{rk}^i is a maximal

subset maj$\{SCT^i_{rk}\}$ of SCT^i_{rk} such that if $t_m \in$ maj$\{SCT^i_{rk}\}$, then there exists no other transition $t_s \in SCT^i_{rk}$ and t_s / t_{m_i}

A transition belonging to the set maj$\{SCT^i_{rk}\}$ is called a majorant of the conflict subcluster SCT^i_{rk}.

Note: Since the colors of the color set C form only a partial ordering rather than a total ordering, the majorant of a conflict subcluster does not necessarily have to be unique. This is why maj$\{SCT^i_{rk}\}$ can be a set rather than a single transition.

In order to simplify notations in the following definitions, we shall denote by SCF^i_r the set of all conflict subclusters of a conflict cluster CT^i_r. Let $SCF^i_r = \{SCT^i_{r1}, \ldots, SCT^i_{rn}\}$. From Definition 2.3.4 it follows that:

$$\bigcup_{k=1}^{|SCF^i_r|} SCT^i_{rk} = CT^i_r$$

However, SCF^i_r is not necessarily a partition of the parent conflict cluster CT^i_r since a transition $t_s \in CT^i_r$ may participate in several distinct conflict subclusters.

With the definitions given above, we can proceed to define the conditions under which a transition of a CPN can fire:

Definition 2.3.7

A transition t_k is firable in some color marking CM^i if and only if t_k is a majorant of all the conflict subclusters in which it is contained. Any firable transition may be selected to fire in CM^i.

We shall sometimes refer to Definition 2.3.7 as to the "execution rule of the CPM." The question can be raised whether the selection rule of Definition 2.3.7 can always enable one to find a firable transition from a non-empty set of enabled transitions. The answer to this question is positive, as proved in the following lemma.

Lemma 2.3.3

There exists a firable transition in any conflict cluster.

Proof: Let CT^i_r be a conflict cluster.

Let:

$$COL_r^i = \{q_s(\mathcal{D}(\theta_s^i)) \mid t_s \in CT_r^i\}$$

Evidently, COL_r^i is a finite subset of the set of colors X. Consequently, COL_r^i must have at least one maximal element. Let us assume that c is a maximal element of COL_r^i and suppose t_k is a transition of CT_r^i such that $q_k(\mathcal{D}(\theta_k^i)) = c$. Hence, t_k is a majorant of all the conflict subclusters in which it is contained and, therefore, firable.

□

Note: $q_k(\mathcal{D}(\theta_k^i))$ is a maximal element of the set COL_r^i is a sufficient condition for t_k to be firable in the color marking CM^i, but not a necessary condition.

We shall now give a few definitions which we shall make use of in later sections. Suppose t is a transition firable in some color marking CM^i. Let us assume that CM^{i+1} is the color marking obtained after the firing of t in CM^i. We shall denote this fact by $CM^i\ [t>\ CM^{i+1}$. We can extend this notion in the following definition:

Definition 2.3.8

Given an arbitrary color marking CM^i, a firing sequence from CM^i is a string $\gamma_j = t_{j1} \ldots t_{jk}$ of transition names ($\gamma_j \in T^*$) such that there exist the color markings CM^{i+r}, $1 \leq r \leq k$, and $CM^{i+r-1}\ [t_{jr}>\ CM^{i+r}$.

If CM^{i+k} is the color marking obtained after the execution of the firing sequence γ_j from CM^i, we shall say that γ_j leads from CM^i to the color marking CM^{i+k} and shall denote this fact by $CM^i\ [\gamma_j>\ CM^{i+k}$.

Definition 2.3.9

Given a CPM $\mathcal{P} = (CN, CM^o)$, the set of firing sequences of $\mathcal{P}$ is defined to be:

$$S(CM^o) = \{\gamma \mid \gamma \in T^* \text{ and } CM^o\ [\gamma>\ CM^r, \text{ for some arbitrary color marking } CM^r\}.$$

If we are given a certain final color marking CM^f at which the execution of $\mathcal{P}$ has to stop, we can define the set of terminal firing sequences:

Definition 2.3.10

Given a CPM $\mathscr{P} = (CN, CM^o)$ and a final color marking CM^f, the set of terminal firing sequences of $\mathscr{P}$ is defined to be:

$$T(CM^o, CM^f) = \{\gamma \mid \gamma \in T^* \text{ and } CM^o\ [\gamma\rangle\ CM^f\}$$

Note: The empty string λ may or may not belong to $T(CM^o, CM^f)$ depending on whether $CM^o = CM^f$ or not. On the other hand, λ is always an element of $S(CM^o)$.

Definition 2.3.11

Given a CPM $\mathscr{P} = (CN, CM^o)$, a color marking CM is said to be reachable from CM^o if there exists at least one firing sequence $\gamma \in T^*$ such that $CM^o\ [\gamma\rangle\ CM$.

The set of all color markings reachable from CM^o is denoted by $R(CM^o)$.

Definition 2.3.12

Given a CPM $\mathscr{P} = (CN, CM^o)$, the set of active color markings reachable from CM^o is the subset $E(CM^o)$ of $R(CM^o)$ such that:

$$E(CM^o) = \{CM^i \mid CM^i \in R(CM^o) \text{ and } E^i \neq \phi\}$$

Definition 2.3.13

Let $\mathscr{P}_1 = (CN_1, CM_1^o)$ and $\mathscr{P}_2 = (CN_2, CM_2^o)$ be two arbitrary CPM's and let $R_1(CM_1^o)$ and $R_2(CM_2^o)$ be the sets of all reachable color markings of $\mathscr{P}_1$ and $\mathscr{P}_2$, respectively. Then, the set of all reachable direct sum color markings of $\mathscr{P}_1$ and $\mathscr{P}_2$ is defined to be:

$$R_1(CM_1^o) \nabla R_2(CM_2^o) = \{CM_1^i \nabla CM_2^j \mid CM_1^i \in R_1(CM_1^o) \text{ and } CM_2^j \in R_2(CM_2^o)\}$$

At this point, an explanation is required. In the definitions pertinent to the CPM which we have presented so far, no assumptions were made about the lattice of colors $C = (X, \leq)$, and in particular about the cardinality of the set X. Also, no restrictions were imposed on the type of the color threshold, color output or of the priority functions. In fact, any total, single-valued mappings can be used. Such a wide selection range could eventually decrease the possibility of developing effective analysis tools for the CPM. Our intention, however, was to define a new, general framework for the modelling of

systems exhibiting concurrency which then can be further restricted according to the needs for modelling. We have defined two special classes of the CPM, namely the C-CPM and the EC-CPM. The formal definitions of these two classes of the CPM will be given in the sections of this chapter immediately following. The properties of the C-CPM and of the EC-CPM will be studied in detail in the sequel. Their usefulness is implicity justified by their simplicity and will be illustrated by means of numerous modelling examples. In spite of their simplicity, the C-CPM and the EC-CPM are shown to be extremely powerful models of concurrent systems.

Section 2.4 THE C-COLORED PETRI MODEL

The development of this particular class of the CPM was inspired by the following producer-consumer synchronization problem reported in [KOSA-73]. The coordination system contains two producer processes P_1 and P_2, two buffers B_1 and B_2, and two consumer processes C_1 and C_2. The two buffers are not shared, that is producer process P_i ($i = 1, 2$) deposits items only in buffer B_i while consumer process C_i attempts to consume items only from the associated buffer B_i. Either producer process can be activated at any time and it produces and deposits an item in the corresponding buffer. On the other hand, a consumer process can be activated only if the associated buffer is not empty. An additional constraint is imposed, namely if both C_1 and C_2 are activated, then C_1 has priority to consume over C_2. Otherwise viewed, C_1 and C_2 consume items from their associated buffers under the control of a decider module which enforces the priority rule mentioned above. This producer-consumer synchronization problem is represented schematically in Figure 2.4.1.

The basic coordination aspect of this synchronization system is the fact that a priority hierarchy is imposed among the set of underlying processes. From this point of view, the decider module can be regarded as a critical region with a priority rule over the competing processes.

It has been proved ([KOSA-73)] that the producer-consumer

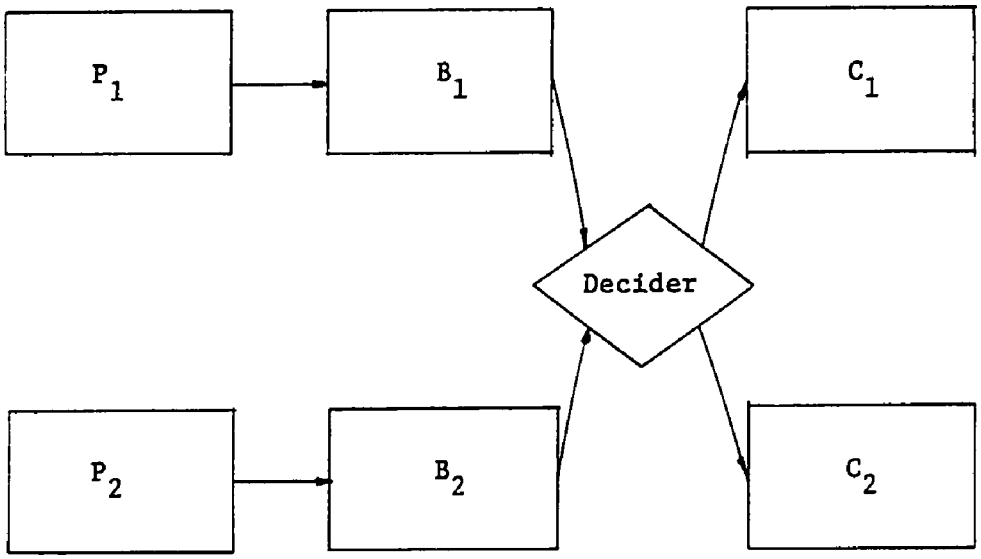


Figure 2.4.1

Schematical Representation of a Producer-Consumer Synchronization Problem

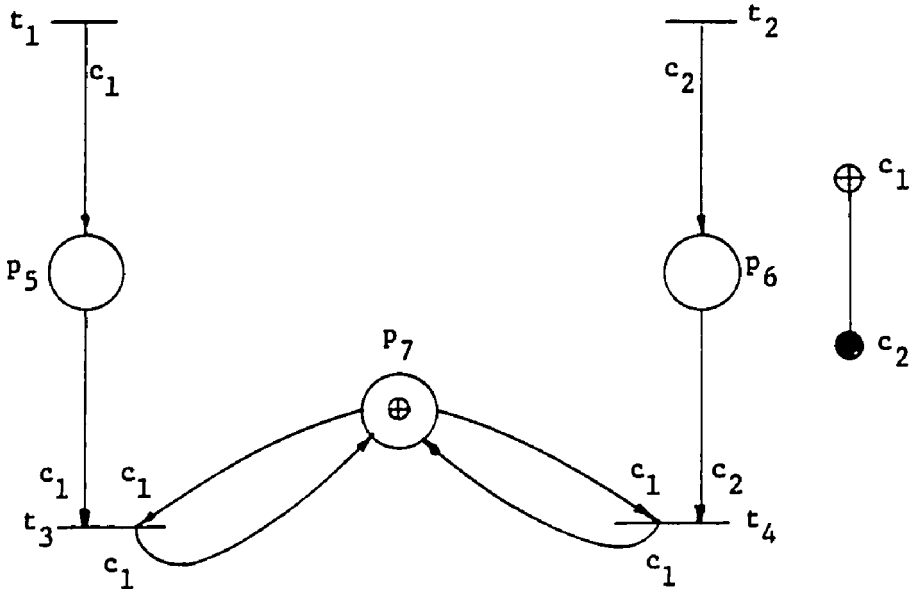


Figure 2.4.2

C-CPM of the Producer-Consumer Synchronization Problem of Figure 2.4.1

synchronization problem described above cannot be modelled correctly by any Petri Net. On the other hand, it was shown ([PETE-73], [AGAR-75]) that the representation capabilities of the Petri Net Model are equivalent to the representation capabilities of several other formal models of concurrent systems, such as the Complex Graph Model of Computation, the P-Nets, the Vector Addition and Vector Replacement Systems, etc. Consequently, these formal models cannot represent correctly this type of synchronization systems, either.

In what follows we shall define a very simple class of the CPM, called the C-Colored Petri Model, which can represent in a natural way synchronization systems which incorporate priority hierarchies, such as that described at the beginning of this section.

Definition 2.4.1

A C-Colored Petri Net (C-CPN) is a Colored Petri Net $CN = (N, C, F, R, Q)$ where:

1. $N = (T, P, I, O)$ is a GPN. Let A be the set of arcs associated with the sets T and P.

2. $C = (X, \leq)$ is the set of colors associated with N, where X is a finite set.

3. $F: A \rightarrow f$ is a total mapping. The range f of the mapping F is a set of functions:

$$f = \{f^i_{jk} \mid f^i_{jk} = F(a^i_{jk}),\ a^i_{jk} \in A \text{ and } f^i_{jk}: \mathcal{P}(X) \rightarrow c^i_{jk} \text{ if } a^i_{jk} \text{ is an input arc, } c^i_{jk} \in X, \text{ or } f^i_{jk}: \mathcal{P}(X) \times \mathcal{P}(X) \rightarrow c^i_{jk} \text{ if } a^i_{jk} \text{ is an output arc}\}.$$

4. $R = R_{GE} = \{(c,c') \mid c \in X,\ c' \in X \text{ and } c \geq c'\}$. Since all models in the C-CPM class will employ the relation R_{GE} for the selection of enabling colors, we shall omit it when defining a C-CPN in the remainder.

5. $Q: T \rightarrow q$ is a total, single-valued mapping. The range q of the mapping Q is a singleton set containing the function g.l.b.: $\mathcal{P}(X) \rightarrow X$, where for any set $D \in \mathcal{P}(X)$, $D \neq \phi$, g.l.b.(D) denotes the greatest lower bound of the set D and refers to the ordering relation $\leq$ defined on the set X. By convention, $g.l.b.(\phi) = x_o$ where $x_o \in X$ is arbitrarily selected. In the sequel, we shall omit the

mapping Q from the definition of any C-CPN under the assumption that the mapping Q for the respective C-CPN obeys the definition specified above.

The definitions pertinent to the CPM given in the previous sections can immediately be translated to the case of the C-CPN (where the color set is a finite lattice).

Definition 2.4.2

A C-Colored Petri Model (C-CPM) is a CPM $CP = (CN, CM^o)$ such that:

1. $CN = (N, C, F)$ is a C-Colored Petri Net.
2. CM^o is the initial color marking of CN.

For convenience, we have not changed in the case of the C-CPM the notations used with the CPM.

Regarding the dynamic behavior of the C-CPM, we note that given a C-CPN CN in some color marking CM^i and an arbitrary transition t_k of CN, enabled in CM^i, the priority of the transition t_k is $g.\ell.b.(\mathcal{D}(\theta_k^i))$, where θ_k^i represents the bag of enabling colors of transition t_k in the color marking CM^i. Thus, the priority of t_k depends both on the color marking CM^i and on the particular selection of enabling colors in CM^i. Consequently, the priority of a transition may vary from one color marking to another.

Figure 2.4.2 displays the C-CPM representation of the producer-consumer synchronization problem described at the beginning of this section. In the figure, the color threshold and the color output functions are marked adjacent to the corresponding arcs. The set of colors is given in its Hasse diagram representation.

Transitions t_1 and t_2 represent the producer processes P_1 and P_2, respectively. Similarly, transitions t_3 and t_4 model the consumer processes C_1 and C_2, respectively. Places p_5 and p_6 represent the two buffers B_1 and B_2, respectively while place p_7 permits the implementation of the priority rule enforced, in the modelled synchronization system, by the decider module. If both places p_5 and p_6 are not empty, then the conflict at place p_7 for the token of color c_1 is always solved in favor of transition t_3. It is easy to see that

transition t_4 can fire if and only if transition t_3 is disabled or, equivalently, if and only if place p_5 is empty. Thus, the conditions under which transitions t_3 and t_4 can fire model exactly the conditions under which the respective consumer processes C_1 and C_2, could have consumed an item from the associated buffers.

In order for the reader to gain familiarity with the C-Colored Petri Model, we present another modelling example using the C-CPM. We have selected a well-known coordination problem of practical significance presented in [COUR-71], namely the readers-writers synchronization problem. The coordination system involves two classes of processes which access a shared resource. The first class of processes is formed by the "writers." Each process in this class must have exclusive access to the resource. The processes of the second class are called the "readers" and an unlimited number of processes from this class may simultaneously access the resource. We have chosen the case where the readers have priority over the writers, i.e. no reader is kept waiting unless a writer has already gained access to the resource. Figure 2.4.3 displays the original solution using semaphores, as given in [COUR-71].

As stated in [PETE-73] and [CERF-72], the P-Net and the Complex Graph Model of Computation, respectively, cannot provide a correct representation of the interaction between readers and writers. Since the Complex Graph Model of Computation and the Petri Net Model are equivalent, as far as their capabilities of representation are concerned ([PETE-73], [AGAR-75]), the difficulty of modelling the readers-writers synchronization problem extends to the Petri Net Model as well. The basic reason for the inadequacy of these models stems from the fact that they cannot implement decisions of the form "if both readers and writers are waiting to access the shared resource, then reader processes have priority over writer processes." We note that in the original solution of [COUR-71], the priority of the readers over the writers could be established only through the use of conditional P and V primitives.

The solution to the readers-writers synchronization problem

```
READER

integer readcount;  (initial value = 0)
semaphore mutex, w; (initial value for both = 1)

        P(mutex);
        readcount: = readcount + 1;
        if readcount = 1, then P(w);
        V(mutex);
              .
              .
              .
           reading is performed
              .
              .
              .
        P(mutex);
        readcount: = readcount - 1;
        if readcount = 0, then V(w);
        V(mutex);


        WRITER

        P(w);
          .
          .
          .
          writing is performed
          .
          .
          .
        V(w);
```

Figure 2.4.3

Solution to the Readers-Writers Synchronization Problem Using Semaphores.

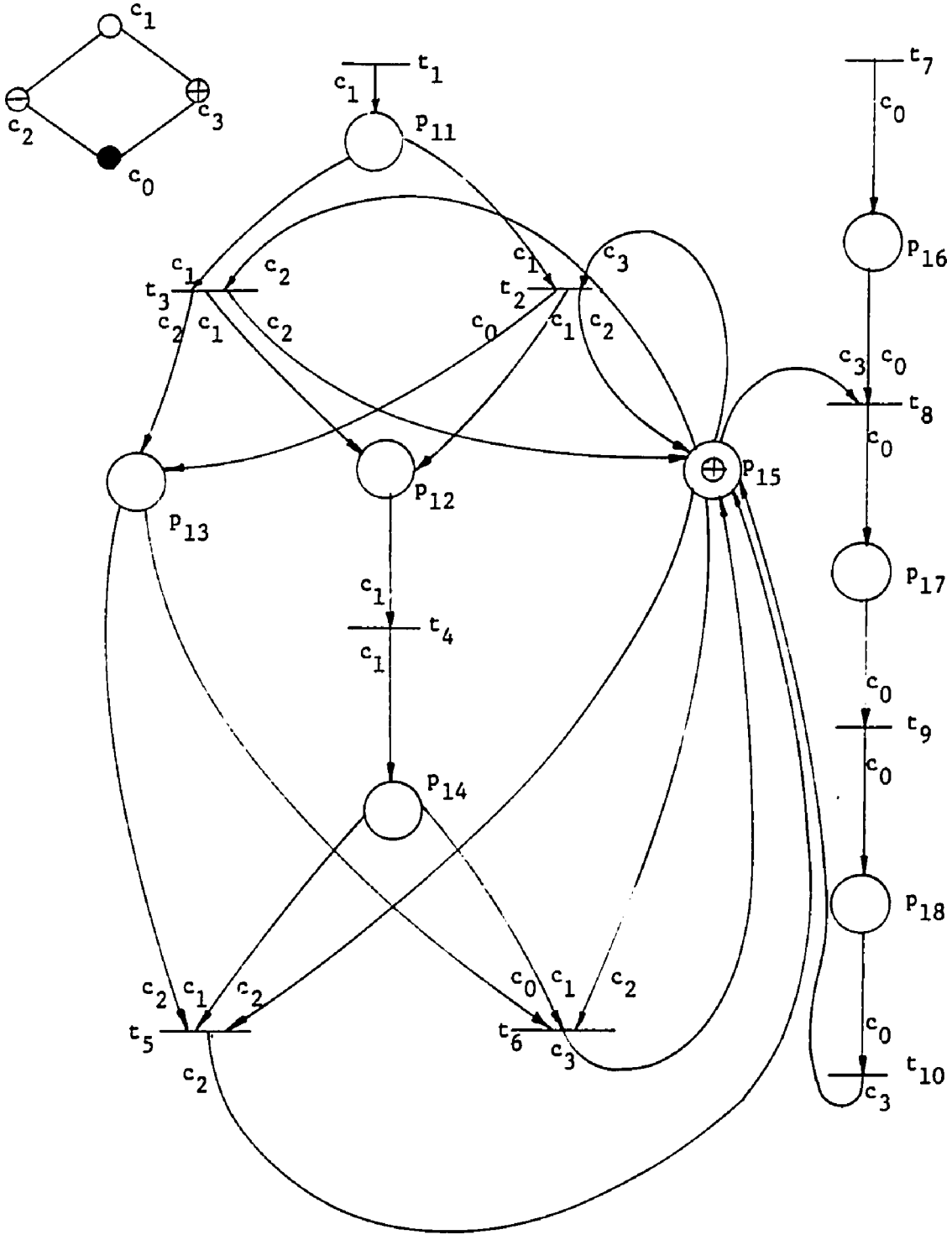


Figure 2.4.4

C-CPM Representation of the Readers-Writers Synchronization Problem

using the C-CPM is exhibited in Figure 2.4.4. There, transitions t_1 and t_7 "generate" an unlimited number of readers and writers, i.e. model the unrestricted arrival of readers and writers to the critical region. Transition t_2 models the case where the first reader enters the critical region while transition t_3 models the subsequent entrance of other reader processes in the critical region. Conversely, transition t_5 simulates the exit of reader processes from the critical region while other reader processes are still accessing the shared resource. Transition t_6 models the exit of the last reader from the critical region. On the other hand, transitions t_8 and t_{10} model the entrance and exit of a writer from the critical region, respectively. Transitions t_4 and t_9 represent the actual read and write operations, respectively. The markings of the places p_{11} and p_{16} indicate the number of reader and writer processes, respectively, waiting to enter the critical region. Place p_{13} plays the same role as the variable readcount in the original solution, namely indicates the number of reader processes currently accessing the resource. Place p_{15} indicates the status of the critical region and thus, combines the roles of the variable readcount and of the semaphore w. If a writer enters the critical region, i.e. transition t_8 fires, the token of color c_3 in place p_{15} is consumed without being replaced, thus inhibiting any readers from entering the critical region. This token is restored only after transition t_{10} has fired and thus, no reader, or for that matter another writer, can gain access to the resource while a writer is using it.

On the other hand, after a reader has entered the critical region, i.e. transition t_2 has fired, place p_{15} contains a token of color c_2. This color marking maintains transition t_8 disabled, thus preventing any writer from entering the critical region while readers are accessing the shared resource. We note that transition t_2 places a token of color c_o in place p_{13} while subsequent readers entering the critical region, i.e. subsequent firings of transition t_3, will place tokens of color c_2 in the same place. On the other hand, transition t_5 can select as enabling color from place p_{13} only the color c_2

while transition t_6 can select as enabling color from the same place only the color c_o, according to Definitions 2.2.2 and 2.4.1. If there are several readers in the critical region, then t_5 and t_6 will always become enabled simultaneously. Transition t_5, however, will have higher priority than transition t_6. Consequently, transition t_6 can fire only if transition t_5 is disabled which occurs exactly when the last reader exists from the critical region. The firing of transition t_6 restores the token of color c_3 in place p_{15}, and thus allows (after all readers have left the critical region) the writer processes to eventually gain control over the shared resource. We note that if several readers and writers are waiting to enter the critical region (places p_{11} and p_{16} are both not empty) but no readers nor writers are in the critical region, then there exists a conflict at place p_{15} for the token of color c_3 among transitions t_2 and t_8. Since t_2 has higher priority than t_8, readers will gain first access to the resource.

It should be clear by now that the C-CPM offers a simple and correct representation of the interaction between readers and writers. It also becomes apparent that the introduction of colors and of priority rules in resolving conflicts opens new possibilities with respect to the representation capabilities of the Petri Net Model. As will be seen later on, the fact that the transitions may dynamically change their priorities adds to the versatility of the modelling capabilities of the C-CPM. We shall examine in detail the representation power of the C-CPM in Chapter IV.

Section 2.5 THE EC-COLORED PETRI MODEL

Originally, we were led towards the development of this particular class of the CPM by yet another synchronization problem presented in [KOSA-73]. The synchronization problem we are referring to involves two producer processes, P_1 and P_2, and two consumer processes, C_1 and C_2, which share a common buffer. The buffer is accessed in a LIFO mode. Thus, at any instant, a producer can be activated and it will produce and deposit an item in the top position of the buffer. If the buffer is not empty, either of the two consumer processes can be

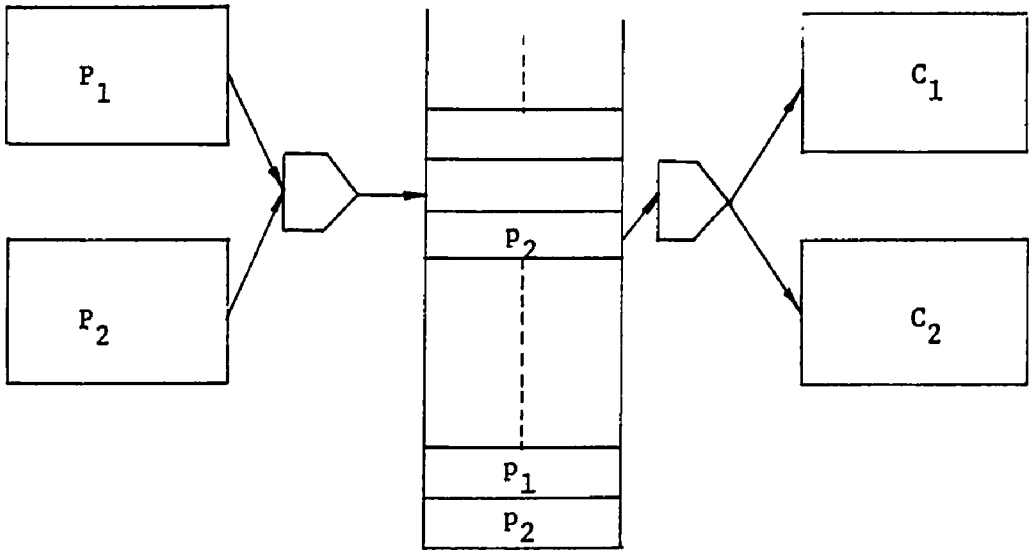


Figure 2.5.1

Schematical Representation of a Producer-Consumer Synchronization Problem

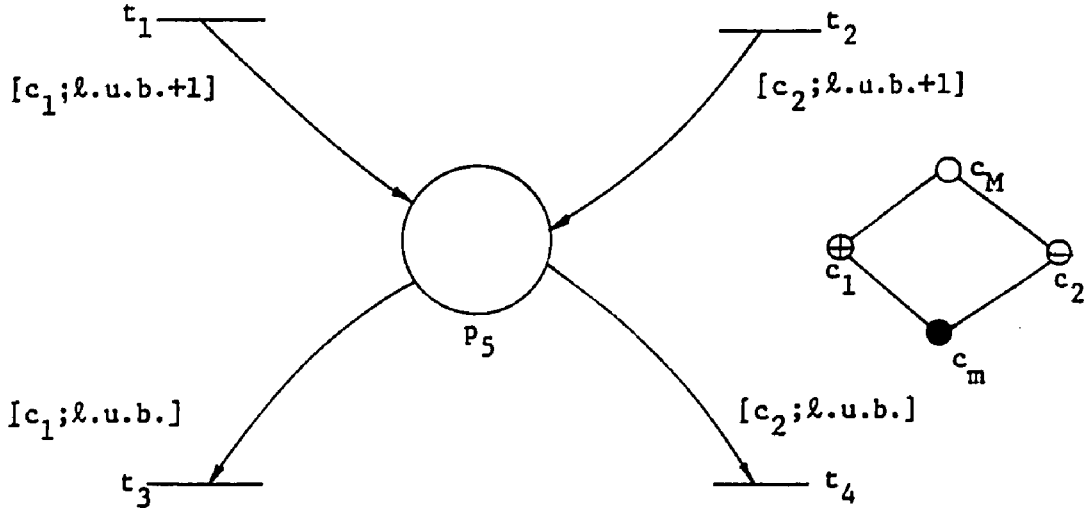


Figure 2.5.2

EC-CPM Representation of the Producer-Consumer Synchronization Problem of Figure 2.5.1

activated and it will attempt to consume the item currently at the top of the buffer. Nevertheless, consumer C_i ($i = 1, 2$) is allowed to consume the item in the top position of the buffer only if it was produced by producer process P_i. This producer-consumer synchronization system is represented schematically in Figure 2.5.1.

The coordination problem described above involves two characteristic aspects which must be incorporated in any correct modelling attempt:

a) The consumer and producer processes are organized in pairs and consequently, a consumer is allowed to consume from the common buffer only those items which were produced by its associate producer process. Therefore, each item in the shared buffer must uniquely identify the corresponding producer.

b) The order of arrival of the items in the buffer must be recorded for subsequent use in the coordination of the consumer processes or, equivalently, in the policy for the retrieval of items from the buffer.

This synchronization problem can easily be extended to the case where there are m producer processes ($2 \leq m < \infty$), k consumer processes ($2 \leq k < \infty$) and where more than one consumer processes may be associated with the same producer process. Also, the policy for the retrieval of items from the shared buffer can be modified from LIFO to FIFO while preserving the basic characteristics of the original problem.

As shown in [KOSA-73], this particular consumer-producer coordination problem cannot be represented by a Petri Net Model. Again, in view of the equivalence relations among formal models of concurrent systems presented in [PETE-73] and [AGAR-75], this assertion can be extended to other well-known formal models such as the Complex Graph Model of Computation, the Finite State Parallel Program Schemata, the P-Nets, the Vector Addition and Vector Replacement Systems, etc. On the other hand, we notice that the modelling characteristic a) mentioned above can easily be implemented in the C-CPM through the use of colored tokens. Let us, however, mention at this point a drawback

in the way the colored tokens are handled by the C-CPM. Consider an arbitrary place p_j in some color marking CM^i. Let us assume that the color bag of p_j has the following configuration: $C_j^i = < c_1^2\ c_4^5\ c_6 >$ where c_1, c_4 and c_6 are colors of the finite color set C associated with the respective C-CPM. In most cases, it is impossible to determine the order in which the particular tokens of colors c_1, c_4 and c_6 have arrived in p_j. In view of the modelling characteristic b) indicated earlier, such information is essential for a natural and correct modelling of the synchronization problem we have described at the beginning of this section. As will be proved later on, this producer-consumer synchronization problem cannot be modelled in a manner which preserves the "natural structure" of the system even by formal models which are strictly more powerful than the Petri Net Model, such as the Priority Petri Model ([HACK-75]), the Extended Petri Model ([AGAR-75]) and the C-CPM.

In what follows, we shall define the EC-CPM, a class of the CPM which can easily handle this type of synchronization situations while preserving the natural structure of the system. The EC-CPM is defined such that it has the capacity to "remember" the order of arrival of colored tokens in the places of a net. As will be seen later on, the availability of this kind of information positively influences the representation capabilities of the EC-Colored Petri Model.

The sets of colors which could be associated with the C-CPM were lattices $C = (X,\leq)$, where X was restricted to a finite set. Let us now consider such a finite lattice $C = (X,\leq)$ with a special configuration:

(I) $X = \{c_1, \ldots, c_n, c_m, c_M\}$

(II) The partial ordering $\leq$ defined on X is such that:

IIa. For any $c_i \in X$, $1 \leq i \leq n$, we have $c_m < c_i < c_M$.

IIb. For any $c_i \in X$ and any $c_j \in X$, where $1 \leq i \leq n$, $1 \leq j \leq n$ and $i \neq j$, we have $c_i \not\leq c_j$ and $c_j \not\leq c_i$, i.e. the elements of the set $\{c_1, \ldots, c_n\}$ are incomparable under the partial ordering $\leq$.

We shall now define an extension of the color sets $C = (X,\leq)$ which satisfy conditions (I) and (II) mentioned above. Let $Z^+ = \{1, 2, 3, \ldots\}$ be the set of positive integers.

Definition 2.5.1

Given a finite color set $C = (X,\leq)$ which satisfies conditions (I) and (II) specified above, the extension of the color set C is a lattice $U(C) = (U, \overline{\leq})$ where:

1. $U = \{(c, m) \mid c \in X, m \in Z^+\}$, i.e. $U = X \times Z^+$.
2. $\overline{\leq}$ is a partial ordering relation defined on the set U as follows:

If $u_i = (c_i, m_i) \in U$ and $u_j = (c_j, m_j) \in U$, then $u_i \overline{\leq} u_j$ if and only if $c_i \leq c_j$ in C and $m_i \underline{\alpha} m_j$ (the latter $\underline{\alpha}$ relation denotes the usual "greater than or equal to" ordering relation of the positive integers).

For convenience, we shall continue to call the two-tuples of U(C) colors.

Note: The extension U(C) of some color set $C = (X,\leq)$ actually represents the direct product of the latices $C = (X,\leq)$ and $(Z^+,\underline{\alpha})$. According to Theorem 7, Section I.4 of [BIRK-67], U(C) is certainly a lattice. By the use of positive integers as the second coordinate of the elements in U(C), the extension of a color set C becomes in fact an infinite set of colors. Figure 2.5.3 exhibits a typical color set U(C).

Let us consider a GPN $N = (T, P, I, O)$ and let $C = (X,\leq)$ be some finite color set which satisfies the conditions (I) and (II) specified earlier. In the following definitions, we shall assume that the extension U(C) of the color set C is the color set associated with N, rather than C. For the sake of clarity, we shall adjust some of the definitions pertinent to the CPM to this special case where the color set is two-dimensional and infinite.

Definition 2.5.2

Given an arbitrary marking M^i of N, the color bag of some place $p_j \in P$ is defined to be:

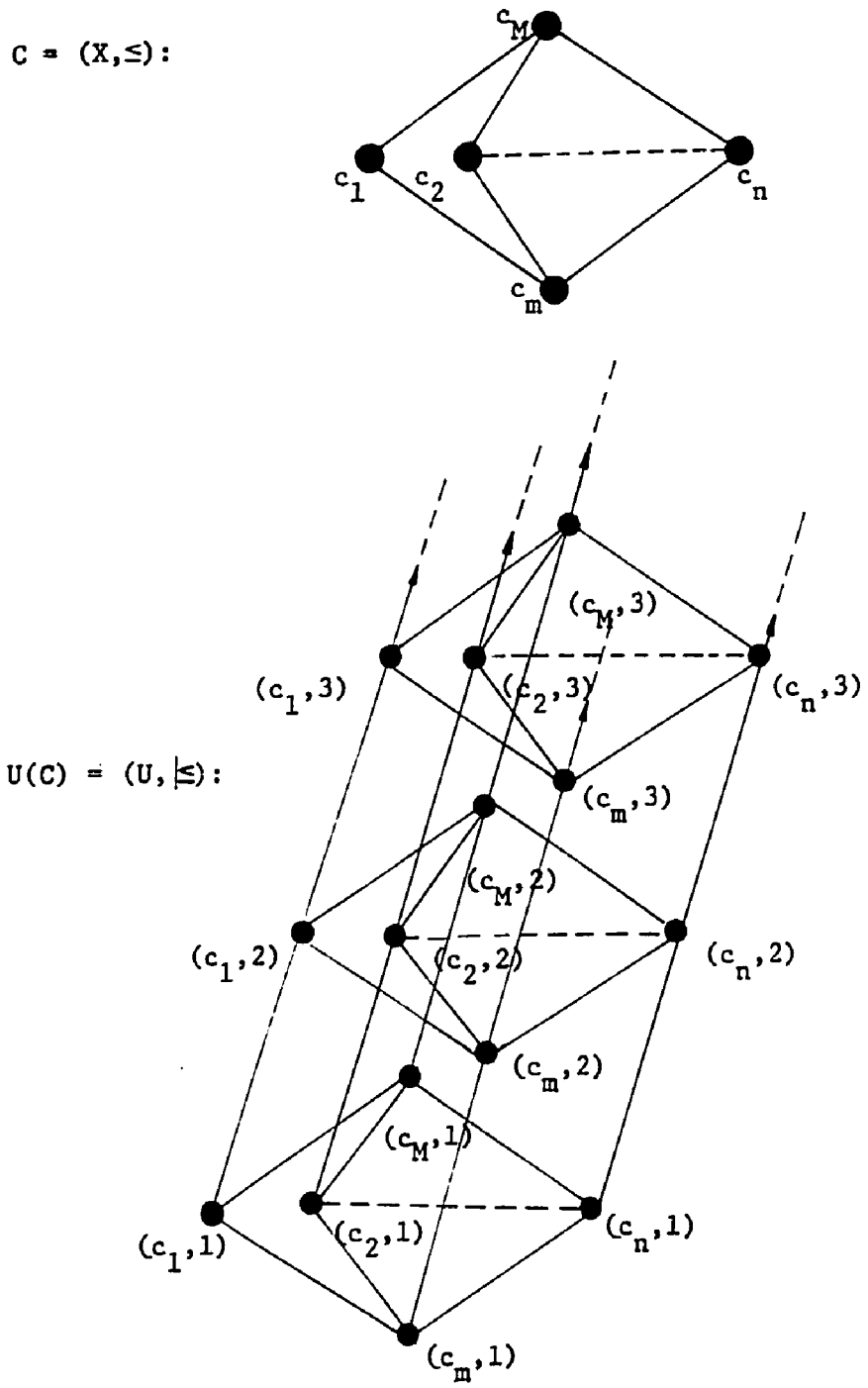


Figure 2.5.3

Example of the Extension of a Color Set C.

$$U_j^i = < (c_r, m)^n \mid c_r \in X,\ m \in Z^+ \text{ and } (c_r, m) \text{ is the color of } n \text{ tokens present in } p_j \text{ in the marking } M^i >$$

Then:

Definition 2.5.3

Given some marking M^i of N, the color marking of N is a total, single-valued mapping UM^i: $P \rightarrow < U >^*$. Again, $< U >^*$ denotes the bag closure of the set U.

We can proceed now to define the EC-CPM, as a class of the CPM obtained by imposing additional restrictions on the structure of the color set and on the type of mappings admissible as color threshold, color output and priority functions of the CPM.

Definition 2.5.4

An EC-Colored Petri Net (EC-CPN) is a CPN CN = (N, U(C), H, R_E, Q) where:

1. N = (T, P, I, O) is a Generalized Petri Net.

2. U(C) = $(U, \preceq)$ is the extension of some finite color set C = $(X, \leq)$ which satisfies the conditions (I) and (II). U(C) is the color set associated with the GPN N.

3. H is a total, single-valued mapping; H: A → h. A denotes the set of arcs associated with the sets T and P. The range h of the mapping H is the set of functions:

$h = \{h_{jk}^i \mid h_{jk}^i = (f_{jk}^i, g_{jk}^i) = H(a_{jk}^i)$ for some arc $a_{jk}^i \in A$,

where f_{jk}^i: $\mathcal{P}(\{c_1, \ldots, c_n\}) \rightarrow c_{jk}^i$ with $c_{jk}^i \in X - \{c_m, c_M\}$

and g_{jk}^i: $\mathcal{P}(Z^+) \rightarrow Z\}$.

Z denotes the set of integers.

The functions g_{jk}^i are further restricted as follows:

3a. If a_{jk}^i is an input arc, then $g_{jk}^i(B)$ is one of the following functions:

$$\ell.u.b.(B) - s_i \quad ; \quad s_i \in Z^o$$
$$g.\ell.b.(B) + m_i \quad ; \quad m_i \in Z^o$$
$$z_i \quad ; \quad z_i \in Z^+$$

where $B \in \mathcal{P}(Z^+)$.

3b. If a^i_{jk} is an output arc, then $g^i_{jk}(B)$ is one of the following functions:

$$\ell.u.b.(B) + s_i \quad ; \quad s_i \in Z^o$$
$$g.\ell.b.(B) + m_i \quad ; \quad m_i \in Z^o$$
$$z_i \quad ; \quad z_i \in Z^+$$

where $B \in \mathcal{P}(Z^+)$.

Here, for any set $B \in \mathcal{P}(Z^+)$, $\ell.u.b.(B)$ and $g.\ell.b.(B)$ denote the lowest upper-bound and the greatest lower-bound of the set B, respectively, and refer to the usual "greater than or equal to" ordering relation on Z^+. We shall make the convention that $\ell.u.b.(B) = g.\ell.b.(B) = 1$ if $B = \phi$.

4. $R_E = \{(u_i, u_j) \mid u_i \in U, u_j \in U \text{ and } u_i = u_j\}$. Since all models in the EC-CPM class will use the relation R_E, we shall omit it when specifying a particular EC-CPN in the sequel.

5. $Q: T \to \{q\}$ is a total, single-valued mapping, where q is the function defined by

$$q: \mathcal{P}(U) \to (c_M, 1)$$

In the sequel, we shall omit the mapping Q from the definition of any EC-CPN with the understanding that the mapping Q for the respective EC-CPN obeys the definition given above.

As one can imagine, the mapping H associates with each arc $a^i_{jk} \in A$ a two-tuple (f^i_{jk}, g^i_{jk}) which is called the color threshold function or the color output function of the arc a^i_{jk}, depending on whether a^i_{jk} is an input arc or an output arc, respectively. We shall briefly explain now how the selection of the enabling colors is made in the case of the EC-CPM. Let us consider an EC-CPN $CN = (N, U(C), H)$ in some given color marking UM^i. Let $t_k \in T$ be an arbitrary transition of N and suppose $p_j \in \mathcal{D}(I_k)$. Let us also assume that $I(p_j, t_k) = m$ and, consequently, that there are m input arcs from p_j to t_k, $a^1_{jk}, \ldots, a^m_{jk}$.

We notice that if U^i_j is the color bag of p_j in the color marking UM^i, then $\mathcal{D}(U^i_j)$ is in fact a binary relation. We can define $\mathcal{D}^1(U^i_j)$ and $\mathcal{D}^2(U^i_j)$ as the domain and the range, respectively, of the relation $\mathcal{D}(U^i_j)$:

$\mathcal{D}^1(U_j^i) = \{c \mid c \in X - \{c_m, c_M\}$ and $(c, m) \in \mathcal{D}(U_j^i)$ for some $m \in Z^+\}$

$\mathcal{D}^2(U_j^i) = \{m \mid m \in Z^+$ and $(c, m) \in \mathcal{D}(U_j^i)$ for some $c \in X - \{c_m, c_M\}\}$

Let us also define the following subsets of $\mathcal{D}^1(U_j^i)$, for $1 \leq r \leq m$:

$$\mathcal{D}_{jk}^r(U_j^i) = \{c \mid (c, g_{jk}^r(\mathcal{D}^2(U_j^i))) \in \mathcal{D}(U_j^i)\}$$

The mapping g_{jk}^r selects the second coordinate of the enabling color for the input arc a_{jk}^r, $1 \leq r \leq I(p_j, t_k)$, in the given color marking UM^i. The first coordinate of the enabling color for a_{jk}^r is produced by the mapping f_{jk}^r and is the color $f_{jk}^r(\mathcal{D}_{jk}^r(U_j^i)) = c_{jk}^r$. Then,

Definition 2.5.5

A color (c, n) of a token present in place p_j in the color marking UM^i is said to be the enabling color of transition t_k on input arc a_{jk}^r if $(c, n) = (c_{jk}^r, g_{jk}^r(\mathcal{D}^2(U_j^i)))$.

Due to the type of functions f_{jk}^r and g_{jk}^r and to the relation R_E employed in the EC-CPM class, unlike the C-CPM class, the enabling color of t_k on some input arc a_{jk}^r in a given color marking UM^i, is unique. This fact greatly simplifies the execution of the EC-CPM.

Let us denote by θ_{jk}^i and θ_k^i the bag of enabling colors of t_k from input place p_j and the bag of enabling colors of transition t_k, respectively. The domain $\mathcal{D}(\theta_k^i)$ of the bag of enabling colors of transition t_k is a binary relation. Let $\mathcal{D}^1(\theta_k^i)$ be the domain of the relation $\mathcal{D}(\theta_k^i)$.

Consider an arbitrary transition t_k and suppose $p_r \in \mathcal{D}(O_k)$. Let us assume that $O(t_k, p_r) = n$ and that there are n output arcs $a_{kr}^1, \ldots, a_{kr}^n$ from t_k to p_r. For each such output arc a_{kr}^s, the pair $(f_{kr}^s, g_{kr}^s) = H(a_{kr}^s)$ determines the color of the token placed by a_{kr}^s in the output place p_r in case t_k fires. The first coordinate of the color of the token produced by t_k on output arc a_{kr}^s, $1 \leq s \leq O(t_k, p_r)$, is $f_{kr}^s(\mathcal{D}^1(\theta_k^i)) = c_{kr}^s$, where $c_{kr}^s \in X - \{c_m, c_M\}$. The second coordinate of the color of the token produced by t_k on the output arc a_{kr}^s is given by $g_{kr}^s(\mathcal{D}^2(U_r^{i-}))$, where:

$$U_r^{i-} = \begin{cases} U_r^i - \theta_{rk}^i & \text{if } p_r \in \mathscr{D}(I_k) \\ U_r^i & \text{otherwise.} \end{cases}$$

We should mention here that the role of the g_{jk}^r functions is merely to structure the color bag of a given place in a desired fashion (LIFO and FIFO included). Thus, the g_{jk}^r function of some input arc a_{jk}^r permits the respective input arc to select its enabling color only from within the bag of tokens placed in a certain position in the color bag of the input place p_j. On the other hand, an output arc a_{kr}^s uses the function g_{kr}^s associated with it to place a token in a certain position within the color bag of the output place p_r. The position of a token in some color bag is considered to be given by the second coordinate of its color.

Finally:

Definition 2.5.6

An EC-Colored Petri Model (EC-CPM) is a system $ECP = (CN, UM^o)$ where:

1. $CN = (N, U(C), H)$ is an EC-Colored Petri Net
2. UM^o is the initial color marking of CN. For any place $p_j \in P$, $\mathscr{D}^1(U_j^o) \cap \{c_m, c_M\} = \phi$.

Let us consider next the dynamic behavior of the EC-CPM. The definitions of "enabled transition" and of "firing of a transition" given in Section 2.3 for the CPM can immediately be particularized for the case of the EC-CPM. Note that in Definition 2.5.4, the priority function q was defined from the power set $\mathscr{P}(U)$ into $(c_M, 1)$. Thus, given some EC-CPN $CN = (N, U(C), H)$ in some color marking UM^i, for any two enabled transitions $t_k \in E^i$ and $t_s \in E^i$, we have:

$$q(\mathscr{D}(\theta_k^i)) = (c_M, 1) = q(\mathscr{D}(\theta_s^i))$$

Consequently, all transitions enabled in some arbitrary color marking UM^i have the same priority to fire. Certainly, the set of enabled transitions E^i may still be partitioned in conflict clusters which in turn may contain several conflict subclusters. Nevertheless, any enabled transition is also a majorant of all conflict subclusters in which it is contained. Therefore, the search for a firable

transition reduces in the case of the EC-CPM to the search for an enabled transition.

It should be clear by now that the places and transitions of an EC-CPM form rather general queueing mechanisms which include widely used policies such as LIFO, FIFO, ranked insertion and any combination thereof. We hope that this approach to the modelling of systems exhibiting concurrency will provide a modelling framework which will preserve the natural process structure of many synchronization structures encountered in real-life. We shall proceed now to model the producer-consumer synchronization problem described at the beginning of this section. The EC-CPM representation of this coordination system is given in Figure 2.5.2. The color threshold and color output functions are marked adjacent to the corresponding arcs. Transitions t_1 and t_2 model the two producer processes P_1 and P_2, respectively while transitions t_3 and t_4 represent the consumer processes C_1 and C_2, respectively. The place p_5 models the shared buffer.

We shall conclude this section with another modelling example using the EC-CPM. We have pointed out earlier that for any EC-CPN CN in some color marking UM^i, all enabled transitions have the same priority to fire in UM^i. Thus, the search for a transition firable in UM^i does not involve any priority hierarchy among the enabled transitions. Nevertheless, one can still model in the framework of the EC-CPM synchronization systems which inherently incorporate priority hierarchies among the component subsystems. In order to illustrate this assertion, we have selected for modelling the following coordination structure suggested in [CERF-72]. The synchronization system contains k classes of accessors which compete for a shared resource (e.g. a data base). The accessors can gain control over the resource under the following constraints:

i) An accessor belonging to the priority class j, $1 \leq j \leq k - 1$, can access the resource only if no accessor belonging to some priority class i, $i > j$, is waiting.

ii) The resource is to be accessed in a mutually exclusive fashion, that is no two accessors (even from the same priority class)

should simultaneously have control over the shared resource.

This synchronization problem actually represents a generalization of the readers-writers synchronization problem presented in the previous section. We note that in [CERF-72], a correct solution to the synchronization problem stated above could be given using the Complex Graph Model of Computation only after the number of accessors in each class was bounded by some given, fixed positive integer.

The EC-CPM representation of this synchronization problem is exhibited in Figure 2.5.4. We note that in the figure certain color threshold and/or color output functions do not have their second coordinate indicated (i.e. the corresponding "g" function is missing). In order to simplify notations, we have made the convention that whenever for an arbitrary arc $a^i_{jk_i}$ the corresponding function $g^i_{jk}: \mathcal{P}(Z^+) \to Z$ is of the form $g^i_{jk}(B) = 1$ for any set $B \in \mathcal{P}(Z^+)$, then it may be omitted from the specification of the respective EC-CPN. This convention will be used throughout the remainder of this report.

In Figure 2.5.4 each transition t_i, $1 \leq i \leq k$, generates an unlimited number of accessors, i.e. models the unrestricted arrival of accessors from the priority class i to the critical region. On the other hand, the transitions t_i, $k + 1 \leq i \leq 2k$, model the entrance of an accessor belonging to class $j = i - k$ to the critical region. It is easy to see that the firing of a transition from this group prohibits other accessors from gaining control over the resource. The transitions t_i, $2k + 1 \leq i \leq 3k$, model the actual access of an accessor from the priority class $j = i - 2k$ to the shared resource while the transitions t_i, $3k + 1 \leq i \leq 4k$, represent the exit of an accessor belonging to the class $j = i - 3k$ from the critical region. The firing of a transition from the latter group reenables the waiting accessors to eventually obtain control over the resource. It should be obvious by now that our solution satisfies the mutual exclusion requirement regarding the presence of accessors in the critical region.

Notice that place p_{4k+1} is "structured" as a list with k ranks. The occurrence in the system of accessors belonging to the i-th priority class is modelled by transition t_i by placing tokens in the

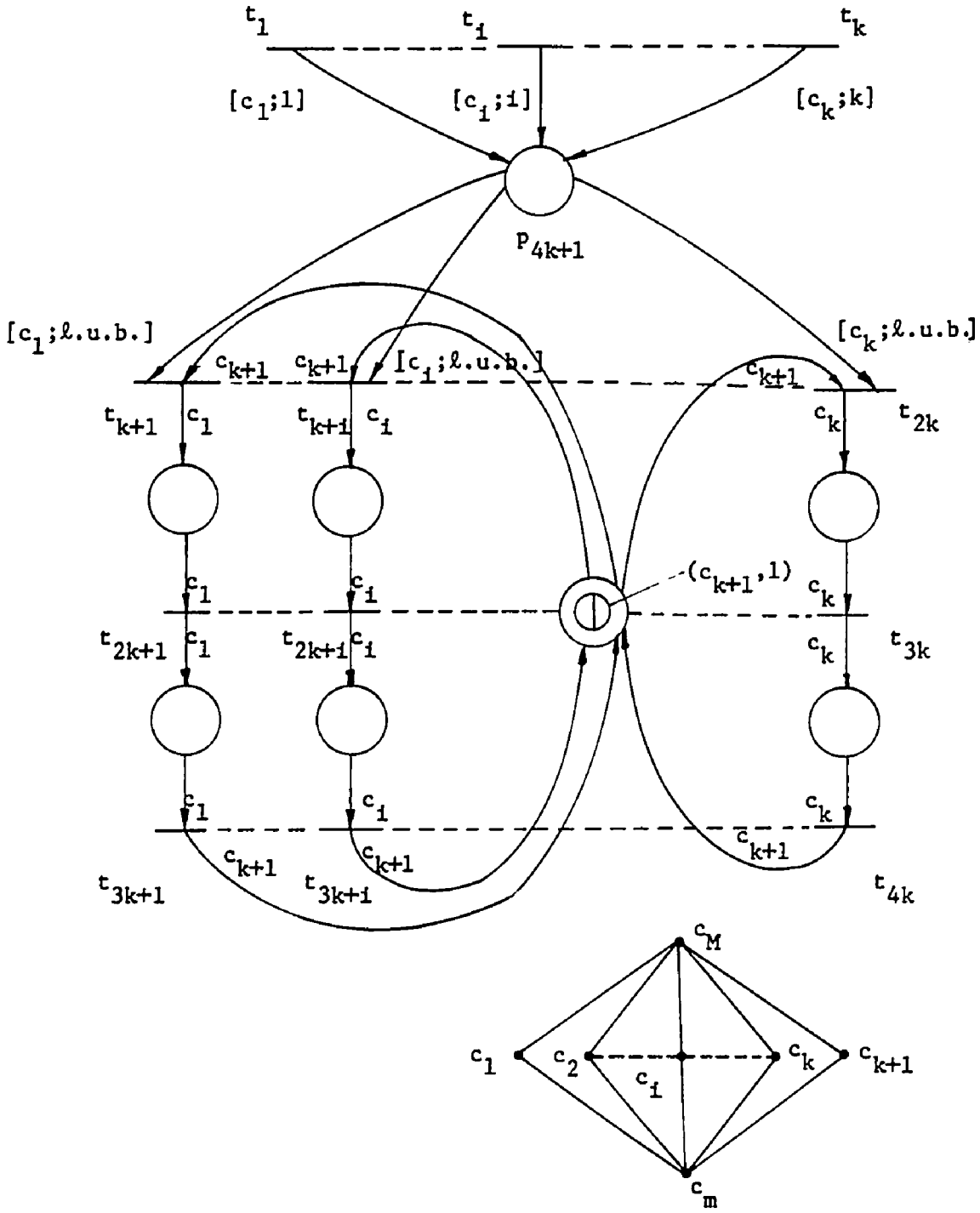


Figure 2.5.4

k-Accessor Classes Synchronization Problem

i-th rank upon firing. On the other hand, the transitions t_{k+i}, $1 \leq i \leq k$, will extract a token only from the highest non-empty rank. If j is the highest non-empty rank, $1 \leq j \leq k$, then from the latter group only transition t_{k+j} is enabled. It can easily be verified that the EC-CPM of Figure 2.5.4 indeed implements the priority rule required in the statement of the problem. Thus, the EC-CPM can correctly model synchronization structures which involve priority hierarchies among the coordinated processes. The representation capabilities of the EC-CPM will be analyzed in great detail in Chapter III. In Chapter IV we shall compare the representation power of the EC-CPM with that of the C-CPM.

CHAPTER III

THE REPRESENTATION CAPABILITIES OF THE EC-COLORED PETRI MODEL

Section 3.1 INTRODUCTORY REMARKS

The Colored Petri Model has emerged from the association of the Generalized Petri Nets with sets of colors as a compact and general model of systems exhibiting concurrency. In Chapter II, two classes of the CPM having simple structures, namely the C-CPM and the EC-CPM, have been defined. Our next objective is to analyze the modelling power of these two classes of the CPM and to find a measure of their representation capabilities, as accurately as possible. Otherwise stated, we would like to get some insight into what is gained, as far as the representation capabilities of the model are concerned, by introducing colored tokens in the Petri Net Model. In Sections 2.4 and 2.5 we have modelled several synchronization problems using the C-CPM or the EC-CPM. As already mentioned, some of those synchronization problems, in particular the producer-consumer coordination problems suggested in [KOSA-73], were proved (also in [KOSA-73]) not to be correctly representable by the Petri Net Model, though there exist more powerful versions of the Petri Net Model such as the Extended Petri Model ([AGAR-75]) or the Priority Petri Model ([HACK-75]) which can represent these synchronization problems. This fact indicates that the use of colored tokens positively affects the representation capabilities of the Petri Net Model and thus encourages one to attempt a more in-depth analysis of the representation power of the C-CPM and of the EC-CPM class.

Unfortunately, the term "representation power" of a formal model of asynchronous concurrent systems is not yet precisely defined and, consequently, there is no well-established method by means of which the range of the class of systems representable by a particular formal model can be precisely determined. One method which in our opinion came closest to providing an answer to this problem was the study of the formal language families which can be generated by the Petri Net

Model ([PETE-73], [HACK-75]). In this context, each transition of a Petri Net can be considered to correspond to the activation of a particular process of the system modelled by the respective net (e.g. each transition can be made to correspond to a computational step of a parallel program). Then, a firing sequence of the net corresponds to a process coordination sequence of the real-life system. From this point of view, the structure of the net, that is the particular arrangement of arcs, places and transitions models the constraints existing in the real-life system, constraints which force the system to execute only certain synchronization sequences over the set of underlying processes and not all possible such sequences. It follows that if the real-life system is correctly modelled by the Petri Net in question, then there exists a one-to-one correspondence between the coordination sequences compatible with the system and the firing sequences of the net which models it. Suppose now that each distinct process occurring in the modelled system receives a distinct label and that all transitions of the net which represent an instance of the same process carry the label of that process. In this way, any firing sequence of the net generates a string over the alphabet formed by the process labels. In this context, a Petri Net can be viewed as an automaton generating a particular language, namely the set of all strings over the process label alphabet corresponding to all possible firing sequences which can be performed by the net. The set of coordination sequences representable by Petri Nets can thus be determined by studying the language families which can be generated by Petri Nets.

We intend to employ this method in our study of the representation capabilities of the C-CPM and of the EC-CPM. We shall first give a few definitions with the intent to formalize the concepts discussed above.

Definition 3.1.1

A Labelled Colored Petri Model is a system $\mathscr{P}_\Sigma = (\mathscr{P}, \Sigma, L)$ where:

1. $\mathscr{P} = (CN, CM^0)$ is a CPM.
2. Σ is a finite alphabet.

3. L is a total, single-valued mapping from the set of transitions T of $\mathscr{P}$ into Σ; L: T → Σ. L is called the <u>labelling function</u> of $\mathscr{P}_\Sigma$.

The definition of the labelling function can easily be extended to firing sequences. Let $t\gamma$ be a firing sequence of $\mathscr{P}$ from some arbitrary color marking CM^i (Definition 2.3.7). Then $L(t\gamma)$ is defined recursively as follows:

$$L(t\gamma) = \begin{cases} L(t) & \text{if } \gamma = \lambda \\ L(t)L(\gamma) & \text{otherwise} \end{cases}$$

where $L(t)L(\gamma)$ represents the concatenation of the strings $L(t)$ and $L(\gamma)$. By convention, $L(\gamma) = \lambda$ if and only if $\gamma = \lambda$, the empty firing sequence. We shall call $L(\gamma)$ the <u>label sequence</u> of the firing sequence γ. Obviously, the label sequences corresponding to the firing sequences compatible with some Labelled CPM form a language over the alphabet Σ.

In what follows, we shall often use the term "family of languages." If $\mathscr{A}$ is a family of languages, we shall assume that:

i) there exists $A \in \mathscr{A}$ such that $A \neq \phi$.

ii) if $A \in \mathscr{A}$, then there exists a finite alphabet $\mathscr{a}$ such that $A \subseteq \mathscr{a}^*$.

In accordance with this convention, let us consider the following families of languages:

<u>Definition 3.1.2</u>

$\mathscr{S}$ is the family of languages such that $S \in \mathscr{S}$ if and only if there exists a CPM $\mathscr{P} = (CN, CM^o)$ where $CN = (N, C, F, R, Q)$, $N = (T, P, I, O)$ and $S = S(CM^o)$. Obviously, $S \subseteq T^*$.

$\mathscr{S}_o$ is the family of languages such that $S \in \mathscr{S}_o$ if and only if there exists a CPM $\mathscr{P} = (CN, CM^o)$ and a final color marking of $\mathscr{P}$, $CM^f \neq CM^o$, such that $S = T(CM^o, CM^f)$. Obviously, $S \subseteq T^*$.

Notice that the labelling function L as specified in Definition 3.1.1 is a non-erasing homomorphism from T^* into Σ^*, whose range

contains only one-symbol strings. We shall call this type of homomorphism a λ-free renaming. For any family of languages $\mathscr{A}$, we shall define the image of $\mathscr{A}$ under λ-free renaming to be the set:

$\mathscr{I}(\mathscr{A}) = \{L(A) \mid A \in \mathscr{A}$ and L is a λ-free renaming on A$\}$.

The families of languages of interest to our study are then defined as follows:

Definition 3.1.3

The family of computation sequence sets of the CPM is the family of languages Λ, where $\Lambda = \mathscr{I}(\mathscr{P})$. Similarly, the family of terminal computation sequence sets of the CPM is the family of languages Λ_o, where $\Lambda_o = \mathscr{I}(\mathscr{P}_o)$.

Note: The languages in Λ_o cannot contain the empty string λ. This constraint is motivated by the convenience which it introduces in certain proofs and constructs which will follow. The language families introduced above actually represent synchronization sequences compatible with asynchronous concurrent systems modelled by the CPM. From this point of view, the empty string and, equivalently, the "empty coordination sequence" are not of much practical importance. Thus, this constraint is not exceedingly severe and does not influence the generality of the results obtained. As opposed to Λ_o, the languages in the Λ family must contain the empty string.

Similarly defined language families were studied in [HACK-75] with respect to the Petri Net Model, the Extended Petri Model and the Priority Petri Model. Thus, a common basis of comparison between these models, the C-CPM and the EC-CPM is provided.

We point out that describing the coordination sequences of a system by means of label sequences corresponds to a totally ordered execution of the respective coordination sequences. Note, however, that the label sequences which can be generated by a net are the images under some λ-free renaming of the firing sequences which can be executed by that net. A generally accepted execution rule incorporated in the Petri Net Model and in the various modified versions of it reported in the literature is that no matter how many transitions

are simultaneously enabled, only one of them is chosen to be fired. Consequently, firing sequences are totally ordered sequences of transition names. The execution rule mentioned above is based on the assumption that the operations of a system are modelled as instantaneous and non-simultaneous events. This assumption appears not to restrict the adequacy of the Petri Net Model and of its variants for the modelling of systems exhibiting concurrency. For a detailed discussion of this claim, the reader is referred to Section 2.4.1 of [PETE-73].

All the definitions given in this section can immediately be translated for the C-CPM class and for the EC-CPM class. We shall denote by Λ(C-CPM), Λ_o(C-CPM), Λ(EC-CPM), Λ_o(EC-CPM) the Λ and Λ_o language families generated by the C-CPM and EC-CPM, respectively.

In the remainder of this chapter, we shall study in detail the closure properties and the extent of the family of computation sequence sets and of the family of terminal computation sequence sets corresponding to the EC-CPM. Similar problems, related to the C-CPM, will form the subject of Chapter IV.

Section 3.2 COMPOSITION PROPERTIES OF THE EC-CPM

Let us first investigate the closure of the language families Λ(EC-CPM) and Λ_o(EC-CPM) under the operation of concurrency, denoted by Δ. Given some finite alphabet Σ, the operator Δ is recursively defined (in terms of regular expressions) as follows:

i. $a \Delta \lambda = \lambda \Delta a = a$ for all $a \in \Sigma$

ii. $a'w' \Delta a''w'' = a'(w' \Delta a''w'') + a''(a'w' \Delta w'')$ for all a', a'' in Σ and all w', w'' in Σ^*. If W' and W'' are two languages over the alphabet Σ, then:

$$W' \Delta W'' = \{w' \Delta w'' \mid w' \in W', w'' \in W''\}$$

In words, if $w' \in W'$ and $w'' \in W''$, then the language $W' \Delta W''$ contains all possible interleavings of the strings w' and w''.

Given a Labelled EC-CPM ECP_Σ, we denote by $\Lambda(ECP_\Sigma)$ the language in Λ(EC-CPM) generated by ECP_Σ. If a final color marking UM^f of ECP_Σ is also given, then $\Lambda_o(ECP_\Sigma, UM^f)$ denotes the language in

Λ_o(EC-CPM) generated by ECP_Σ (similar notational conventions will be used in connection with the C-CPM, as well).

Theorem 3.2.1

Λ(EC-CPM) and Λ_o(EC-CPM) are closed under the concurrency operation.

Proof: Let $ECP_{\Sigma'} = (ECP', \Sigma', L')$ and $ECP_{\Sigma''} = (ECP'', \Sigma'', L'')$ be two Labelled EC-CPM's, where $ECP' = (CN', UM^{o'})$ and $ECP'' = (CN'', UM^{o''})$. Let $CN' = (N', U(C'), H')$ and $CN'' = (N'', U(C''), H'')$ where $N' = (T', P', I', O')$ and $N'' = (T'', P'', I'', O'')$.

We construct a Labelled EC-CPM $ECP_\Sigma = (ECP, \Sigma, L)$ such that $\Lambda(ECP_\Sigma) = \Lambda(ECP_{\Sigma'}) \Delta \Lambda(ECP_{\Sigma''})$. In case the final color markings $UM^{f'}$ and $UM^{f''}$ of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, respectively, are also given, we define a final color marking UM^f of ECP_Σ such that $\Lambda_o(ECP_\Sigma, UM^f) = \Lambda_o(ECP_{\Sigma'}, UM^{f'}) \Delta \Lambda_o(ECP_{\Sigma''}, UM^{f''})$. Let $ECP = (CN, UM^o)$ where $CN = (N, U(C), H)$ and $N = (T, P, I, O)$.

We assume that $T' \cap T'' = \phi$ and $P' \cap P'' = \phi$; these conditions can always be achieved through appropriate renaming.

Let us now juxtapose the Labelled EC-CPM's $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, that is let us consider $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ as parts of the same net. For example, Figure 3.2.3 displays the juxtaposition of the Labelled EC-CPM's of Figure 3.2.1 and 3.2.2. In the figures, the labels of the transitions are indicated underneath the corresponding transition names; i.e. $\underset{a}{t_k}$ means that transition t_k carries the label a.

Next, let us introduce a new place, let it be denoted by p_c, in the net obtained by the juxtaposition of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$. Suppose $C' = (X', \leq')$ and $C'' = (X'', \leq'')$, where $X' = \{c'_1, \ldots, c'_s, c'_m, c'_M\}$ and $X'' = \{c''_1, \ldots, c''_p, c''_m, c''_M\}$. Let us also introduce a new color, c, and let $X = \{c'_1, \ldots, c'_s\} \cup \{c''_1, \ldots, c''_p\} \cup \{c, c_m, c_M\}$. Then, $C = (X, \leq)$ and the partial ordering $\leq$ is defined such that it satisfies conditions (I) and (II) given in Section 2.5. The extention U(C) is the set of colors associated with the Labelled EC-CPM ECP_Σ.

For each transition t_k, $I(p_c, t_k) = O(t_k, p_c) = 1$ and $H(a^1_{ck}) = H(a^1_{kc}) = [c;1]$. Figure 3.2.4 exhibits this construct, as

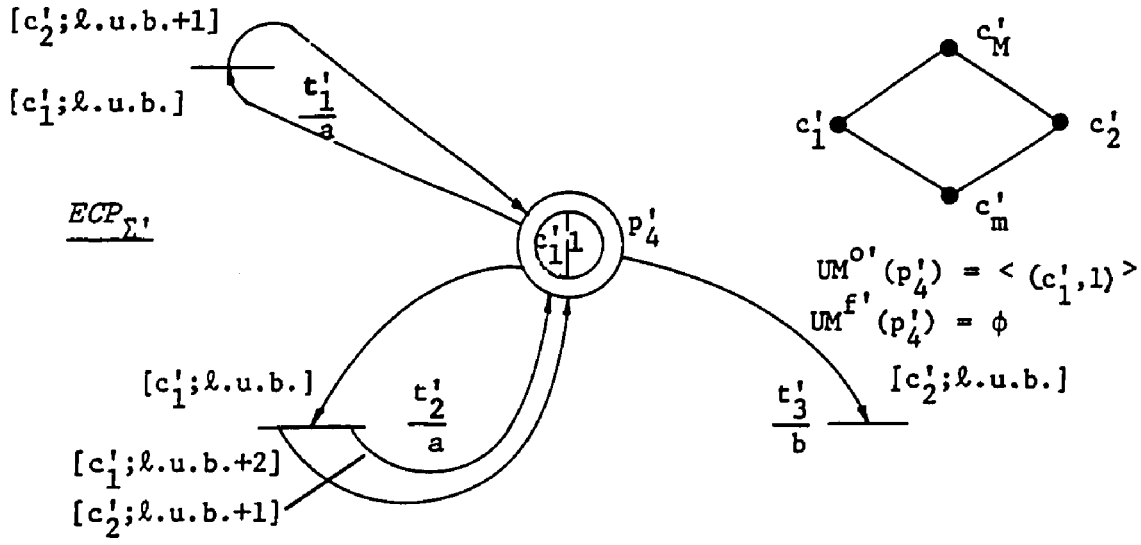


Figure 3.2.1
Sample Labelled EC-CPM

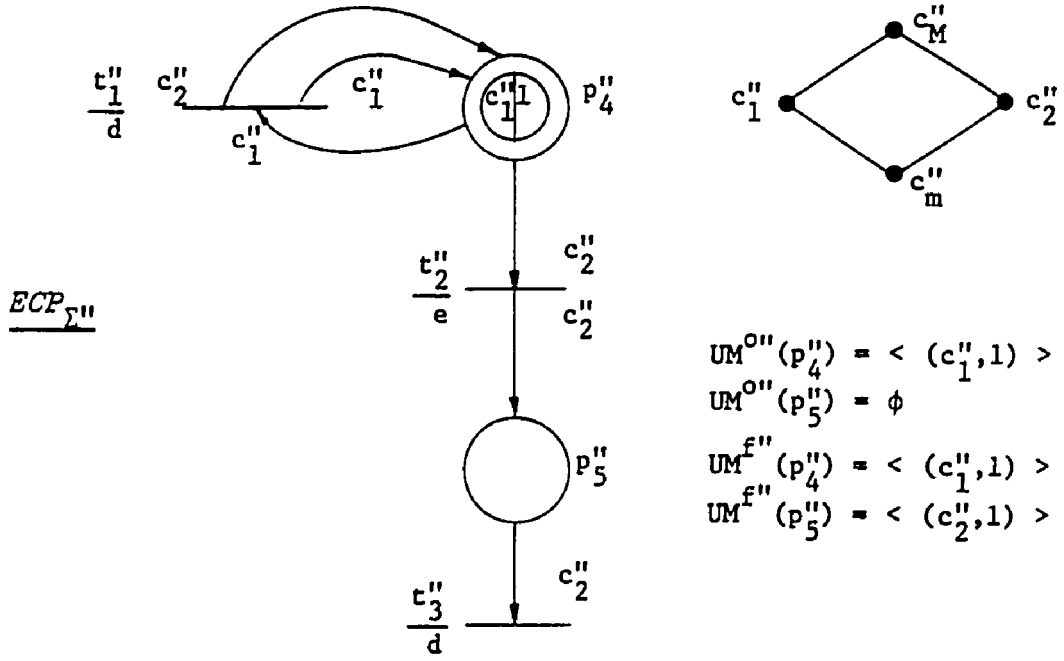


Figure 3.2.2
Sample Labelled EC-CPM

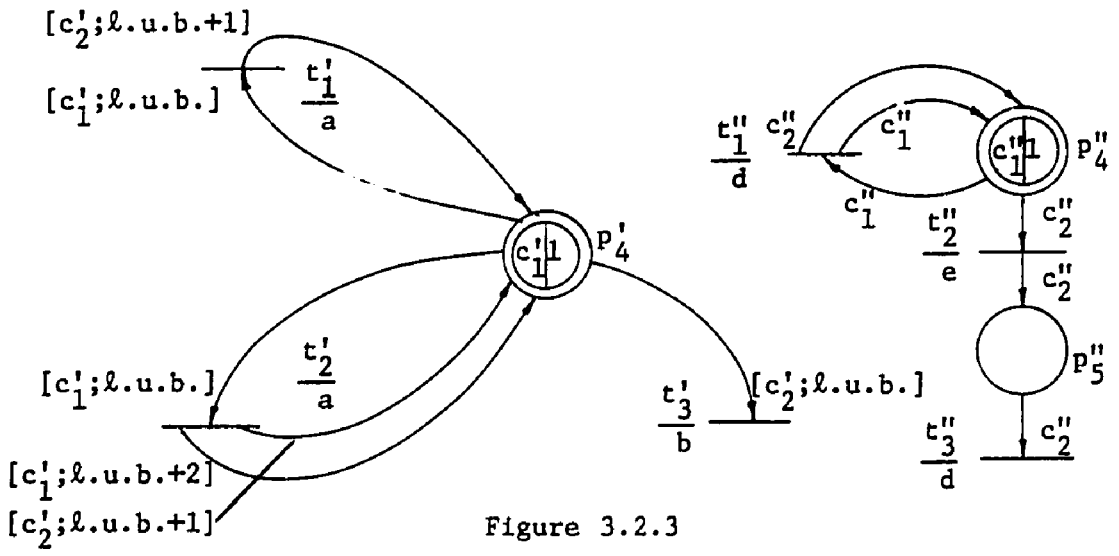


Figure 3.2.3

Juxtaposition of the Labelled EC-CPM's of Figure 3.2.1 and 3.2.2

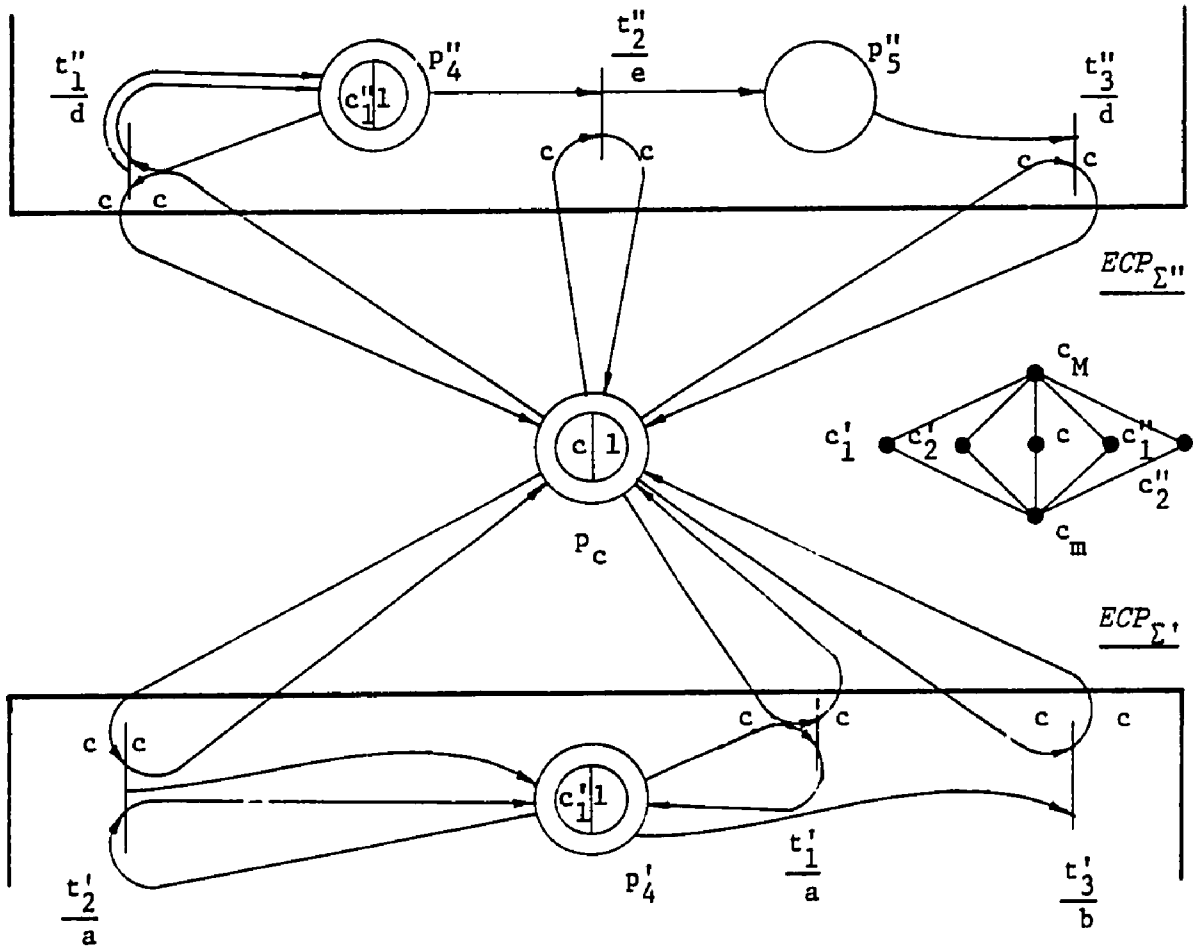


Figure 3.2.4
The Introduction of a Control Place

applied to the Labelled EC-CPM's of Figure 3.2.1 and 3.2.2. We say that each transition "self-loops" on p_c. Alternatively, we sometimes refer to a place, such as p_c, on which all transitions of a net self-loop as a "control place."

Let us now define the initial color marking UM^o of ECP_Σ. We set $UM^o(p_c) = < (c,1) >$ and define the restriction of UM^o to $P' \cup P''$, i.e. $UM^o/P' \cup P''$, by $UM^o/P' \cup P'' = UM^{o'} \nabla UM^{o''}$ (see Figure 3.2.4).

This completes the construction of the Labelled EC-CPM ECP_Σ. The control place p_c does not affect the conditions under which any transition of ECP_Σ is enabled. Consequently, each reachable color marking UM^i of ECP_Σ will satisfy the following conditions:

i. $UM^i(p_c) = UM^o(p_c)$

ii. $UM^i/P' \cup P'' \in R'(UM^{o'}) \nabla R''(UM^{o''})$

Hence, $\Lambda(ECP_\Sigma) = \Lambda(ECP_{\Sigma'}) \Delta \Lambda(ECP_{\Sigma''})$. The final color marking UM^f of ECP_Σ is defined by $UM^f(p_c) = UM^o(p_c)$ and $UM^f/P' \cup P'' = UM^{f'} \nabla UM^{f''}$. Under this condition, $\Lambda_o(ECP_\Sigma, UM^f) = \Lambda_o(ECP_{\Sigma'}, UM^{f'}) \Delta \Lambda_o(ECP_{\Sigma''}, UM^{f''})$. □

The construct of Theorem 3.2.1 allows for the arbitrary interleaving of label sequences generated by $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$. Taking advantage of the control place p_c and of the availability of colored tokens we can, however, "force" the juxtaposed Labelled EC-CPM's $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ to execute only firing sequences which follow a certain "pattern." One such "pattern" is described by the <u>perfect shuffle</u> operation, denoted by $\downarrow$. Let Σ be a finite alphabet and suppose $w' = a'_1 a'_2 \ldots a'_m$ and $w'' = a''_1 a''_2 \ldots a''_m$ are two strings in Σ^+. Then,

$$w' \downarrow w'' = a'_1 a''_1 a'_2 a''_2 \ldots a'_m a''_m$$

By convention, $\lambda \downarrow \lambda = \lambda$. Note that the perfect shuffle operation can be applied only to equal-length strings. Note also that the $\downarrow$ operator is not commutative. If W' and W'' are languages over the alphabet Σ, then:

$$W' \downarrow W'' = \{w' \downarrow w'' \mid w' \in W', w'' \in W''\}$$

<u>Corollary 1</u>

Λ_o(EC-CPM) is closed under the perfect shuffle operation. Λ(EC-CPM) is not closed under the perfect shuffle operation.

Proof: Consider the following modifications to the construct of Theorem 3.2.1. Instead of introducing one new color c in the color set C, let us introduce two distinct new colors, denoted by c' and c", respectively. Next, for each transition t_k, let

$$H(a^1_{ck}) = \begin{cases} [c';1] & \text{if } t_k \in T' \\ [c";1] & \text{if } t_k \in T" \end{cases}$$

$$H(a^1_{kc}) = \begin{cases} [c";1] & \text{if } t_k \in T' \\ [c';1] & \text{if } t_k \in T" \end{cases}$$

The initial color marking UM^o becomes: $UM^o(p_c) = < (c',1) >$, $UM^o/P' \cup P" = UM^{o'} \nabla UM^{o"}$.

Thus, the control place is used to alternate the firing of a transition of $ECP_{\Sigma'}$ with the firing of a transition of $ECP_{\Sigma"}$. In the initial color marking UM^o, only transitions belonging to $ECP_{\Sigma'}$ can be enabled and thus the first transition to fire in any firing sequence must belong to $ECP_{\Sigma'}$. The final color marking UM^f of ECP_Σ is defined by $UM^f(p_c) = < (c',1) >$ and $UM^f/P' \cup P" = UM^{f'} \nabla UM^{f"}$. With this final color marking $\Lambda_o(ECP_\Sigma, UM^f) = \Lambda_o(ECP_{\Sigma'}, UM^{f'}) + \Lambda_o(ECP_{\Sigma"}, UM^{f"})$. If instead we want ECP_Σ to generate $\Lambda_o(ECP_{\Sigma"}, UM^{f"}) + \Lambda_o(ECP_{\Sigma'}, UM^{f'})$, all we have to do is to modify the initial and final color markings, respectively, as follows: $UM^o(p_c) = UM^f(p_c) = < (c",1) >$.

Let us now consider the language family Λ(EC-CPM). Note that according to Definition 3.1.1, the labelling function L of any Labelled EC-CPM must satisfy the following condition:

If γ is a nonempty firing sequence and $\gamma = \delta\sigma$, then $L(\gamma) = L(\delta)L(\sigma)$. In particular, if $\gamma = t_{k1}\ldots t_{km}$, then $L(\gamma) = L(t_{k1})\ldots L(t_{km})$, where $L(t_{kj}) \in \Sigma$, for $1 \le j \le m$ (Σ is the finite label alphabet associated with the respective Labelled EC-CPM).

Let $ECP_{\Sigma'}$ and $ECP_{\Sigma"}$ be two arbitrary Labelled EC-CPM's and suppose there exists a Labelled EC-CPM ECP_Σ such that $\Lambda(ECP_\Sigma) = \Lambda(ECP_{\Sigma'}) + \Lambda(ECP_{\Sigma"})$. Let $w' = a'_1\ldots a'_m$ and $w" = a"_1\ldots a"_m$ be two nonempty strings in $\Lambda(ECP_{\Sigma'})$ and $\Lambda(ECP_{\Sigma"})$, respectively. Hence, there exists a firing sequence $\gamma = t'_{k1}t"_{k2}\ldots t'_{km}t"_{km}$ in $S(UM^o)$ (here UM^o denotes the initial color marking of ECP_Σ) such that

$L(\gamma) = w' \diamond w'' = a'_1 a''_2 \ldots a'_m a''_m$. Consequently, the prefix $\gamma' = t'_{k1} t''_{k1} \ldots t'_{km}$ of γ, for example, is in $S(UM^o)$ as well. Therefore, $L(\gamma') = a'_1 a''_1 \ldots a'_m$ is contained in the language $\Lambda(ECP_\Sigma)$. This fact contradicts, however, the assumption that $\Lambda(ECP_\Sigma) = \Lambda(ECP_{\Sigma'}) \diamond \Lambda(ECP_{\Sigma''})$ since $a'_1 a''_1 \ldots a'_m \notin \Lambda(ECP_{\Sigma'}) \diamond \Lambda(ECP_{\Sigma''})$. □

"Patterns," more complicated than the perfect shuffle can certainly be imagined. A more general result will be proved in Theorem 3.3.3.

Let us now investigate the closure of the language families Λ(EC-CPM) and Λ_o(EC-CPM) under the union operation. If W' and W" are languages over some finite alphabet Σ then,

$$W' \cup W'' = \{w \mid w \in W' \text{ or } w \in W'' \text{ or both}\}$$

Theorem 3.2.2

Λ(EC-CPM) and Λ_o(EC-CPM) are closed under union.

Proof: Let us first consider the language family Λ(EC-CPM). Let $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ be two arbitrary Labelled EC-CPM's. We construct a Labelled EC-CPM ECP_Σ such that $\Lambda(ECP_\Sigma) = \Lambda(ECP_{\Sigma'}) \cup \Lambda(ECP_{\Sigma''})$. The construct employed is basically the same as that of Theorem 3.2.1, with the following modifications. In addition to the color c, we introduce two new distinct colors c' and c" in the color set C. Next, for each transition t_k, we set:

$$H(a^1_{ck}) = H(a^1_{kc}) = \begin{cases} [c';1] & \text{if } t_k \in T' \\ [c";1] & \text{if } t_k \in T" \end{cases}$$

Let now $E^{o'}$ denote the set of transitions of $ECP_{\Sigma'}$ enabled in the initial color marking $UM^{o'}$. For each transition $t'_r \in E^{o'}$, we add to ECP_Σ a transition t'_{rs} which has the same label, the same input and output connections as well as the same corresponding color threshold and color output functions as the transition t'_r, except for the input arc from the control place p_c, where $H(a^1_{c\,rs}) = [c;1]$.

The transitions t'_{rs} introduced above are sometimes referred to in the literature as "start transitions." A similar construct is performed for the transitions in $E^{o"}$, where $E^{o"}$ denotes the set of

transitions of $ECP_{\Sigma''}$ enabled in the initial color marking $UM^{o''}$.

Finally, the initial color marking UM^{o} of ECP_{Σ} is defined by $UM^{o}(p_c) = < (c,1) >$ and $UM^{o}/P' \cup P'' = UM^{o'} \nabla UM^{o''}$. In the initial color marking UM^{o}, due to the color bag of p_c, the only enabled transitions are the start transition corresponding to $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$. The firing of some start transition t_{js} will change the color bag of the control place to $< (c',1) >$ or $< (c'',1) >$ depending on whether t_{js} corresponds to $ECP_{\Sigma'}$ or $ECP_{\Sigma''}$, respectively. By our construct, after a start transition has fired, ECP_{Σ} can execute a firing sequence in either $ECP_{\Sigma'}$ or $ECP_{\Sigma''}$ (but not both) depending on the respective start transition. Moreover, no start transition can fire again along that firing sequence. We can conclude that the label sequences generated by ECP_{Σ} form the language $\Lambda(ECP_{\Sigma'}) \cup \Lambda(ECP_{\Sigma''})$.

The construct presented above is schematically presented in Figure 3.2.5.

For languages from the family Λ_o(EC-CPM) we can employ the same construct as above with certain modifications required by the fact that we have to ensure a unique final color marking for ECP_{Σ}.

If t'_{rs} is a start transition of ECP_{Σ} corresponding to $ECP_{\Sigma'}$, then (in addition to the places in P') we connect the places of $ECP_{\Sigma''}$, whose color bags are nonempty in $UM^{o''}$, to t'_{rs} as input places. We also connect the places of $ECP_{\Sigma''}$ which contain nonempty color bags in $UM^{f''}$ to t'_{rs} as output places. The input incidence function I of ECP_{Σ} and the color threshold functions of t'_{rs} are defined such that when fired, t'_{rs} collects from the places of $ECP_{\Sigma''}$ all the tokens present in the initial color marking $UM^{o''}$ (thus leading to an intermediate color marking in which all places of $ECP_{\Sigma''}$ are empty) . The output incidence function O of ECP_{Σ} and the output color functions of t'_{rs} are defined such that t'_{rs} deposits upon firing the final color marking $UM^{f''}$ in the places of $ECP_{\Sigma''}$. We have already shown that the firing of t'_{rs} triggers off a firing sequence in $ECP_{\Sigma'}$, which does not affect the color marking of $ECP_{\Sigma''}$. With the alteration described above, the firing of t'_{rs} also changes the color marking of $ECP_{\Sigma''}$ from $UM^{o''}$ to

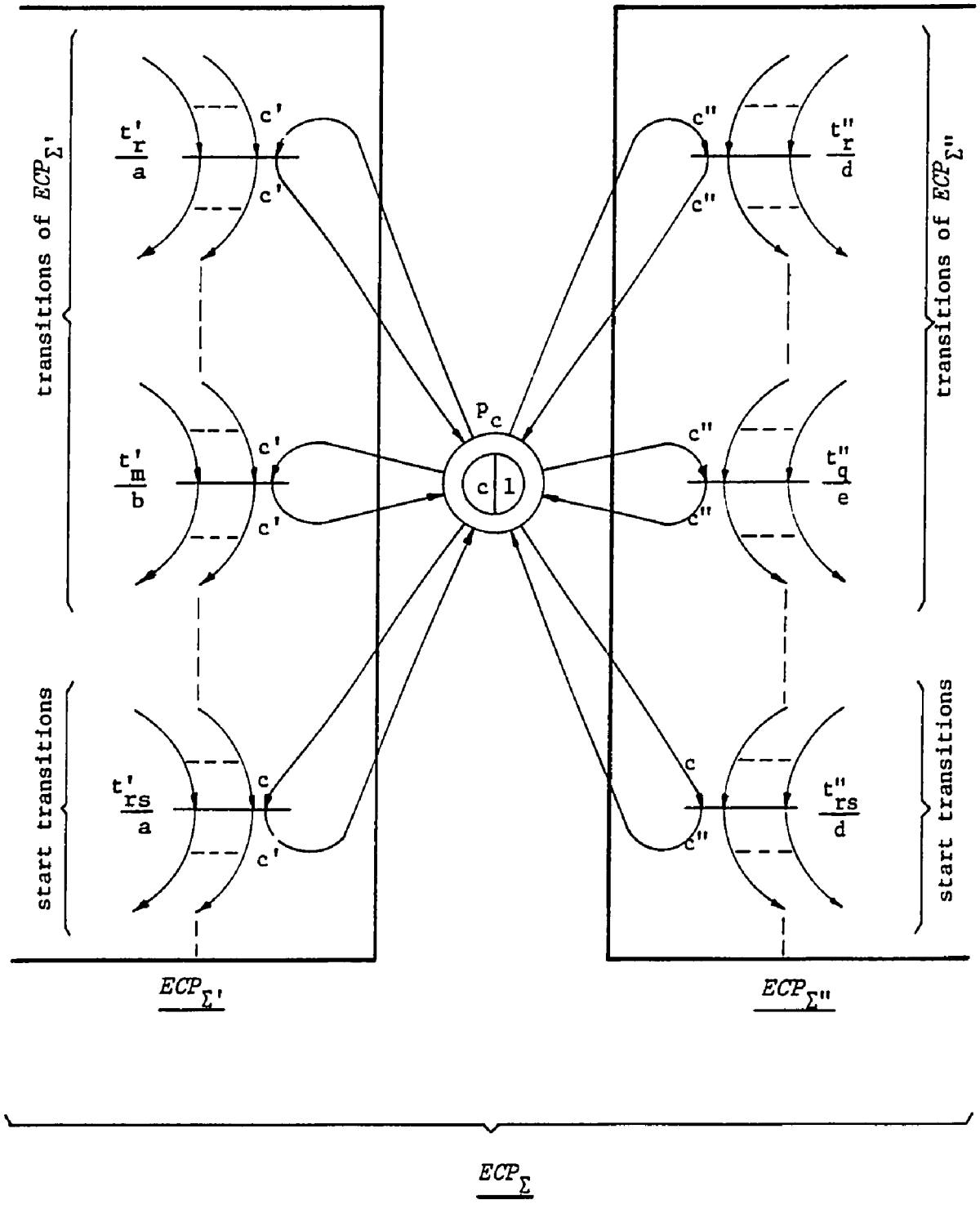


Figure 3.2.5
Construct for the Union Operation

$UM^{f''}$. If the firing sequence started off by t'_{rs} is a terminal firing sequence of $ECP_{\Sigma'}$, then the restriction to P' ∪ P" of the color marking obtained in ECP_{Σ} at the end of that firing sequence is $UM^{f'} \nabla UM^{f''}$.

A similar modification, with respect to $ECP_{\Sigma'}$, is performed on all start transitions of ECP_{Σ} corresponding to $ECP_{\Sigma''}$.

We still have to arrange for a unique final color marking of the control place p_c. For this purpose, for each transition t_k we add to ECP_{Σ} a copy of t_k, denoted by t_{kstop}. The transition t_{kstop} has exactly the same label, the same input and output connections as well as the same corresponding color threshold and color output functions as t_k, except for the control place p_c. The transition t_{kstop} has p_c only as an input place, i.e. $O(t_{kstop}, p_c) = 0$. Obviously, t_{kstop} is enabled exactly when the corresponding transition t_k is enabled. Moreover, the firing of t_{kstop} produces the same color marking change as the firing of t_k, with the exception of the place p_c. The firing of t_{kstop} removes the token from p_c and, therefore, leaves all the transitions of ECP_{Σ} disabled. The transitions t_{kstop} are sometimes referred to in the literature as "stop transitions."

Let us now define the final color marking UM^f of ECP_{Σ} by $UM^f(p_c) = \phi$ and $UM^f/P' \cup P'' = UM^{f'} \nabla UM^{f''}$. It should be easy to see that this final color marking can be reached only through the firing of a stop transition. Hence, for any nonempty firing sequence δt_{kstop}, $UM^o[\delta t_{kstop}> UM^f$ if and only if $UM^o[\delta t_k> UM^i$ and $UM^i/P' \cup P'' = UM^{f'} \nabla UM^{f''}$. Note that the prefix δ cannot contain any occurrence of a stop transition.

We conclude that $\Lambda_o(ECP_{\Sigma}, UM^f) = \Lambda_o(ECP_{\Sigma'}, UM^{f'}) \cup \Lambda_o(ECP_{\Sigma''}, UM^{f''})$. □

Let us now investigate the closure of the language families Λ(EC-CPM) and Λ_o(EC-CPM) under the operation of <u>concatenation</u>. If W' and W" are two languages over a finite alphabet Σ, then the concatenation of W' and W" is the set:

$$W'W'' = \{w'w'' \mid w' \in W', w'' \in W''\}.$$

Theorem 3.2.3

Λ(EC-CPM) and Λ_o(EC-CPM) are closed under concatenation.

Proof: Let $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ be two arbitrary Labelled EC-CPM's. We construct a Labelled EC-CPM ECP_{Σ} such that $\Lambda(ECP_{\Sigma})=\Lambda(ECP_{\Sigma'})\Lambda(ECP_{\Sigma''})$. If the final color markings $UM^{f'}$ and $UM^{f''}$ of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, respectively, are also given, then we define a final color marking UM^{f} of ECP_{Σ} such that $\Lambda_o(ECP_{\Sigma}, UM^{f}) = \Lambda_o(ECP_{\Sigma'}, UM^{f'})\Lambda_o(ECP_{\Sigma''}, UM^{f''})$. The construct employed here is a simplified version of the construct used in Theorem 3.2.2 in order to show the closure of the language family Λ(EC-CPM) under union.

Thus, suppose we have juxtaposed the Labelled EC-CPM's $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ and let the transitions of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ self-loop on a common control place p_c. We introduce two distinct new colors, c' and c", in the color set C formed as shown in the proof of Theorem 3.2.1 (instead of only one new color, c). For each transition t_k:

$$H(a^1_{ck}) = H(a^1_{kc}) = \begin{cases} [c';1] & \text{if } t_k \in T' \\ [c";1] & \text{if } t_k \in T" \end{cases}$$

Next, for each transition $t''_r \in E^{O''}$, let us add to ECP_{Σ} a start transition t''_{rs} which has the same label, the same input and output connections as well as the same corresponding color threshold and color output functions as t''_r, except for the control place p_c. We set $H(a^1_{c\,rs}) = [c';1]$. In the case of this construct, it is not necessary to introduce in ECP_{Σ} start transitions corresponding to $ECP_{\Sigma'}$.

The initial color marking UM^O of ECP_{Σ} is defined by $UM^O(p_c) = < (c',1) >$ and $UM^O/P' \cup P'' = UM^{O'} \nabla UM^{O''}$.

The transitions enabled in UM^O either belong to $ECP_{\Sigma'}$ or are start transitions corresponding to $ECP_{\Sigma''}$. Let us assume that a transition t'_k of $ECP_{\Sigma'}$ is selected to fire in UM^O. Then, t'_k starts off a firing sequence δ' in $ECP_{\Sigma'}$, which does not alter the color marking of the places of $ECP_{\Sigma''}$. At any time, δ' can be terminated by the firing of some start transition t''_{rs} of $ECP_{\Sigma''}$. Upon the firing of

t''_{rs}, the color bag of the control place p_c becomes $< (c'',1) >$ which leaves all the transitions of $ECP_{\Sigma'}$ and the start transitions corresponding to $ECP_{\Sigma''}$ disabled. The firing of t''_{rs} also triggers off a firing sequence δ'' in $ECP_{\Sigma''}$, which does not affect the color marking of $ECP_{\Sigma'}$. We can conclude that each firing sequence γ of ECP_{Σ} is of the form $\gamma = \delta'\delta''$ where δ' is a firing sequence of $ECP_{\Sigma'}$ and δ'' is a firing sequence of $ECP_{\Sigma''}$. If $UM^o[\delta'> UM'[\delta''> UM''$, then $UM'/P'' = UM^{o''}$ and $UM''/P' = UM'/P'$. Thus, $\Lambda(ECP_{\Sigma}) = \Lambda(ECP_{\Sigma'})\Lambda(ECP_{\Sigma''})$. If the final color markings $UM^{f'}$ and $UM^{f''}$ of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, respectively, are also given, then UM^f is defined by $UM^f(p_c) = < (c'',1) >$ and $UM^f/P' \dot{\cup} P'' = UM^{f'} \nabla UM^{f''}$. A firing sequence $\gamma = \delta'\delta''$ of ECP_{Σ} is a terminal firing sequence, i.e. $UM^o[\gamma> UM^f$, if and only if $UM^o[\delta'> UM'[\delta''> UM''$ where $UM'/P' = UM^{f'}$, $UM''/P'' = UM^{f''}$, i.e. δ' and δ'' are terminal firing sequences of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, respectively. Therefore, $\Lambda_o(ECP_{\Sigma}, UM^f) = \Lambda_o(ECP_{\Sigma'}, UM^{f'})\Lambda_o(ECP_{\Sigma''}, UM^{f''})$.

Figure 3.2.6 demonstrates the construct of this proof. □

Before showing the closure of the language families Λ(EC-CPM) and Λ_o(EC-CPM) under the operation of intersection, we introduce a lemma.

Let ECP_{Σ} be a Labelled EC-CPM and let UM^o and UM^f be its initial and final color marking, respectively. Let us assume that $|\Sigma| > 1$ and suppose d is a label in Σ. Let $ECP_{\Sigma'}$ denote the Labelled EC-CPM obtained by removing from ECP_{Σ} all transitions labelled d ($\Sigma' = \Sigma - \{d\}$). Then:

Lemma 3.2.1

$\Lambda(ECP_{\Sigma'}) \subseteq \Lambda(ECP_{\Sigma})$ and $\Lambda_o(ECP_{\Sigma'}, UM^f) \subseteq \Lambda_o(ECP_{\Sigma}, UM^f)$.

Proof: The proof is by induction. Obviously, $E^{o'} \subseteq E^o$. According to Definition 2.5.4, all enabled transitions of an EC-CPM have the same priority to fire. Consequently, if $\gamma \in S'(UM^o)$ is a firing sequence of length 1, then $\gamma \in S(UM^o)$. Let us now assume that if $\gamma \in S'(UM^o)$ is a firing sequence of length k, $k > 1$, then $\gamma \in S(UM^o)$. But if $UM^o[\gamma> UM^i$, then $E^{i'} \subseteq E^i$. By the same argument as aobve, if $t_r \in E^{i'}$ is selected to fire in $ECP_{\Sigma'}$, then t_r can be fired

in $ECP_{\Sigma'}$, as well. Hence, all firing sequences of length k + 1 from $S'(UM^o)$ are in $S(UM^o)$. We can conclude that $S'(UM^o) \subseteq S(UM^o)$ and, in particular, $T'(UM^o, UM^f) \subseteq T(UM^o, UM^f)$.

It is interesting to mention the following fact. It is undecidable whether the removal of a transition changes the language generated by a Generalized Petri Net (Theorem 8.4 [HACK-75]). On the other hand, it should be obvious that $\Lambda(GPN) \subseteq \Lambda(EC\text{-}CPM)$ and $\Lambda_o(GPN) \subseteq \Lambda_o(EC\text{-}CPM)$. Hence, it is undecidable whether $\Lambda(ECP_{\Sigma'})$ and $\Lambda_o(ECP_{\Sigma'}, UM^f)$ are in fact proper subsets of $\Lambda(ECP_{\Sigma})$ and $\Lambda_o(ECP_{\Sigma}, UM^f)$, respectively.

□

Let us now consider the operation of intersection. If W' and W'' are languages over some finite alphabet Σ, then

$$W' \cap W'' = \{w \mid w \in W' \text{ and } w \in W''\}$$

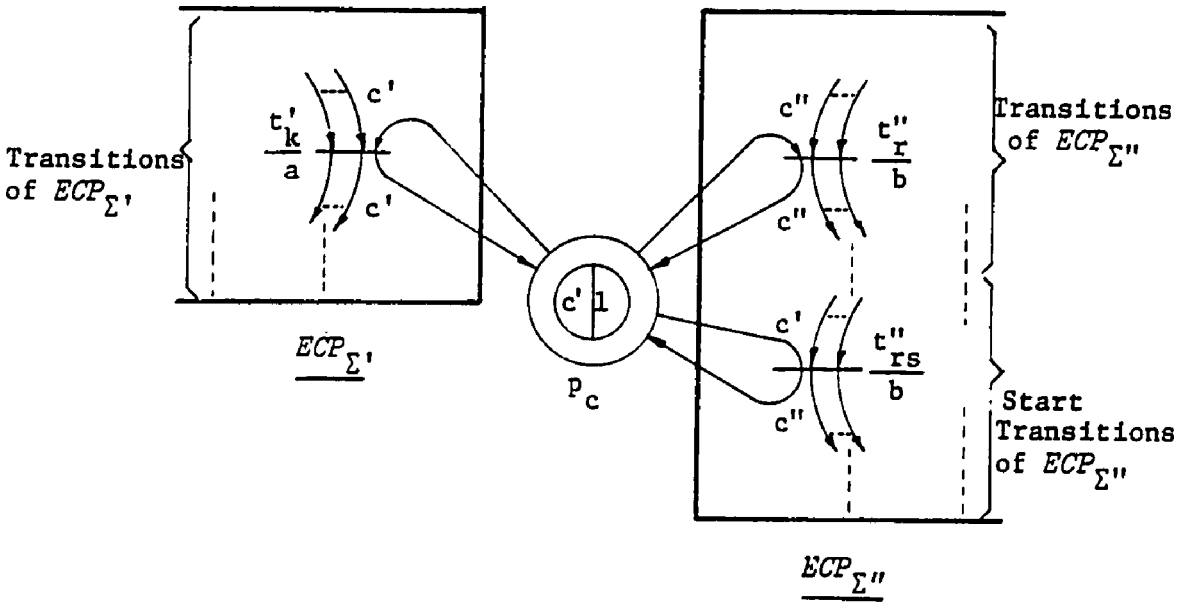


Figure 3.2.6

Construct for the Operation of Concatenation

Theorem 3.2.4

Λ(EC-CPM) and Λ_o(EC-CPM) are closed under intersection.

Proof: Let us first consider the language family Λ(EC-CPM). Let $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ be two Labelled EC-CPM's and suppose $w = a_1a_2\ldots a_m$ is a string in $\Lambda(ECP_{\Sigma'}) \cap \Lambda(ECP_{\Sigma''})$. Consequently, there exist the firing sequences $\gamma' = t'_{k1}t'_{k2}\ldots t'_{km}$ and $\gamma'' = t''_{k1}t''_{k2}\ldots t''_{km}$ in $S(UM^{o'})$ and $S(UM^{o''})$, respectively, such that $L'(\gamma') = L''(\gamma'') = w$. Obviously, $|\gamma'| = |\gamma''|$ and $L'(t'_{kj}) = L''(t''_{kj}) = a_j$, for $1 \leq j \leq m$.

Our construct is based on the above remarks. We define a Labelled EC-CPM ECP_Σ such that $\Lambda(ECP_\Sigma) = \Lambda(ECP_{\Sigma'}) \cap \Lambda(ECP_{\Sigma''})$. If $C' = (\{c'_1,\ldots,c'_s, c'_m, c'_M\}, \leq')$ and $C'' = (\{c''_1,\ldots,c''_p, c''_m, c''_M\}, \leq'')$ then $C = (X, \leq)$ where $X = \{c'_1,\ldots,c'_s\} \cup \{c''_1,\ldots,c''_p\} \cup \{c_m, c_M\}$ and the partial ordering $\leq$ satisfies the conditions (I) and (II) given in Section 2.5. U(C) is the color set associated with ECP_Σ.

The Labelled EC-CPM ECP_Σ preserves the places of both $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, i.e. $P = P' \cup P''$ (again, we assume that $P' \cap P'' = \phi$). In addition, we combine the transitions of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$ in such a way that ECP_Σ is forced to perform only firing sequences whose corresponding label sequences are common to $\Lambda(ECP_{\Sigma'})$ and $\Lambda(ECP_{\Sigma''})$. Let t'_k and t''_r be two transitions of $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, respectively, such that $L'(t'_k) = L''(t''_r)$. We introduce for this pair of transitions one transition, let us denote it by t_{kr}, in ECP_Σ. The transition t_{kr} has the input and output connections of both t'_k and t''_r as well as the same corresponding color threshold and color output functions. Moreover, t_{kr} is assigned the common label of both t'_k and t''_r. Figure 3.2.8 exhibits the transition t_{kr} obtained from the sample transitions t'_k and t''_r of Figure 3.2.7. The transition t_{kr} is enabled exactly when both t'_k and t''_r are enabled and its firing has the combined effect of the firings of t'_k and t''_r (or, the effect produced by the simultaneous firing of t'_k and t''_r). This construct is performed for each pair of identically labelled transitions from $ECP_{\Sigma'}$ and $ECP_{\Sigma''}$, respectively. Based on Lemma 3.2.1, transitions which carry labels not in $\Sigma' \cap \Sigma''$ are eliminated since they cannot possibly participate

in the generation of label sequences in $\Lambda(ECP_{\Sigma'}) \cap \Lambda(ECP_{\Sigma''})$.

Finally, the initial color marking UM^o of ECP_Σ is defined by $UM^o = UM^{o'} \nabla UM^{o''}$. The complete construct is presented in Figure 3.2.10 for the sample Labelled EC-CPM's of Figure 3.2.9 and 3.2.1

We can see that $\Lambda(ECP_{\Sigma'}) \cap \Lambda(ECP_{\Sigma''}) \subseteq \Lambda(ECP_\Sigma)$. We now show that it is not a proper subset. It is easy to see that $R(UM^o) \subseteq R'(UM^{o'}) \bar{\nabla} R''(UM^{o''})$. Let $UM^i \in E(UM^o)$ and let $t_{kr} \in E^i$. Then, by our construct, there exist the transitions t'_k in $ECP_{\Sigma'}$ and t''_r in $ECP_{\Sigma''}$ such that $L(t_{kr}) = L'(t'_k) = L''(t''_r)$, t'_k is enabled in the color marking UM^i/P' of $ECP_{\Sigma'}$, and t''_r is enabled in the color marking UM^i/P'' of $ECP_{\Sigma''}$(recall that $E(UM^o) \subseteq R(UM^o)$ by Definition 2.3.12). Therefore, the composite net does not generate any new label sequences. Hence, $\Lambda(ECP_\Sigma) = \Lambda(ECP_{\Sigma'}) \cap \Lambda(ECP_{\Sigma''})$.

The above construct applies to the language family Λ_o(EC-CPM), as well. If $UM^f = UM^{f'} \nabla UM^{f''}$ then
$\Lambda_o(ECP_\Sigma, UM^f) = \Lambda_o(ECP_{\Sigma'}, UM^{f'}) \cap \Lambda_o(ECP_{\Sigma''}, UM^{f''})$.

□

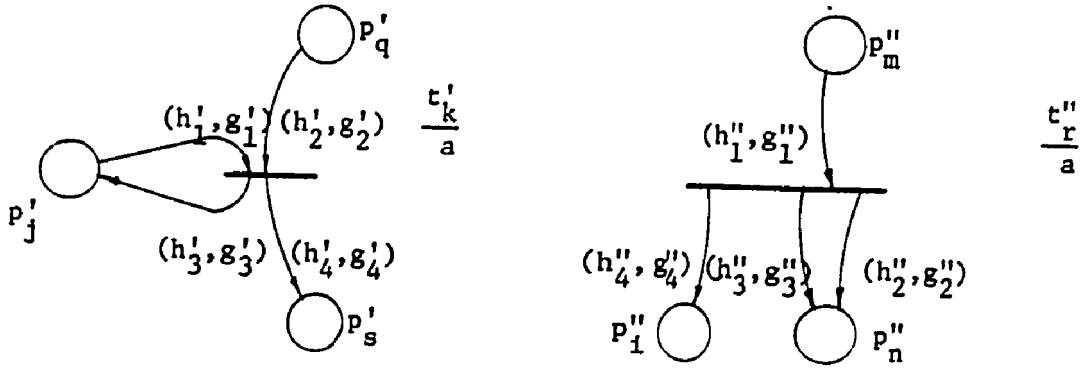


Figure 3.2.7
Sample Transitions

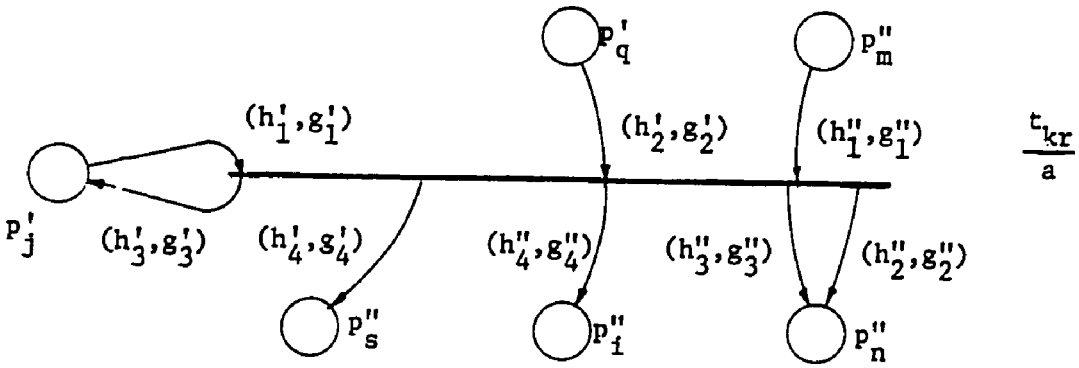


Figure 3.2.8

The Composition of the Transitions t'_k and t''_r

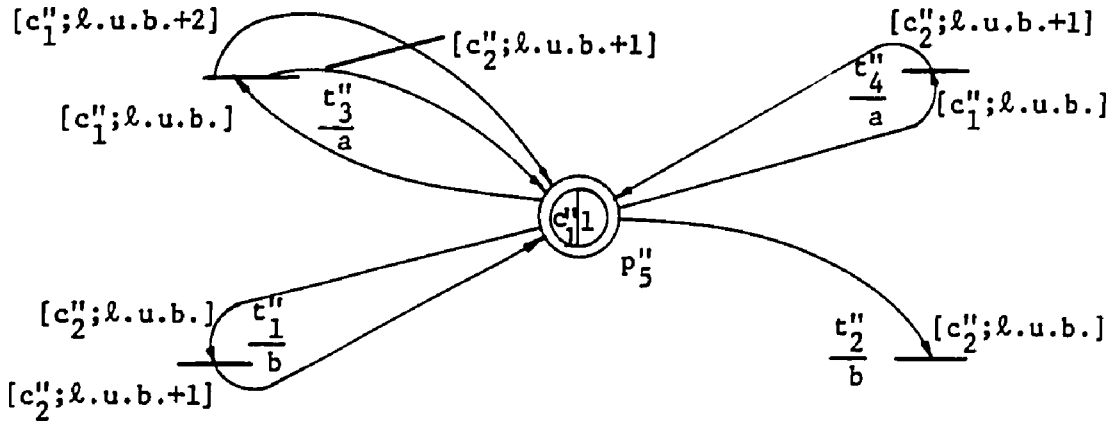


Figure 3.2.9

Sample Labelled EC-CPM

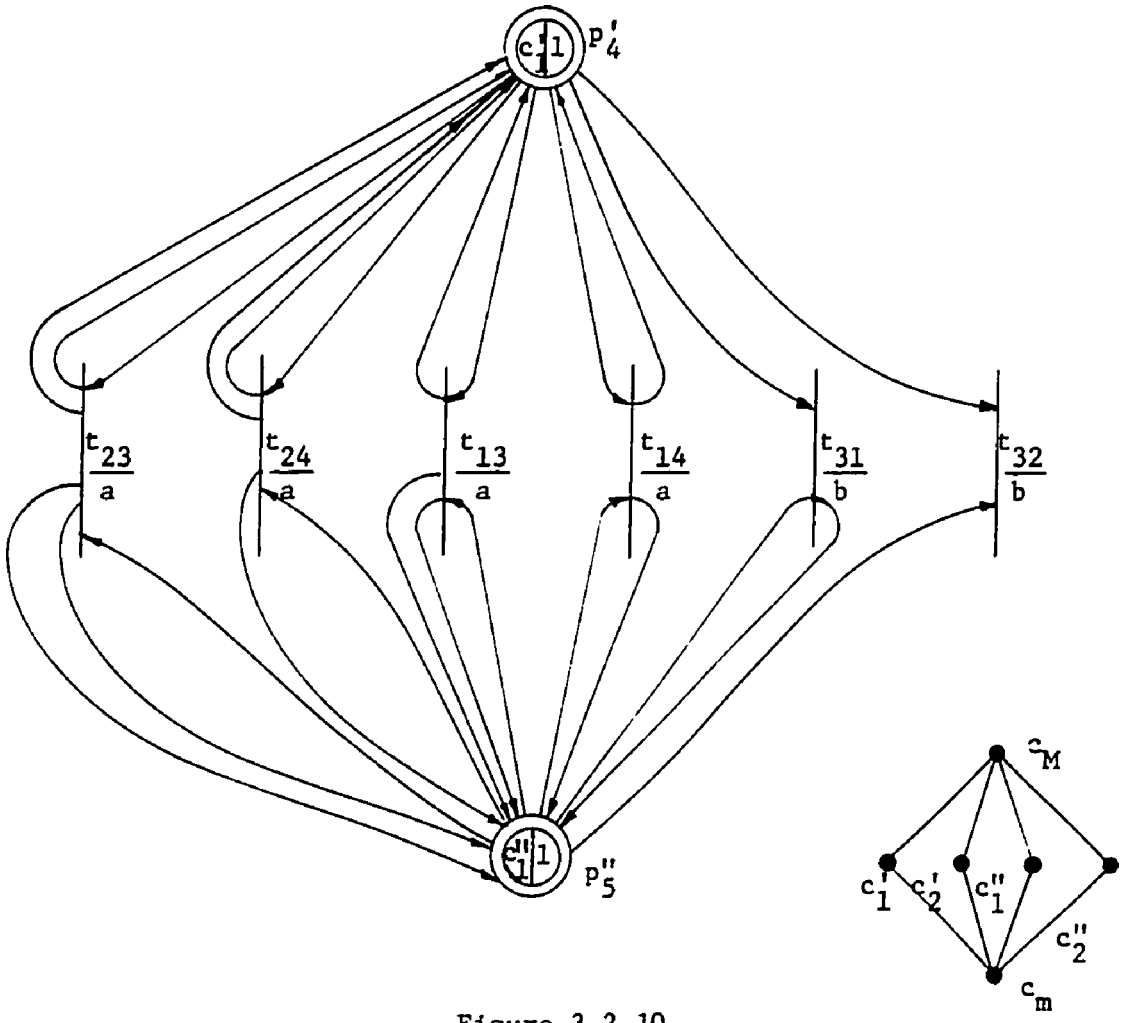


Figure 3.2.10

Construct for the Intersection Operation

Let us now proceed to investigate the closure of the language family Λ_o(EC-CPM) under indefinite concatenation. First, we present a few introductory remarks.

Let ECP_Σ be a Labelled EC-CPM in some color marking UM^i. Suppose t_k is a transition of ECP_Σ enabled in UM^i. According to the definitions given in Section 2.5 (especially because of the fact that the relation R_E is used in the selection of enabling colors), if $UM^i[t_k> UM^j$ then the color marking UM^j is uniquely determined by UM^i and t_k.

Suppose now that we are given a fixed color marking UM^j and suppose there exists a color marking UM^i such that $UM^i[t_k> UM^j$.

Let UM^{i-} denote the intermediate color marking obtained after t_k has removed its enabling tokens in UM^i but before t_k has deposited its output tokens (see Definition 2.3.2).

Lemma 3.2.2

The intermediate color marking UM^{i-} is uniquely determined by UM^j and t_k.

Proof: We give a procedure for obtaining UM^{i-}. Let p_r be a place of ECP_Σ. If $O(t_k, p_r) = 0$, then $U_r^{i-} = U_r^j$. Let us assume that $O(t_k, p_r) = m$, $m > 0$. Consequently, there are m output arcs, $a_{kr}^1, \ldots, a_{kr}^m$, from t_k to p_r. Obviously, we must have

$$\sum_{u_p \in \mathscr{D}(U_r^j)} \#(u_p, U_r^j) = n \geq m$$

If $m = n$, then $U_r^{i-} = \phi$. Let us then suppose that $n > m$, i.e. $U_r^{i-} \neq \phi$. We consider the following possible cases:

1. $H(a_{kr}^s) = [c_s; z_s]$ for all s, $1 \leq s \leq m$, where $z_s \in Z^+$.

Consequently,

$$U_r^j = U_r^{i-} \cup \langle (c_s, z_s) \mid 1 \leq s \leq m \rangle$$

Thus, the color bag U_r^{i-} is obtained by removing the output tokens of t_k from U_r^j:

$$U_r^{i-} = U_r^j - \langle (c_s, z_s) \mid 1 \leq s \leq m \rangle$$

2. $H(a_{kr}^s) = [c_s; \text{g.}\ell\text{.b.} + m_s]$ for all s, $1 \leq s \leq m$, where $m_s \in Z^o$. Then

$$U_r^j = U_r^{i-} \cup \langle (c_s, \text{g.}\ell\text{.b.}(\mathscr{D}^2(U_r^{i-})) + m_s) \mid 1 \leq s \leq m \rangle$$

Since $m_s \geq 0$, for $1 \leq s \leq m$, we have

$$\text{g.}\ell\text{.b.}(\mathscr{D}^2(U_r^{i-})) = \text{g.}\ell\text{.b.}(\mathscr{D}^2(U_r^j)) = K'$$

Hence,

$$U_r^{i-} = U_r^j - \langle (c_s, K' + m_s) \mid 1 \leq s \leq m \rangle$$

3. $H(a_{kr}^s) = [c_s; \ell\text{.u.b.} + m_s]$ for all s, $1 \leq s \leq m$, where $m_s \in Z^o$. Then:

$$U_r^j = U_r^{i-} \cup \langle (c_s, \ell\text{.u.b.}(\mathscr{D}^2(U_r^{i-})) + m_s) \mid 1 \leq s \leq m \rangle$$

Let $K'' = \max\{m_s \mid 1 \leq s \leq m\}$. Since $m_s \geq 0$, we must have:

$$\ell.u.b.(\mathscr{D}^2(U_r^j)) = K''' = \ell.u.b.(\mathscr{D}^2(U_r^{i-})) + K''$$

Hence,

$$\ell.u.b.(\mathscr{D}^2(U_r^{i-})) = K''' - K'' = M$$

and $U_r^{i-} = U_r^j - \langle (c_s, M + m_s) \mid 1 \leq s \leq m \rangle$

4. The color output functions $H(a_{kr}^s)$, $1 \leq s \leq m$, are a combination of the three cases examined above. Note that the colors of the output tokens in the case 1 above do not depend on the color bag U_r^{i-}. Thus, these tokens can immediately be identified in the color bag U_r^j. Let $U_r^{j'}$ denote the color bag obtained by removing from U_r^j the colors of all tokens corresponding to arcs of type 1. The colors of the output tokens corresponding to arcs of type 2 are determined as shown above using $U_r^{j'}$ instead of U_r^j, i.e. taking $K' = g.\ell.b.(\mathscr{D}^2(U_r^{j'}))$. Let $U_r^{j''}$ denote the color bag obtained by removing from $U_r^{j'}$ the colors of all tokens corresponding to arcs of type 2. Then, the colors of the tokens corresponding to arcs of type 3 are determined as shown above using $U_r^{j''}$ instead of U_r^j, i.e. taking $M = \ell.u.b.(\mathscr{D}^2(U_r^{j''})) - K''$. A little consideration will show that the color bag obtained by removing from $U_r^{j''}$ the colors of all tokens corresponding to arcs of type 3 is U_r^{i-}.

□

We note that given an arbitrary color marking UM^j and a transition t_k of ECP_Σ there may not exist any intermediate color marking UM^{i-} which generates UM^j after t_k has deposited its output tokens. Not only must the procedure of Lemma 3.2.2 be applicable to the transition t_k and the color marking UM^j, but the following two conditions must be satisfied by the resulting color bag U_r^{i-} and the intermediate color bags $U_r^{j'}$ and $U_r^{j''}$, for all places $p_r \in P$:

1. $\ell.u.b.(\mathscr{D}^2(U_r^{j''})) - K'' \geq g.\ell.b.(\mathscr{D}^2(U_r^{j'}))$.
2. $g.\ell.b.(\mathscr{D}^2(U_r^{i-})) = g.\ell.b.(\mathscr{D}^2(U_r^{j'}))$ and
   $\ell.u.b.(\mathscr{D}^2(U_r^{i-})) = \ell.u.b.(\mathscr{D}^2(U_r^{j''})) - K''$.

Note also that even though UM^{i-} is unique, there may exist none or one or more than one color markings UM^i which generate the same intermediate color marking UM^{i-} after t_k has removed its enabling tokens.

If one or more such color markings UM^i exist, one possible color marking UM^i can be constructed from UM^{i-} using the following method. Let p_r be a place of ECP_Σ. If $I(p_r, t_k) = 0$, then $U_r^i = U_r^{i-}$. Suppose now that $I(p_r, t_k) = m$, $m > 0$, and that $H(a_{rk}^s) = [c_s; z_s]$, $z_s \in Z^+$, for $1 \leq s \leq v$, $H(a_{rk}^s) = [c_s; \text{g.}\ell\text{.b.} + m_s]$, $m_s \in Z^o$, for $v + 1 \leq s \leq v + w$ and $H(a_{rk}^s) = [c_s; \ell\text{.u.b.} - m_s]$, $m_s \in Z^o$, for $v + w + 1 \leq s \leq m$. Let:

$$U_r^{i'} = U_r^{i-} \cup \langle (c_s, z_s) \mid 1 \leq s \leq v \rangle$$

and let $Lo = \text{g.}\ell\text{.b.}(\mathscr{D}^2(U_r^{i'}))$,

$Hi = \ell\text{.u.b.}(\mathscr{D}^2(U_r^{i'}))$

Let us also introduce the notations

$$K' = \max\{m_s \mid v + 1 \leq s \leq v + w\}$$

and $K'' = \max\{m_s \mid v + w + 1 \leq s \leq m\}$

Then, if:

i. $K' \leq Hi - Lo$ and $K'' \leq Hi - Lo$, then

$$U_r^i = U_r^{i'} \cup \langle (c_s, Lo + m_s) \mid v + 1 \leq s \leq v + w \rangle \cup \langle (c_s, Hi - m_s) \mid v + w + 1 \leq s \leq m \rangle$$

ii. $K'' \leq Hi - Lo$ but $K' > Hi - Lo$, then

$$U_r^i = U_r^{i'} \cup \langle (c_s, Lo + m_s) \mid v + 1 \leq s \leq v + w \rangle \cup \langle (c_s, Lo + K' - m_s) \mid v + w + 1 \leq s \leq m \rangle$$

iii. $K' \leq Hi - Lo$ but $K'' > Hi - Lo$. Note that if $K'' \geq Hi$, then there does not exist any color marking UM^i which produces the given intermediate color marking UM^{i-} after t_k has removed its enabling tokens. If $Hi - Lo < K'' < Hi$,

$$U_r^i = U_r^{i'} \cup \langle (c_s, Hi - K'' + m_s) \mid v + 1 \leq s \leq v + w \rangle \cup \langle (c_s, Hi - m_s) \mid v + w + 1 \leq s \leq m \rangle$$

iv. $K' > Hi - Lo$ and $K'' > Hi - Lo$. If $K' \geq K''$, then

$$U_r^i = U_r^{i'} \cup \langle (c_s, Lo + m_s) \mid v + 1 \leq s \leq v + w \rangle \cup \langle (c_s, Lo + K' - m_s) \mid v + w + 1 \leq s \leq m \rangle$$

If $K'' > K'$ and $K'' \geq Hi$ then again there does not exist any color marking UM^i which generates the given intermediate color marking UM^{i-} after t_k has removed its enabling tokens. If $Hi > K'' > K'$, then

$$U_r^{i^k} = U^{i'} \cup \langle (c_s, Hi - K'' + m_s) \mid v + 1 \leq s \leq v + w \rangle \cup \langle (c_s, Hi - m_s) \mid v + w + 1 \leq s \leq m \rangle$$

This contruct is applied to all places p_r of ECP_Σ. The method can certainly be particularized for the case when one or two of the categories of color threshold functions considered above is missing. It should be clear that t_k is enabled in UM^i and that the given intermediate color marking UM^{i-} is obtained from UM^i after t_k removes its enabling tokens. Hence, $UM^i[t_k > UM^j$.

We conclude this discussion with the remark that even if a color marking UM^i can be formed using the above methods that does not mean that UM^i is a reachable color marking of ECP_Σ.

The results presented above are important in the following perspective. Suppose we are given a fixed color marking of ECP_Σ, say the final color marking UM^f. Let t_k be a transition of ECP_Σ for which we have determined that there exists at least one color marking UM^i such that $UM^i[t_k > UM^f$ and, hence, there exists an intermediate color marking UM^{i-} which augmented with the output tokens of t_k produces the given color marking UM^f. Thus, t_k is one of the transitions which may fire last in a terminal firing sequence of ECP_Σ. Suppose now that we want to construct a Labelled EC-CPM ECP'_Σ, whose final color marking is $UM^{f'}$ where for each place p_r of ECP'_Σ, $UM^{f'}(p_r)=\phi$, and which generates the same language in Λ_o(EC-CPM) as ECP_Σ ($UM^{f'}$ will also be referred to as the "zero marking"). This can be accomplished by means of the following construct.

First, we introduce a new color, c, and a new place p_c on which all transitions of ECP_Σ self-loop. For each transition t_q, let $H(a^1_{cq}) = H(a^1_{qc}) = [c;1]$. The introduction of the control place p_c does not modify the Λ_o-language of ECP_Σ.

Next, we add to ECP_Σ a copy of t_k, where t_k is a possible "last" transition, denoted by t_{kstop}. The transition t_{kstop} will have the same label, the same input connections and the same corresponding color threshold functions as t_k. On the other hand, for each place p_r of ECP_Σ, including the control place p_c, $O(t_k, p_r) = 0$. In addition we connect as input places to t_{kstop} all places p_r of ECP_Σ for which $UM^{i-}(p_r) \neq \phi$. The input incidence function of ECP_Σ and the corresponding color threshold functions are defined such that t_{kstop} removes from each such place p_r the color bag U^{i-}_r upon firing. Thus,

if (c, q) is a color in U_r^{i-} then there exists an input arc $a_r{}^s{}_{kstop}$ such that $H(a_r{}^s{}_{kstop}) = [c; q]$.

In any color marking UM^i in which t_{kstop} is enabled the bag of enabling colors of t_{kstop} from the input place p_r is

$$\theta^i_{r\ kstop} = \theta^i_{rk} \cup U^{i-}_r$$

where θ^i_{rk} is the bag of enabling colors of t_k from the input place p_r. Hence, t_{kstop} is enabled in UM^i only if t_k is enabled in UM^i. Moreover, $UM^i[t_{kstop} > UM^{f'}$ if and only if $UM^i[t_k > UM^f$.

Suppose now that this construct is performed for all transitions t_k of ECP_Σ which can lead to the final color marking UM^f. Let ECP'_Σ be the resulting Labelled EC-CPM. Then, $\Lambda_o(ECP_\Sigma, UM^f) = \Lambda_o(ECP'_\Sigma, UM^{f'})$. Note that in the color marking $UM^{f'}$, all transitions, including the stop transitions, are disabled. Moreover, only the firing of a stop transition can lead to the color marking $UM^{f'}$ since only stop transitions remove but do not restore the token in the control place p_c.

The type of stop transitions described here is employed in the construct to be presented next in connection with the operation of indefinite concatenation (+).

Let W be a language over some finite alphabet Σ. The indefinite concatenation of W is defined to be the set:

$$W^+ = \bigcup_{i=1}^{\infty} W^i$$

where

$$W^i = \{w_1 w_2 \ldots w_i \mid w_j \in W,\ 1 \le j \le i\}$$

Theorem 3.2.5

The language family Λ_o(EC-CPM) is closed under indefinite concatenation.

Proof: Let $ECP_\Sigma = (ECP, \Sigma, L)$ be a Labelled EC-CPM, where $ECP = (CN, UM^o)$, $CN = (N, U(C), H)$ and $N = (T, P, I, O)$. Let UM^f be the final color marking of ECP_Σ. Then

$$\Lambda_o(ECP_\Sigma, UM^f)^i = \{L(\gamma_1)L(\gamma_2)\ldots L(\gamma_i) \mid \gamma_j \in T(UM^o, UM^f),\ 1 \le j \le i\}$$
$$= L(T(UM^o, UM^f)^i)$$

Hence, $\Lambda_o(ECP_\Sigma, UM^f)^+ = L(T(UM^o, UM^f)^+)$

Our construct implements this definition. Let us first present a few language preserving transformations on ECP_Σ. Suppose we add to ECP_Σ a copy of each place $p_j \in P$, denoted by p_{jj}. We shall refer to p_{jj} as to the "counter place corresponding to p_j." Let P' denote the set of all such counter places; obviously, $|P| = |P'|$. Next, let us introduce in the color set C three new colors, denoted by c, c' and c", respectively. The initial color marking $UM^{o'}$ of the resulting net is defined by:

$UM^{o'}/P = UM^{o}$

$UM^{o'}(p_{jj}) = < (c,1),(c",2)^{M^{o}(p_j)} >$ for all places $p_{jj} \in P'$.

For each transition t_k and all places $p_{jj} \in P'$, we set:

$I(p_{jj}, t_k) = I(p_j, t_k)$

$H(a_{jj}{}^{s}{}_{k}) = [c";2]$ for all s, $1 \le s \le I(p_{jj}, t_k)$

$O(t_k, p_{jj}) = O(t_k, p_j)$

$H(a_{k}{}^{s}{}_{jj}) = [c";2]$ for all s, $1 \le s \le O(t_k, p_{jj})$

The counter place p_{jj} is meant to keep a count of the number of tokens present in the corresponding original place p_j. The input and output connections of each transition t_k to the places in P' are defined such that the effect of the firing of t_k on the color marking of each counter place p_{jj} reflects the marking change produced by the firing of t_k on the corresponding place $p_j \in P$.

Suppose t_k is a transition enabled in the initial color marking $UM^{o'}$ and $UM^{o'}[t_k> UM^{1'}$. Then, for all places $p_{jj} \in P'$, $UM^{1'}(p_{jj}) = < (c,1), (c",2)^{n_j} >$ where

$$n_j = M^{o'}(p_{jj}) - 1 - I(p_{jj}, t_k) + O(t_k, p_{jj})$$
$$= M^{o'}(p_j) - I(p_j, t_k) + O(t_k, p_j) = M^{1'}(p_j)$$

Suppose now that for each firing sequence γ of length m, $m > 1$, where $UM^{o'}[\gamma> UM^{i'}$,

$UM^{i'}(p_{jj}) = < (c,1),(c",2)^{M^{i'}(p_j)} >$ for all places $p_{jj} \in P'$.

Let t_k be a transition enabled in the color marking $UM^{i'}$ and suppose $UM^{i'}[t_k> UM^{i+1'}$. Then, by the same argument as above,

$UM^{i+1'}(p_{jj}) = < (c,1),(c",2)^{M^{i+1'}(p_j)} >$ for all places $p_{jj} \in P'$.

We can conclude that if $UM^{r'}$ is a reachable color marking of the modified net, then

$UM^{r'}(p_{jj}) = \langle (c,1),(c'',2)^{M^{r'}(p_j)} \rangle$ for all places $p_{jj} \in P'$.

Obviously, $UM^{r'}(p_{jj}) = \langle (c,1) \rangle$ if and only if $UM^{r'}(p_j) = \phi$.

Let now t_k be a transition of the original Labelled EC-CPM ECP_Σ, enabled in the initial color marking UM^o. Then, for each place $p_{jj} \in P'$

$$\#((c'',2), U^{o'}_{jj}) = M^o(p_j) \geq I(p_j, t_k) = I(p_{jj}, t_k)$$

Since $UM^{o'}/P = UM^o$ and since the transitions of the modified net are connected to the places in P exactly the same way as the transitions of the original net, t_k is also enabled in $UM^{o'}$. Moreover, if $UM^o[t_k\rangle UM^1$ in ECP_Σ and $UM^{o'}[t_k\rangle UM^{1'}$ in the modified net, then $UM^{1'}/P = UM^1$.

Suppose now that for each firing sequence γ of length m, $m > 1$, if γ can be executed in ECP_Σ, then γ can also be executed in the modified net. Suppose also that if $UM^o[\gamma\rangle UM^i$ in ECP_Σ, then in the modified net $UM^{o'}[\gamma\rangle UM^{i'}$, where $UM^{i'}/P = UM^i$. Let t_k be a transition of ECP_Σ enabled in the color marking UM^i. According to our previous result, for all places $p_{jj} \in P'$

$$\#((c'',2), U^{i'}_{jj}) = M^{i'}(p_j) = M^i(p_j) \geq I(p_j, t_k) = I(p_{jj}, t_k)$$

By the same argument as above, t_k is enabled in $UM^{i'}$ and if $UM^i[t_k\rangle UM^{i+1}$ in ECP_Σ and $UM^{i'}[t_k\rangle UM^{i+1'}$ in the modified net, then $UM^{i+1'}/P = UM^{i+1}$.

We can conclude that if γ is a firing sequence of ECP_Σ, then γ can be executed in the modified net as well. Moreover, if $UM^o[\gamma\rangle UM^j$ in ECP_Σ and $UM^{o'}[\gamma\rangle UM^{j'}$ in the modified net, then $UM^{j'}/P = UM^j$.

Using a similar induction argument, one can show that if γ is a firing sequence executed by the modified net and $UM^{o'}[\gamma\rangle UM^{j'}$, then γ can be performed by the original net ECP_Σ as well and $UM^o[\gamma\rangle UM^j$ where UM^j is the restriction of the color marking $UM^{j'}$ to P.

Let us define the final color marking $UM^{f'}$ of the modified net by:

$$UM^{f'}/P = UM^{f}$$

and for all places $p_{jj} \in P'$, $UM^{f'}(p_{jj}) = < (c,1),(c'',2)^{M^{f}(p_j)} >$. From the above discussion, we conclude that $UM^{o}[\gamma> UM^{f}$ in ECP_{Σ} if and only if $UM^{o'}[\gamma> UM^{f'}$ in the modified net.

Suppose now that t_k is a transition of the modified net for which there exists a color marking $UM^{i'}$ such that $UM^{i'}[t_k> UM^{f'}$ (note that t_k is a "last" transition of the modified net if and only if t_k is a "last" transition of the original Labelled EC-CPM ECP_{Σ}). For each such transition t_k, we add to the modified net a distinct stop transition t_{kstop} of the type discussed in connection with Lemma 3.2.2. For the purpose of this proof, we alter, however, the transitions t_{kstop} as follows. Each stop transition removes from the places in P' only the tokens of color (c",2) but not the (single) token of color (c,1). Let then $UM^{f''}$ be the color marking defined by:

$UM^{f''}(p_j) = \phi$ for all places $p_j \in P$

$UM^{f''}(p_{jj}) = < (c,1) >$ for all places $p_{jj} \in P'$.

Consequently, if δt_k is a firing sequence such that $UM^{o'}[\delta t_k> UM^{f'}$, then $UM^{o'}[\delta t_{kstop}> UM^{f''}$. If, however, $UM^{j'}$ is some color marking such that $UM^{j'}[t_k> UM^{r'}$, $UM^{r'} \neq UM^{f'}$, and t_{kstop} can also fire in $UM^{j'}$, then the firing of t_{kstop} in $UM^{j'}$ does not lead to the color marking $UM^{f''}$. In particular, there exists a place $p_{jj} \in P'$ which will contain tokens of color (c",2) after the firing of t_{kstop}. Thus, the color marking $UM^{f''}$ can be obtained only if t_{kstop} fires in a "proper" pre-final color marking.

Let us now introduce a new place p_c on which all transitions, including the stop transitions, self-loop. For any transition t_q, let:

$$H(a^1_{cq}) = [c';1]$$

$$H(a^1_{qc}) = \begin{cases} [c';1] & \text{if } t_q \text{ is not a stop transition} \\ [c;1] & \text{if } t_q \text{ is a stop transition} \end{cases}$$

Initially, p_c contains a single token of color (c',1). Let also $UM^{f'''}$ be the color marking defined by:

$$UM^{f'''}/P \cup P' = UM^{f''}$$
$$UM^{f'''}(p_c) = \langle (c,1) \rangle$$

Note that the color marking $UM^{f'''}$ can be reached only after the firing of a stop transition. Moreover, in $UM^{f'''}$ all transitions, including the stop transitions, are disabled (due to the color bag of the control place).

Let us now consider the set of transitions of the original Labelled EC-CPM ECP_{Σ} enabled in the initial color marking UM^{o}, i.e. the set E^{o}. For each transition $t_r \in E^{o}$, we add to the modified net a distinct start transition t_{rstart}, which carries the same label as t_r. We set:

$I(p_j, t_{rstart}) = 0$ for all places $p_j \in P$

$I(p_{jj}, t_{rstart}) = 1$ for all places $p_{jj} \in P'$ and

$H(a_{jj}{}^{1}{}_{rstart}) = [c; \ell.u.b.]$

$I(p_c, t_{rstart}) = 1$ and $H(a_{c}{}^{1}{}_{rstart}) = [c;1]$

The control place p_c is used to ensure that the firing of a stop transition can be followed only by the firing of a start transition. Note also that any start transition is enabled, and, therefore, firable, if and only if the current color marking of the modified net is $UM^{f'''}$. Thus, t_{rstart} can fire if and only if the stop transition preceding it has fired in a "proper" pre-final color marking; otherwise, there exist places $p_{jj} \in P'$ which contain tokens of color (c",2) in which case all start transitions are disabled.

Let us now define the output connections and the corresponding output color functions of the start transitions. If $UM^{o}[t_r > UM^{1}$ in ECP_{Σ}, then the color marking UM^{1} is uniquely determined by UM^{o} and t_r. Consequently, the output connections of the corresponding start transition t_{rstart} are defined such that t_{rstart} deposits upon firing in the places of the modified net the color marking $UM^{1'}$, where

$UM^{1'}/P = UM^{1}$

$UM^{1'}(p_{jj}) = \langle (c,1),(c",2)^{M^{1}(p_j)} \rangle$ for all places $p_{jj} \in P'$.

$UM^{1'}(p_c) = < (c',1) >$

Thus, the firing of t_{rstart} triggers off a (terminal) firing sequence of the modified net. A similar construct is performed for each start transition.

Note that due to the color bag of the control place p_c, all start transitions are disabled after the firing of either one of them.

We can now explain how the modified net generates the language $\Lambda_o(ECP_\Sigma, UM^f)^+$. Let us denote by ECP'_Σ the Labelled EC-CPM obtained as the result of applying the modifications presented above to the original Labelled EC-CPM ECP_Σ. Let UM^{oo} be the initial color marking of ECP'_Σ , where:

$$UM^{oo}/P \cup P' = UM^{o'}$$

$$UM^{oo}(p_c) = < (c',1) >$$

(note that the start transitions are disabled in the color marking UM^{oo}). The final color marking UM^{ff} of ECP'_Σ is defined by:

$$UM^{ff}/P \cup P' = UM^{f'}$$

$$UM^{ff}(p_c) = < (c',1) >$$

$UM^{ff} \neq UM^{oo}$ since $UM^o \neq UM^f$, by definition.

Let us first assume that there are no terminal firing sequences of the original Labelled EC-CPM ECP_Σ of length 1, i.e. there does not exist any transition $t_r \in E^o$ such that $UM^o[t_r> UM^f$. Let γ be a firing sequence of ECP_Σ. According to our induction argument, $UM^o[\gamma> UM^f$ in ECP_Σ if and only if $UM^{oo}[\gamma> UM^{ff}$ in ECP'_Σ (note that the presence of the control place p_c does not affect the respective argument). Consider now a composite firing sequence $\gamma \in T(UM^o, UM^f)^+$:

$$\gamma = t_{r1}\delta_1 t_{k1} \cdots t_{rm-1}\delta_{m-1} t_{km-1} t_{rm}\delta_m t_{km},\ m > 1$$

where $UM^o[t_{rj}\delta_j t_{kj}> UM^f$ for all j, $1 \leq j \leq m$.
According to our previous discussion, there exists the firing sequence γ' of ECP'_Σ:

$$\gamma' = t_{r1}\delta_1 t_{k1stop} \cdots t_{rm-1start}\delta_{m-1} t_{km-1stop} t_{rmstart}\delta_m t_{km}$$

and $UM^{oo}[\gamma'> UM^{ff}$. Hence, $L(\gamma) \in \Lambda_o(ECP'_\Sigma, UM^{ff})$.

We note that any terminal firing sequence of ECP'_Σ cannot start with the firing of a start transition or of a stop transition. Moreover, the firing of a start transition or of a stop transition cannot lead to the final color marking UM^{ff}. Consequently, any terminal firing sequence of ECP'_Σ which contains start and stop transitions must be of the form:

$$\delta' = t_{r1}\delta_1 t_{k1stop} t_{r2start}\delta_2 t_{k2stop}\cdots t_{rm-1start}\delta_{m-1} t_{km-1stop} t_{rmstart}\delta_m t_{km}$$

where the start and stop transitions appear explicitly. Since it is assumed that $UM^{oo}[\delta'> UM^{ff}$, we must have:

$$UM^{oo}[t_{r1}\delta_1 t_{k1stop}> UM^{f'''}$$

$$UM^{f'''}[t_{rjstart}\delta_j t_{kjstop}> UM^{f'''} \qquad \text{for all } j,\ 1 < j < m$$

$$UM^{f'''}[t_{rmstart}\delta_m t_{km}> UM^{ff}$$

Consequently, there exist the terminal firing sequences of ECP_Σ $t_{rj}\delta_j t_{kj}$, $1 \le j \le m$. Therefore, $L'(\delta') \in \Lambda_o(ECP_\Sigma, UM^f)^+$.

We can conclude that $\Lambda_o(ECP_\Sigma, UM^f)^+ = \Lambda_o(ECP'_\Sigma, UM^{ff})$.

Suppose now that there exists a transition t_r of the original Labelled EC-CPM ECP_Σ such that $UM^o[t_r> UM^f$. This case can easily be taken care of in our construct, as follows. The Labelled EC-CPM ECP'_Σ already contains a start transition t_{rstart} and a stop transition t_{rstop} corresponding to t_r. We add to ECP'_Σ yet another start transition corresponding to t_r, denoted by t_{rs}. The transition t_{rs} is assigned the same label as t_r. We set:

$$I(p_j, t_{rs}) = O(t_{rs}, p_j) = 0 \quad \text{for all places } p_j \in P$$

$$I(p_{jj}, t_{rs}) = O(t_{rs}, p_{jj}) = 1 \text{ for all places } p_{jj} \in P'$$

and $H(a_{jj}{}^{1}{}_{rs}) = [c;\ \ell.u.b.]$, $H(a_{rs}{}^{1}{}_{jj}) = [c;1]$

$$I(p_c, t_{rs}) = O(t_{rs}, p_c) = 1$$

and $H(a_{c}{}^{1}{}_{rs}) = H(a_{rs}{}^{1}{}_{c}) = [c;1]$

Clearly, t_{rs} is enabled only in the color marking $UM^{f'''}$.

Moreover, $UM^{f'''}[t_{rs}> UM^{f'''}$. This construct is performed for all transitions such as t_r. As an example of the use of the transition t_{rs}, consider the following firing sequences in $T(UM^o, UM^f)^+$:

$$\underbrace{t_r}$$
$$\underbrace{t_r\, t_{r1}\delta_1 t_{k1}}\cdots$$
$$\underbrace{t_r\delta t_k}\,\underbrace{t_r}\,\underbrace{t_r}\cdots\underbrace{t_r\, t_{rm}\delta_m t_{km}}$$
$$\cdots\underbrace{t_{rm}\delta_m t_{km}}\,\underbrace{t_r}\,\underbrace{t_r}\cdots\underbrace{t_r\, t_r}$$
$$\cdots\underbrace{t_{rj}\delta_j t_{kj}}\,\underbrace{t_r}\cdots\underbrace{t_r\, t_r\delta t_k}\,\underbrace{t_{rj}\delta_j t_{kj}}\cdots$$

(above, the individual firing sequences in $T(UM^o, UM^f)$ are bracketed). These firing sequences are simulated in our construct by:

$$\underbrace{t_r}$$
$$\underbrace{t_{rstop}\, t_{r1start}\delta_1 t_{k1stop}}\cdots$$
$$\underbrace{t_r\delta t_{kstop}}\,\underbrace{t_{rs}}\,\underbrace{t_{rs}}\cdots\underbrace{t_{rs}\, t_{rmstart}\delta_m t_{kmstop}}$$
$$\cdots\underbrace{t_{rmstart}\delta_m t_{kmstop}}\,\underbrace{t_{rs}}\,\underbrace{t_{rs}}\cdots\underbrace{t_{rs}\, t_{rstart}}$$
$$\cdots\underbrace{t_{rjstart}\delta_j t_{kjstop}}\,\underbrace{t_{rs}}\cdots\underbrace{t_{rs}\, t_{rstart}\delta t_{kstop}}\,\underbrace{t_{rjstart}\delta_j t_{kjstop}}\cdots$$

□

Note that if in the construct of Theorem 3.2.5 we set $UM^{oo} = UM^{ff} = UM^{f'''}$, then clearly $\Lambda_o(ECP'_\Sigma, UM^{f'''}) = \Lambda_o(ECP_\Sigma, UM^f)^*$ (* denotes the Kleene Star). Note, however, that the empty firing sequence is now a terminal firing sequence of the Labelled EC-CPM ECP'_Σ which is not permitted by the strict definition of the language family Λ_o(EC-CPM).

Section 3.3 CLOSURE PROPERTIES OF Λ_o(EC-CPM) UNDER MAPPINGS

In this section we investigate the closure properties of the family of terminal computation sequence sets of the EC-CPM under various substitution and transducer mappings. Let us first prove the following result we shall make use of later in this section:

Theorem 3.3.1

Any λ-free context-free language is in Λ_o(EC-CPM).

Proof: Let W be a λ-free context-free language, i.e. $\lambda \notin W$. By Theorem 4.6 of [HOPC-69], there exists a context-free grammar $G = (V_N, V_T, P, S)$ in Greibach normal form which generates W. Here, V_N denotes the set of variables, V_T denotes the set of terminal symbols, P denotes the set of productions and S denotes the start symbol of the grammar G. Even though we use the same notation, P, for the set of productions of a grammar and for the set of places of a GPN, the distinction will be clear from the context. Since G is in Greibach normal form, each production in P is of the form $A \to a\alpha$ where $A \in V_N$, $a \in V_T$ and $\alpha \in V_N^*$. We construct a Labelled EC-CPM $ECP_\Sigma = (ECP, \Sigma, L)$ and a final color marking UM^f of ECP_Σ such that $\Lambda_o(ECP_\Sigma, UM^f) = W$. Let $ECP = (CN, UM^o)$ where $CN = (N, U(C), H)$ and $N = (T, P, I, O)$.

Let us first form the set of colors $C = (X, \leq)$. Suppose $V_N = \{A_1, \ldots, A_n\}$. For each distinct variable A_j we introduce in X a distinct color c_j, $1 \leq j \leq n$. Thus, $C = (\{c_1, \ldots, c_n, c_m, c_M\}, \leq)$ where the partial ordering ≤ satisfies the conditions (I) and (II) given in Section 2.5. The extension U(C) is the color set associated with ECP_Σ.

Next, let us construct the GPN N and the mapping H. N will have only one place, let us denote it by p_s. For each production $A_j \to a\alpha_j$ of G, we introduce a distinct transition t_k in T. For each such transition t_k, $I(p_s, t_k) = 1$ and $H(a^1_{sk}) = [c_j; \ell.u.b.]$ (c_j is the color introduced for the variable A_j). If $\alpha_j = \lambda$, then $O(t_k, p_s) = 0$, otherwise, $O(t_k, p_s) = |\alpha_j|$. Let us assume that $\alpha_j = A_{j1} \ldots A_{jr}$, $r > 0$. Consequently, there are r output arcs from t_k to p_s and $H(a^i_{ks}) = [c_{ji}; \ell.u.b. + q]$ where c_{ji} denotes the color introduced for the variable A_{ji}, $1 \leq i \leq r$, and $q = r - i + 1$.

We set $\Sigma = V_T$ and define the labelling function L by $L(t_k) = a$ where a is the terminal symbol appearing in the production $A_j \to a\alpha_j$ for which the transition t_k has been created.

The initial color marking UM^o is defined by $UM^o(p_s) = \langle (c_S, 1) \rangle$ where c_S is the color introduced for the variable S. The final color marking is given by $UM^f(p_s) = \phi$.

This construct is examplified in Figure 3.3.1 for the context-free language $\{ww^R \mid w \in \{a,b\}^+\}$.

A straightforward induction argument will show that $\gamma \in T(UM^o, UM^f)$ if and only if there exists a leftmost derivation of $L(\gamma)$ in G. Consequently, $\Lambda_o(ECP_\Sigma, UM^f) = W$. □

Let us assume that the context-free grammar G above does not contain variables from which no terminal string can be derived. According to Theorem 4.2 of [HOPC-69] this can always be arranged for a context-free grammar (without affecting the requirement that G be in Greibach normal form). Note also that the Labelled EC-CPM ECP_Σ produced by the construct of Theorem 3.3.1 actually simulates leftmost derivations in G. Each terminal firing sequence of ECP_Σ leaves the place p_s empty so that the respective firing sequence cannot be continued. Some thought will show that

$$\Lambda(ECP_\Sigma) = PREF(W) = \{w' \mid w' \in V_T^* \text{ and there exists } w'' \in V_T^* \text{ such that } w'w'' \in W\}$$

Let $\mathscr{CF}$ denote the family of context-free languages and

$$PREF(\mathscr{CF}) = \{PREF(W) \mid W \in \mathscr{CF}\}$$

In connection with Theorem 3.3.1, we obtain:

Corollary 1:

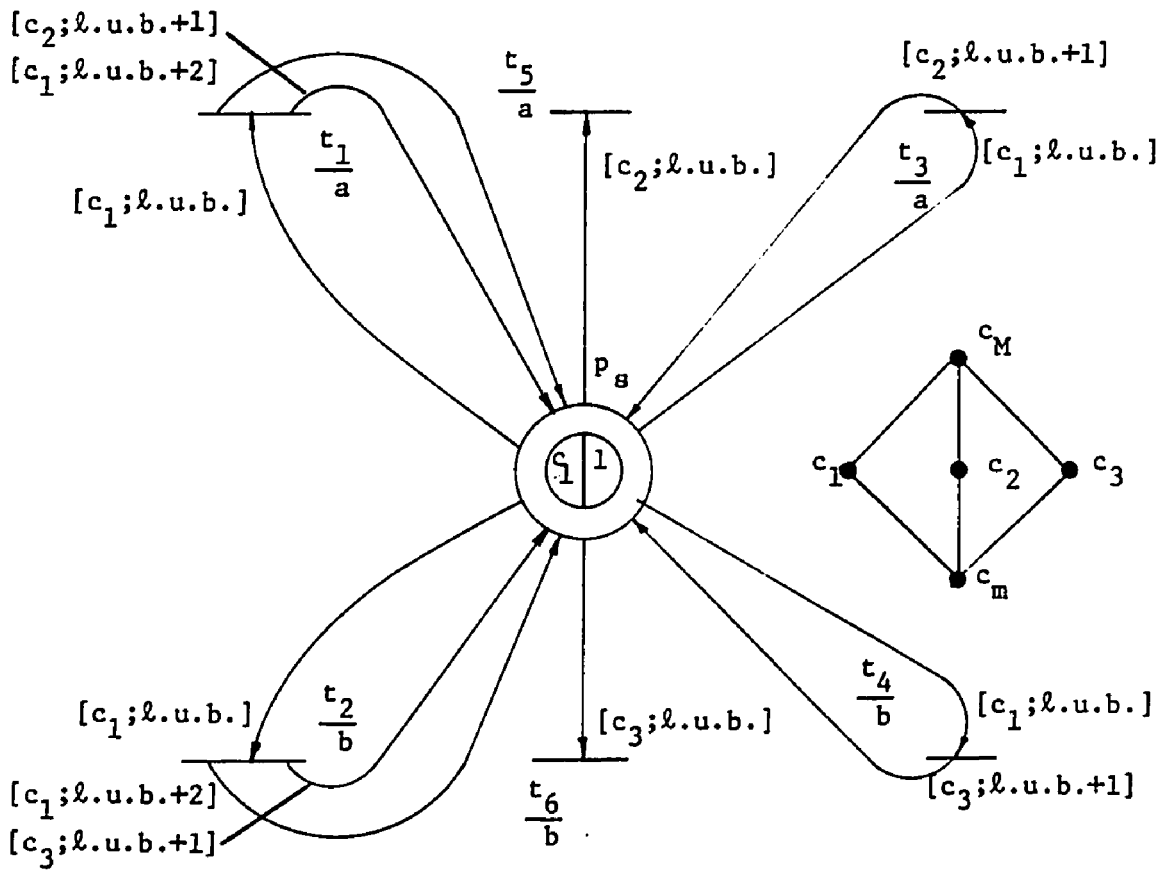
$$PREF(\mathscr{CF}) \subseteq \Lambda(\text{EC-CPM}).$$

Let Δ and Σ be finite alphabets. A substitution s is a mapping from Δ onto subsets of Σ^*. Thus, the mapping s associates with each symbol of Δ a language over the alphabet Σ. We proceed now to investigate the closure of the language family Λ_o(EC-CPM) under various types of substitution mappings.

Theorem 3.3.2

Λ_o(EC-CPM) is closed under substitution.

Proof: Let W be an arbitrary language in Λ_o(EC-CPM) over some alphabet Δ. Let s be a substitution mapping such that for each symbol $a \in \Delta$, $s(a) \in \Lambda_o$(EC-CPM). We show that $s(W) \in \Lambda_o$(EC-CPM).



$W = \{ww^R \mid w \in \{a,b\}^+\}$

P:	$A_1 \to aA_1A_2$	(t_1)	$A_1 \to bA_3$	(t_4)
	$A_1 \to bA_1A_3$	(t_2)	$A_2 \to a$	(t_5)
	$A_1 \to aA_2$	(t_3)	$A_3 \to b$	(t_6)

A_1 is the start symbol

Figure 3.3.1

Sample Labelled EC-CPM Generating a Context-Free Language

Let ECP_Δ be a Labelled EC-CPM and let UM_Δ^f be the final color marking of ECP_Δ such that $\Lambda_o(ECP_\Delta, UM_\Delta^f) = W$. For simplicity, suppose that $s(b) = \{b\}$ for all $b \in \Delta$, except for a distinguished symbol $a \in \Delta$ for which $s(a)$ is an arbitrary language in Λ_o(EC-CPM), not necessarily a singleton set. Suppose also that ECP_Δ contains only one transition, let us denote it by t_a, which has the label a. At the end of this proof we remove these constraints and extend our construct for the general case.

Let now ECP_Σ be a Labelled EC-CPM and let UM^f be the final color marking of ECP_Σ. Let $s(a) = \Lambda_o(ECP_\Sigma, UM^f)$.

Suppose we perform on ECP_Σ the construct of Theorem 3.2.5. Let ECP_Σ' denote the resulting Labelled EC-CPM. If we define the initial color marking UM^{oo} and the final color marking UM^{ff} of ECP_Σ' by $UM^{oo} = UM^{ff} = UM^{f'''}$, then $\Lambda_o(ECP_\Sigma', UM^{ff}) = \Lambda_o(ECP_\Sigma, UM^f)^*$. Here, $UM^{f'''}$ has the same meaning as in the proof of Theorem 3.2.5 (in the remainder of this proof the notational conventions regarding the places, transitions and color markings of ECP_Σ and ECP_Σ' are those employed in the proof of Theorem 3.2.5).

Let us now compose the Labelled EC-CPM's ECP_Δ and ECP_Σ' in order to obtain a Labelled EC-CPM which generates $s(W)$ as an Λ_o-language. First, we eliminate the transition t_a from ECP_Δ. Next, we let all remaining transitions of ECP_Δ self-loop on the control place p_c of ECP_Σ'. For each such transition t_q we set:

$$H(a_{cq}^1) = H(a_{qc}^1) = [c;1]$$

Let us denote by P" the set of places of ECP_Δ, by P the set of places of the original Labelled EC-CPM ECP_Σ and by P' the set of counter places introduced in ECP_Σ'. We connect each start transition of ECP_Σ' to the places in P" the same way as t_a was connected. Consequently, a start transition of ECP_Σ' is enabled if and only if the places in P" contain a color marking in which t_a would have been enabled and the places in $P \cup P' \cup \{p_c\}$ contain the color marking $UM^{f'''}$. Also, the firing of a start transition has the same effect with respect to the places in P" as a firing of t_a.

Let $ECP_{\Delta\Sigma}$ be the Labelled EC-CPM obtained by interconnecting ECP_{Δ} and ECP'_{Σ} as shown above. The initial color marking $UM^{o}_{\Delta\Sigma}$ of $ECP_{\Delta\Sigma}$ is defined by:

$UM^{o}_{\Delta\Sigma}/P'' = UM^{o}_{\Delta}$ (UM^{o}_{Δ} is the initial color marking of ECP_{Δ})

$UM^{o}_{\Delta\Sigma}/P \cup P' \cup \{p_c\} = UM^{f'''}$.

The final color marking $UM^{f}_{\Delta\Sigma}$ of $ECP_{\Delta\Sigma}$ is given by:

$UM^{f}_{\Delta\Sigma}/P'' = UM^{f}_{\Delta}$

$UM^{f}_{\Delta\Sigma}/P \cup P' \cup \{p_c\} = UM^{f'''}$.

Note that $UM^{o}_{\Delta\Sigma} \neq UM^{f}_{\Delta\Sigma}$ since $UM^{o}_{\Delta} \neq UM^{f}_{\Delta}$, by definition.

Since $UM^{o}_{\Delta\Sigma}(p_c) = < (c,1) >$ the transitions enabled in the initial color marking $UM^{o}_{\Delta\Sigma}$ can be transitions of ECP_{Δ} and/or start transitions of ECP'_{Σ}. The firing of any transition corresponding to ECP_{Δ} preserves this color bag of the control place p_c. Only the firing of a start transition of ECP'_{Σ} can change the color bag of p_c to $< (c',1) >$ in which case all transitions corresponding to ECP_{Δ} and the start transitions of ECP'_{Σ} are disabled. At the same time the firing of the respective start transition triggers off a firing sequence in ECP'_{Σ}. Thus, a substitution firing sequence cannot be interrupted by firings of transitions from ECP_{Δ}. Only the firing of a stop transition of ECP'_{Σ} will restore the token of color (c,1) in the control place p_c and thus eventually reenable the transitions of ECP_{Δ}. If the stop transition fires in a "proper" pre-final color marking, then the restriction of the color marking obtained upon its firing to $P \cup P' \cup \{p_c\}$ is $UM^{f'''}$. In this case, the Labelled EC-CPM ECP'_{Σ} can be reused to generate subsequent substitution firing sequences. Otherwise, the start transitions of ECP'_{Σ} cannot become enabled again and therefore, ECP'_{Σ} cannot be restarted along the respective firing sequence of $ECP_{\Delta\Sigma}$. Moreover, that firing sequence cannot lead to the final color marking UM^{ff} since $UM^{ff}/P \cup P' \cup \{p_c\} = UM^{f'''}$.

Suppose now that the Labelled EC-CPM ECP_{Δ} contains m transitions, $t_{a1},\ldots,t_{am}$, which carry the same label a. In this case we have to provide the substitution Labelled EC-CPM ECP'_{Σ} with a separate set of

start transitions corresponding to each such transition t_{aj}, $1 \leq j \leq m$. Each set of start transitions is connected to the places in $P \cup P' \cup \{p_c\}$ as shown in the proof of Theorem 3.2.5. In addition, the start transitions corresponding to t_{aj}, $1 \leq j \leq m$, are connected to the places in P" the same way as t_{aj}.

The construct described above is applied for each symbol in Δ.

□

A substitution mapping s on Δ is called a substitution with λ-free context-free languages if for all symbols $a \in \Delta$, $s(a)$ is a λ-free context-free language. Similarly, the substitution mapping s is called a substitution with λ-free regular languages if $s(a)$ is a λ-free regular language for all $a \in \Delta$. Obviously, substitution with λ-free regular languages is a particular case of substitution with λ-free context-free languages.

A substitution mapping s is called a non-erasing homomorphism if for all $a \in \Delta$, $s(a)$ is a singleton set containing a nonempty, finite-length string. Clearly, a non-erasing homomorphism is a special case of substitution with λ-free regular languages.

From Theorems 3.3.1 and 3.3.2 we immediately have:

Corollary 1

Λ_o(EC-CPM) is closed under substitution with λ-free context-free languages, λ-free regular languages and non-erasing homomorphism.

In Section 3.4 we shall show that the language family Λ(EC-CPM) is not closed under any of these well known substitution types.

The quasi-realtime languages are the languages accepted by nondeterministic multitape Turing machines in real time. Following characterization was given in [BOOK-70b]:

Every quasi-realtime language can be expressed as the length preserving homomorphic image of the intersection of three context-free languages.

From Theorems 3.3.1, 3.2.4 and Corollary 1 above we have:

Corollary 2

Every λ-free quasi-realtime language is in Λ_o(EC-CPM).

Let us now investigate the closure of the language family Λ_o(EC-CPM) under transducer mappings. We start with pushdown transducer mappings.

A pushdown transducer (PDT) can be viewed as a pushdown automaton with an output. On each move, the PDT generates a finite-length output string. In what follows we briefly define the PDT. For a more detailed discussion on this subject the reader is referred to [AHO-72]; our notations are compatible with the notations employed there.

A PDT is a system $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_1, Z_1)$ where Q is a finite set of states, Σ is a finite input alphabet, Γ is a finite pushdown store alphabet and Δ is a finite output alphabet. The initial state of P is $q_1 \in Q$ and the initial pushdown store symbol is $Z_1 \in \Gamma$. The mapping δ is defined from $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$ into finite subsets of $Q \times \Gamma^* \times \Delta^+$. Even though we use the same symbol P to denote both a PDT and the set of places of a GPN, the distinction will be clear from the context.

A configuration of the PDT P is a 4-tuple (q, w, α, y) where $q \in Q$, $w \in \Sigma^*$ denotes the input string remaining to be read, $\alpha \in \Gamma^*$ the current content of the pushdown store and and $y \in \Delta^*$ denotes the output string emitted so far. If $(r, \alpha, z) \in_{\gamma} \delta(q, a, Z)$ then we write $(q, aw, Z\gamma, y) \vdash (r, w, \alpha\gamma, yz)$ for all $w \in \Sigma^*$, $\gamma \in \Gamma^*$ and $y \in \Delta^*$.

The translation defined by P by empty pushdown store is:

$\tau_e(P) = \{(w, y) \mid (q_1, w, Z_1, \lambda) \overset{*}{\vdash} (q, \lambda, \lambda, y)$ for some $q \in Q\}$

Given a language $W \subseteq \Sigma^*$, the image of W under the translation defined by P is:

$\mu_P(W) = \{y \mid (w, y) \in \tau_e(P)$ for some $w \in W\}$

We note that in the above definition, the PDT P generates a nonempty output string on each move. Such a PDT is called a λ-output free PDT. In our study, we consider only λ-output free pushdown transducers.

Theorem 3.3.3

Λ_o(EC-CPM) is closed under λ-output free PDT mappings.

Proof: Let ECP_Σ be an arbitrary Labelled EC-CPM and let UM^f be

the final color marking of ECP_Σ. Let $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_1, Z_1)$ be an arbitrary PDT. We want to construct a Labelled EC-CPM ECP_Δ and a final color marking UM^{ff} of ECP_Δ such that $\Lambda_o(ECP_\Delta, UM^{ff}) = \mu_P(\Lambda_o(ECP_\Sigma, UM^{t}))$.

First we construct a Labelled EC-CPM ECP'_Σ which simulates the PDT P. The construct is similar to that of Theorem 3.3.1. For simplicity, let us first assume that the PDT P generates only one-symbol output strings on each move; at the end of this proof we shall discuss the general case.

The states of P and the symbols in the pushdown store alphabet Γ are encoded in terms of colors from the finite color set C of ECP'_Σ. Suppose $Q = \{q_1,\ldots,q_r\}$ and $\Gamma = \{Z_1,\ldots Z_n\}$. For each state q_j we introduce a distinct color c_j, $1 \le j \le r$, and for each symbol Z_j we introduce a distinct color c_j, $r + 1 \le j \le r + n$. Thus, $C = (\{c_1,\ldots,c_{r+n},c_m,c_M\}, \le)$ where the partial ordering $\le$ satisfies the conditions (I) and (II) given in Section 2.5. The extension U(C) is the color set associated with ECP'_Σ.

Next, let us construct the underlying GPN N of ECP'_Σ. N will have two places, denoted by p_Q and p_S, respectively. Suppose now that $(q_i, \gamma_r, a) \in \delta(q_s, b, Z_j)$. We introduce in N a transition, let us denote it by t_q, which simulates this move of P. We set:

$$I(p_Q, t_q) = O(t_q, p_Q) = 1$$

and $H(a^1_{Qq}) = [c_s;1]$, $H(a^1_{qQ}) = [c_i;1]$

where c_s and c_i are the colors introduced for the states q_s and q_i, respectively. In addition, we set:

$$I(p_S, t_q) = 1 \text{ and } H(a^1_{Sq}) = [c_j;\ell.u.b.]$$

where c_j is the color introduced for the pushdown store symbol Z_j. If $\gamma_r = \lambda$ then $O(t_q, p_S) = 0$. Suppose now that $\gamma_r = Z_{r1}\ldots Z_{rt}$, $t > 0$. Then $O(t_q, p_S) = |\gamma_r|$ and $H(a^s_{qS}) = [c_{rs};\ell.u.b. + m]$ where $m = t - s + 1$ and $1 \le s \le t$. Here, c_{rs} denotes the color introduced for the pushdown symbol Z_{rs}, $1 \le s \le t$. The transition t_q is assigned the label $b \in (\Sigma \cup \{\lambda\})$. This consturct is performed for each 3-tuple $(q_s, b, Z_j) \in Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$ for which the mapping δ is defined and for all 3-tuples $(q_i, \gamma_r, a) \in \delta(q_s, b, Z_j)$.

In our construct we use the color marking of the place p_Q to encode the current state of the PDT while the color marking of the place p_S is used to encode the current content of the pushdown store. The color bags of these places in the initial and final color markings will be defined accordingly.

Let us now consider the problem of providing the PDT P with an input. Recall that the translation $\tau_e(P)$ is to be applied to the Λ_0-language generated by the Labelled EC-CPM ECP_Σ. Thus, the label sequences corresponding to all terminal firing sequences of ECP_Σ are the input strings of P. On the other hand, if t_q is a transition of the Labelled EC-CPM ECP'_Σ and t_q has been introduced for some 3-tuple $(q_i, \gamma_r, a) \in \delta(q_s, b, Z_j)$ then t_q carries the label b, i.e. the input corresponding to the move of the PDT P simulated by t_q. Consequently, for each symbol $b \in \Sigma$, we combine the transitions of ECP'_Σ carrying the label b with the transitions of ECP_Σ carrying the label b the same way as in the construct for the intersection operation. Note, however, that for the transition t_q above the label b may be λ. In the case of a λ-input move, the respective move depends only on the state of the PDT P and on the symbol in the top position of the pushdown store. Consequently, in our construct the transitions which simulate λ-input moves of P are carried over directly to the composite net, without being combined with any transitions of ECP_Σ. The enabled status of a transition in this category depends only on the color markings of the places p_Q and p_S.

After the composite EC-CPM has been formed, all transitions are relabelled according to the one-symbol output strings generated by the corresponding moves of the PDT P. Thus, the transitions obtained by eventually combining the transition t_q above with transitions of the Labelled EC-CPM ECP_Σ are assigned the label a. Since we have assumed P to be λ-output free, after relabelling, all transitions must have a label other than λ. Moreover, the label sequences generated by the composite Labelled EC-CPM are in Δ^+.

The initial color marking UM^{oo} of the composite net is:

$$UM^{oo}/P = UM^{o}$$

$$UM^{oo}(p_Q) = < (c_1,1) >$$

$$UM^{oo}(p_S) = < (c_{r+1},1) >$$

where P denotes the set of places of the Labelled EC-CPM ECP_Σ, UM^o denotes the initial color marking of ECP_Σ and c_1 and c_{r+1} denote the colors introduced for the initial state q_1 of P and the initial push-down store symbol Z_1, respectively.

Let us now consider the final color marking UM^{ff} of the composite net. For the places of the original Labelled EC-CPM ECP_Σ we set $UM^{ff}/P = UM^f$. Since we are simulating a PDT translation by empty store, $UM^{ff}(p_S) = \phi$. Note, however, that at the end of a successful computation, the PDT P may be in any state $q_j \in Q$. In order to ensure a unique final color marking of the place p_Q, we introduce a set of stop transitions. Consider the transition t_q of ECP'_Σ corresponding to the 3-tuple $(q_i, \gamma_r, a) \in \delta(q_s, b, Z_j)$. The transition t_q can leave upon firing the place p_S empty only if $\gamma_r = \lambda$, i.e. only if $O(t_q, p_S) = 0$. Consequently, only this type of transitions is likely to fire "last." Therefore, in the composite net, for each transition which does not have p_S as an output place, we add a copy of it with the same label, the same input and output connections and the same corresponding color threshold and color output functions, except for the place p_Q. These copies, which are the stop transitions of the composite net, remove but do not restore upon firing the token in p_Q, thus leaving all transitions of the composite net disabled. Then, $UM^{ff}(p_Q) = \phi$.

Each stop transition can fire exactly when the transition it "parallels" can fire and with the same effect (with the exception of p_Q). Nevertheless, if a stop transition fires prematurely, for example the place p_S does not become empty upon its firing, then the respective firing sequence cannot be continued and, therefore, cannot lead to the final color marking UM^{ff}.

This completes the construction of the Labelled EC-CPM ECP_Δ.

At the beginning of this proof we have assumed that P emits only one-symbol output strings on any move. Let now $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_1, Z_1)$ be a λ-output free PDT which can emit output strings containing more than one symbol. Let

$P' = (Q, \Sigma, \Gamma, \Delta', \delta', q_1, Z_1)$ be a PDT obtained by modifying P such that each move of P' generates a distinct one-symbol output string. Clearly, for each language $W \subseteq \Sigma^*$:

$$\mu_P(W) = h(\mu_{P'}(W))$$

where h is a non-erasing homomorphism. Since Λ_o(EC-CPM) is closed under non-erasing homomorphism, Theorem 3.3.3 follows. □

Let us now consider another class of transducer mappings, namely the finite state transducer (FST) mappings. An FST is a finite state automaton with an output (a detailed definition of the FST can be found in [AHO-72]). On each move, the FST emits a finite-length output string. The construct of Theorem 3.3.3 can easily be particularized for this type of transducers. Since an FST does not incorporate a pushdown store, the place p_S is not required any more. Consequently, the transitions of the Labelled EC-CPM ECP'_Σ merely self-loop on the place p_Q, each simulating a state transition of the FST. In order to ensure a unique final color marking for the place p_Q the respective FST can be modified such that it has a unique final state rather than a set of final states. Alternatively, we can introduce stop transitions corresponding to the state transitions of the FST in a final state. The stop transitions remove upon firing the token from the place p_Q and thus leave the entire composite net disabled. Thus:

Corollary 1

Λ_o(EC-CPM) is closed under λ-output free FST mappings.

In Section 3.4 we show that the language family Λ(EC-CPM) is not closed under PDT or FST mappings.

Large systems are in general constructed from the composition of several smaller subsystems. If we imagine these component subsystems as being represented each by individual EC-CPM's, it is useful to know what interconnections are possible among EC-CPM's and how do they affect the language generated by the resulting net. This point of view is especially relevant when one is interested in a top-down hierarchical modelling of concurrent systems. Comparing the results presented in Sections 3.2 and 3.3 with the results presented in [HACK-75] and

[PETE-73] relative to the GPN's, we see that the EC-CPM's preserve the closure properties of the GPN's under the usual composition operations of union, intersection, concatenation, etc., and for certain operations extend these closure properties. For example, the language family Λ_o generated by the GPN's is not closed under indefinite concatenation and substitution ([HACK-75]). From the results of [HACK-75] it also follows that this Λ_o language family is not closed under PDT mappings either.

Section 3.4 THE EXTENTS OF THE LANGUAGE FAMILIES Λ(EC-CPM) and Λ_o(EC-CPM).

A question which can naturally be asked is how do the language families Λ(EC-CPM) and Λ_o(EC-CPM) relate to the family of regular languages ($\mathscr{R}$), the family of context-free languages ($\mathscr{CF}$) and the family of context-sensitive languages ($\mathscr{CS}$). A partial answer to this question was given by Theoren 3.3.1. In this section, among other things, we further pursue this topic.

Theorem 3.4.1

Λ(EC-CPM) $\cap \mathscr{R} \neq \phi$ but $\mathscr{R} \not\subseteq \Lambda$(EC-CPM)

Λ(EC-CPM) $\cap$ ($\mathscr{CF}$-$\mathscr{R}$) $\neq \phi$ but ($\mathscr{CF}$-$\mathscr{R}$) $\not\subseteq \Lambda$(EC-CPM)

Λ(EC-CPM) $\cap$ ($\mathscr{CS}$-$\mathscr{CF}$) $\neq \phi$ but ($\mathscr{CS}$-$\mathscr{CF}$) $\not\subseteq \Lambda$(EC-CPM).

Proof: Consider the regular language $a^*c + \lambda$. Using an argument similar to that of Corollary 1 of Theorem 3.2.1, we show that this language is not in Λ(EC-CPM).

Suppose there exists a Labelled EC-CPM ECP_Σ such that $\Lambda(ECP_\Sigma) = a^*c + \lambda$. Let m be a positive integer. Obviously, the string a^mc must be in $\Lambda(ECP_\Sigma)$. Consequently, there exists a firing sequence $\gamma = t_{k1}\ldots t_{km}t_j$ in $S(UM^o)$ such that $L(\gamma) = L(t_{k1})\ldots L(t_{km})L(t_j) = a^mc$. But, the prefix firing sequence $\gamma' = t_{k1}\ldots t_{km}$ of γ is in $S(UM^o)$ as well. Hence, the string $L(\gamma') = a^m$ is in $\Lambda(ECP_\Sigma)$. Since any non-empty string in the language $a^*c + \lambda$ must end with the symbol c, it follows that $\Lambda(ECP_\Sigma) \neq a^*c + \lambda$, which contradicts our assumption.

We note that, in particular, all regular languages of the form {w}, where w is a finite-length string and $|w| > 1$, are not in Λ(EC-CPM).

By the same argument, it can be shown that the context-free language $W = \{a^n b^n \mid n \geq 0\}$ and the context-sensitive language $W' = \{a^n b^n c^n \mid n \geq 0\}$ are not in Λ(EC-CPM) either.

On the other hand, it is trivial to construct a Labelled EC-CPM ECP_Σ such that $\Lambda(ECP_\Sigma) = a^*$. Thus, Λ(EC-CPM) $\cap \mathscr{R} \neq \phi$. Also, by Corollary 1 of Theorem 3.3.1, the language PREF(W) = $\{a^n b^m \mid 0 \leq m \leq n\}$ is in Λ(EC-CPM). Since PREF(W) is strictly a context-free language, i.e. PREF(W) is context-free but not regular, it follows that Λ(EC-CPM) $\cap$ ($\mathscr{CF}$-$\mathscr{R}$) $\neq \phi$. Finally, consider the Labelled EC-CPM ECP'_Σ of Figure 3.4.1. It can be seen that $\Lambda_o(ECP'_\Sigma, UM^{f'}) = W' - \{\lambda\}$ while $\Lambda(ECP'_\Sigma) = \{a^n b^m c^q \mid 0 \leq q \leq m \leq n\}$. $\Lambda(ECP'_\Sigma)$ is strictly a context-sensitive language and, therefore, Λ(EC-CPM) $\cap$ ($\mathscr{CS}$-$\mathscr{CF}$) $\neq \phi$. □

Consider the regular language $W = \{a\}$. By Theorem 3.3.1 $W \in \Lambda_o$(EC-CPM) and by Corollary 1 of Theorem 3.3.1 the language $W' = PREF(W) = \{\lambda, a\}$ is in Λ(EC-CPM). Let W'' be an arbitrary language and let s be the substitution mapping defined by $s(a) = W''$. Hence, $s(W') = W'' \cup \{\lambda\}$. From Theorem 3.4.1, we immediately have:

Corollary 1

Λ(EC-CPM) is not closed under substitution with (λ-free) context-free languages, (λ-free) regular languages and non-erasing homomorphism.

Note, however, that Λ(EC-CPM) is closed under λ-free renaming.

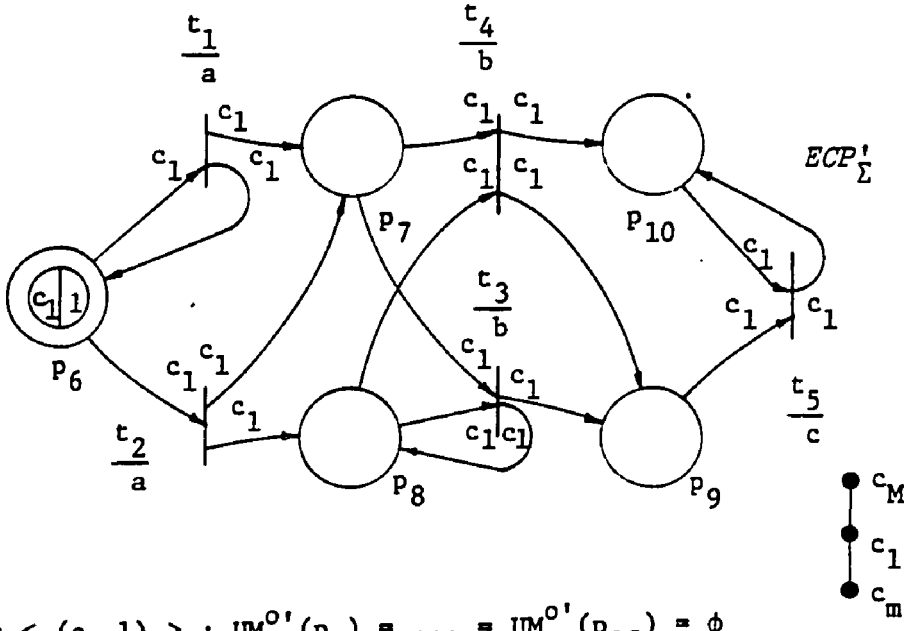
Consider now the regular language $W = a^*$. We have already mentioned that $W \in \Lambda$(EC-CPM). It can be shown that for any context-free language W' there exists a PDT P such that $\mu_P(W) = W'$ (λ-output free if $\lambda \notin W'$). A proof is omitted since it would basically parallel the proof of Theorem 3.3.1. Similarly, it can be shown that for any regular language W'', there exists an FST F such that $\mu_F(W) = W''$ (λ-output free if $\lambda \notin W''$). Hence, we obtain:

Corollary 2

Λ(EC-CPM) is not closed under (λ-output free) PDT mappings and (λ-output free) FST mappings.

From Theorem 3.3.1 we already know that all λ-free context-free languages are in Λ_o(EC-CPM). From the proof of Theorem 3.4.1 we see that there exist context-sensitive languages, such as $\{a^n b^n c^n \mid n > 0\}$, which are in Λ_o(EC-CPM) as well.

Let now ECP_Σ be an arbitrary Labelled EC-CPM and let w be a string in Σ^*. Since the labelling function L corresponding to ECP_Σ is a λ-free renaming, w is in $\Lambda_o(ECP_\Sigma, UM^f)$ ($\Lambda(ECP_\Sigma)$) if and only if there exists a firing sequence γ in $T(UM^o, UM^f)$ ($S(UM^o)$) such that $L(\gamma) = w$. The firing sequence γ need not be unique, nevertheless, all firing sequences which generate w must have the same length, namely $|w|$. There are at most $|T|^{|w|}$ firing sequences of length $|w|$, where T



$UM^{o'}(p_6) = \langle (c_1,1) \rangle$; $UM^{o'}(p_7) = \ldots = UM^{o'}(p_{10}) = \phi$

$UM^{f'}(p_6) = \ldots = UM^{f'}(p_9) = \phi$; $UM^{f'}(p_{10}) = \langle (c_1,1) \rangle$

$\Lambda_o(ECP'_\Sigma, UM^{f'}) = \{a^n b^n c^n \mid n > 0\}$

$\Lambda(ECP'_\Sigma) = \{a^n b^m c^q \mid 0 \leq q \leq m \leq n\}$

Figure 3.4.1
Sample Labelled EC-CPM

denotes the set of transitions of ECP_Σ. Consequently, the languages $\Lambda(ECP_\Sigma)$ and $\Lambda_o(ECP_\Sigma, UM^f)$ are recursive sets. We have, however, proved the following stronger statement:

Theorem 3.4.2

$\Lambda(\text{EC-CPM}) \subset \mathscr{DCS}$

$\Lambda_o(\text{EC-CPM}) \subseteq \mathscr{DCS}$

where $\mathscr{DCS}$ denotes the family of deterministic context-sensitive languages.

Proof: See Appendix B. □

Let us define

$\text{TIME}(f) = \{W(TM) \mid TM \text{ is a non-deterministic multi-tape Turing acceptor which operates within time bound } f\}$

Corollary 1

$\Lambda(\text{EC-CPM}) \subseteq \text{TIME}(x^2)$

$\Lambda_o(\text{EC-CPM}) \subseteq \text{TIME}(x^2)$.

Proof: See Appendix B.

Let Δ and Σ be finite alphabets. We shall call a homomorphism h from Δ^* into Σ^* a renaming if for each symbol $a \in \Delta$ either $h(a) = b$, for some $b \in \Sigma$, or $h(a) = \lambda$. Obviously, a λ-free renaming (as defined in Section 3.1) is a non-erasing renaming.

For any family of languages $\mathscr{A}$, the image of $\mathscr{A}$ under renaming is the set:

$H_\lambda(\mathscr{A}) = \{h(A) \mid A \in \mathscr{A} \text{ and } h \text{ is a renaming on } A\}$.

Let $\mathscr{RE}$ denote the family of recursively enumerable languages.

Theorem 3.4.3

$H_\lambda(\Lambda_o(\text{EC-CPM})) = \mathscr{RE}$.

Proof: By Theorem 1.1.1 of [SALO-73] each recursively enumerable language W can be expressed in the form:

$$W = h(W_1 \cap W_2)$$

where h is a homomorphism and W_1 and W_2 are context-free languages. In fact, from the proof of that theorem appears that h is a renaming

and W_1 and W_2 are λ-free context-free languages. But, for any two λ-free context-free languages W_1 and W_2, by Theorems 3.3.1 and 3.2.4, $W_1 \cap W_2$ is a language in Λ_o(EC-CPM). Hence, $\mathscr{RE} \subseteq H_\lambda(\Lambda_o(\text{EC-CPM}))$.

On the other hand, by Theorem 3.4.2 any language $W \in \Lambda_o$(EC-CPM) is deterministic context-sensitive and, therefore, a recursively enumerable language. Moreover, the family $\mathscr{RE}$ is closed under arbitrary homomorphism (Corollary 9.1 of [HOPC-69]). Hence, $H_\lambda(\Lambda_o(\text{EC-CPM})) \subseteq \mathscr{RE}$.

□

Corollary 1

Λ_o(EC-CPM) is not closed under arbitrary homomorphism.

Let ECP_Σ be an arbitrary Labelled EC-CPM generating some language W over the alphabet Σ. Let h be a renaming on W. Applying the renaming h to W amounts to relabelling the transitions of ECP_Σ according to the definition of h. If h is not a λ-free renaming then some transitions of ECP_Σ will receive the label λ in the process of relabelling. We shall call such transitions λ-transitions. In the string generated by a firing sequence of an EC-CPM with λ-transitions, only the occurrence of transitions whose label is different from λ is recorded. Note that a string generated by an EC-CPM with λ-transitions can in fact correspond to an arbitrarily long firing sequence, the number of λ-transitions fired between the consecutive firing of two transitions whose label is not λ being transparent.

Note also that a Labelled EC-CPM with λ-transitions is not a Labelled EC-CPM in the strict sense of Definition 3.1.1 since the labelling function is required to be a total mapping from T into Σ. Nevertheless, the labelling function can be extended into $\Sigma \cup \{\lambda\}$.

CHAPTER IV

THE REPRESENTATION CAPABILITIES OF THE C-COLORED PETRI MODEL

Section 4.1 INTRODUCTION

In this chapter we shall investigate the extents of the language families Λ(C-CPM) and Λ_o(C-CPM). In our study we shall compare the C-CPM to several other formal models of asynchronous concurrent systems. In this introductory section we shall define two of these models, namely the Priority Petri Model ([HACK-75]) and the Extended Petri Model ([AGAR-75]). The forms of these definitions are our own.

Definition 4.1.1

A Priority Petri Net (PPN) is a system PN = (N, Pr, Y) where:

1. N = (T, P, I, O) is a Generalized Petri Net.
2. Pr = (Q, $\underline{\alpha}$) is a finite partially ordered set.
3. Y: T → Q is a total, single-valued mapping. For any transition $t_k \in T$, $Y(t_k)$ is called the priority of t_k.

As with any GPN, a marking of a Priority Petri Net is defined as a total, single-valued mapping from the set of places P into Z^o, the set of nonnegative integers. Then:

Definition 4.1.2

A Priority Petri Model (PPM) is a system $\mathcal{PN}$ = (PN, M^o) where:

1. PN = (N, Pr, Y) is a Priority Petri Net.
2. M^o is the initial marking of PN.

Definition 4.1.3

A Labelled Priority Petri Model is a system $\mathcal{PN}_\Sigma$ = ($\mathcal{PN}$, Σ, L) where:

1. $\mathcal{PN}$ = (PN, M^o) is a PPM.
2. Σ is a finite label alphabet.
3. L: T → Σ is a λ-free renaming called the labelling function of $\mathcal{PN}_\Sigma$.

Let PN = (N, Pr, Y) be a PPN in some marking M^i.

Definition 4.1.4

A transition $t_k \in T$ is said to be enabled in the marking M^i if and only if $M^i(p_j) \geq I(p_j, t_k)$ for each place $p_j \in \mathcal{D}(I_k)$.

A transition t_k, enabled in some marking M^i, may be selected to fire.

Definition 4.1.5

If an enabled transition t_k fires in the marking M^i and $M^i [t_k > M^{i+1}$ then for each $p_j \in P$:

$$M^{i+1}(p_j) = M^i(p_j) - I(p_j, t_k) + O(t_k, p_j).$$

Let E^i denote the set of transitions enabled in the marking M^i. Let us also define for each $t_k \in T$ the set:

$$T_k = \{t_s \mid t_s \in T \text{ and } Y(t_k) \alpha Y(t_s)\}$$

(Note that for any $t_k \in T$ and $t_s \in T$, $Y(t_k) \alpha Y(t_s)$ means that $Y(t_k) \underline{\alpha} Y(t_s)$ and $Y(t_k) \neq Y(t_s)$).

Definition 4.1.6

A transition t_k can be selected to fire in the marking M^i (i.e., t_k is <u>firable</u> in M^i) if and only if the following conditions hold:

1. $t_k \in E^i$
2. $T_k \cap E^i = \phi$.

Otherwise stated, a transition t_k is firable in the marking M^i if and only if no transition t_s, whose priority $Y(t_s)$ is strictly higher than $Y(t_k)$, is also enabled in M^i.

We shall proceed now to define the Extended Petri Model.

Definition 4.1.7

An <u>Extended Petri Net</u> (EPN) is a modified Generalized Petri Net EN = (T, P, I, O) such that:

1. T is a finite set of transitions; $T = \{t_1, \ldots, t_n\}$
2. P is a finite set of places, $P = \{p_{n+1}, \ldots, p_{n+m}\}$ where $n = |T|$, $m = |P|$ and $T \cap P = \phi$
3. $I: P \times T \rightarrow Z^o \cup \{\zeta\}$ is a total, single-valued mapping called the input incidence function. ζ is a special symbol, $\zeta \notin Z^o$.
4. $O: T \times P \rightarrow Z^o$ is the output incidence function.

Let $(p_j, t_k) \in P \times T$. If $I(p_j, t_k) = s$, where $s \in Z^+$, then p_j is called a normal input place of t_k and s directed arcs $a_{jk}^1, \ldots, a_{jk}^s$ connect p_j to t_k. If $I(p_j, t_k) = \zeta$ then p_j is called an inhibitor input place of t_k. In this case p_j is connected to t_k by a unique arc of a special type, called an inhibitor arc or a zero testing arc. Graphically, an inhibitor arc is represented as a directed arc with a slash across it (Figure 4.1.1). Since the mapping I is single-valued we notice that a place p_j cannot be both a normal and an inhibitor input place for some transition t_k. Nevertheless, p_j may be a normal input place for some transition t_k and an inhibitor input place for some other transition t_s.

A marking of an EPN is defined as usual, namely as a total, single-valued mapping from the set of places into Z°. Thus:

Definition 4.1.8

An Extended Petri Model (EPM) is a system $\mathcal{EN} = (EN, M^\circ)$ where:

1. $EN = (T, P, I, O)$ is an EPN.
2. M° is the initial marking of EN.

Definition 4.1.9

A Labelled Extended Petri Model is a system $\mathcal{EN}_\Sigma = (\mathcal{EN}, \Sigma, L)$ where:

1. $\mathcal{EN} = (EN, M^\circ)$ is an EPM.
2. Σ is a finite label alphabet.
3. $L: T \to \Sigma$ is a total, single-valued mapping called the labelling function of $\mathcal{EN}_\Sigma$.

Let $EN = (T, P, I, O)$ be an EPN in some marking M^i and let t_k be an arbitrary transition of EN. We shall denote by P_k^n the set of normal input places of t_k and by P_k^z the set of inhibitor input places of t_k. Obviously, $\mathcal{D}(I_k) = P_k^n \cup P_k^z$ and $P_k^n \cap P_k^z = \phi$.

Definition 4.1.10

A transition $t_k \in T$ is enabled in the marking M^i if and only if $M^i(p_j) \geq I(p_j, t_k)$ for each $p_j \in P_k^n$ and $M^i(p_r) = 0$ for each $p_r \in P_k^z$.

Any transition t_k, enabled in the marking M^i, may be selected to fire. Thus, in the case of the EPM, the notions of firable transition

and enabled transition coincide.

Definition 4.1.11

If a transition t_k, enabled in the marking M^i fires in M^i and $M^i[t_k > M^{i+1}$ then for each $p_j \in P$:

$$M^{i+1}(p_j) = \begin{cases} M^i(p_j) - I(p_j, t_k) + O(t_k, p_j) & \text{if } p_j \in P - P_k^z \\ M^i(p_j) + O(t_k, p_j) = O(t_k, p_j) & \text{if } p_j \in P_k^z \end{cases}$$

We can see now that the inhibitor arcs are indeed a special kind of arcs since they do not carry tokens during the process of firing of a transition. The function of an inhibitor arc is only to "sense" whether in the current marking a certain input place of the corresponding transition is empty.

Figure 4.1.1 exhibits a sample EPN. In the given marking transition t_2 is enabled while t_1 is disabled.

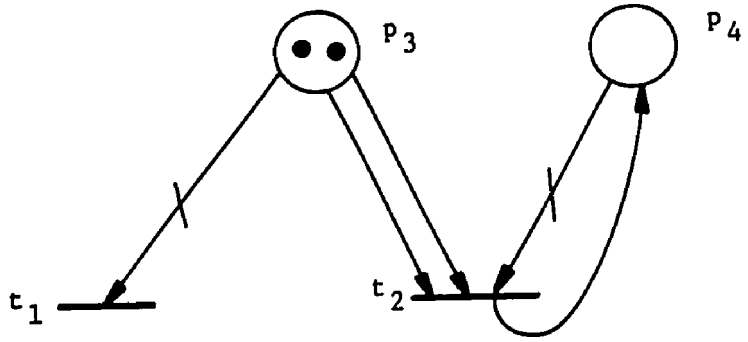


Figure 4.1.1

Sample Extended Petri Net

The families of languages Λ(PPM), Λ_o(PPM), Λ(EPM) and Λ_o(EPM) are defined completely analogous to Λ(C-CPM) and Λ_o(C-CPM).

The PPM was introduced in [HACK-75] and the EPM was first defined in [AGAR-75]. Following result which relates the PPM to the EPM was reported in [HACK-75]:

Theorem 4.1.1

a. Λ(EPM) = Λ(PPM) and Λ_o(EPM) = Λ_o(PPM).

b. $H_\lambda(\Lambda_o(\text{EPM})) = \mathscr{RE}$ where $\mathscr{RE}$ denotes the set of recursively enumerable languages. $H_\lambda(\Lambda(\text{EPM})) = \text{PREF}(\mathscr{RE})$.

We have mentioned this result here because we shall refer to it during our study of the language families Λ(C-CPM) and Λ_o(C-CPM). Before we proceed to investigate the latter language families we shall first introduce in the following sections a few modifications to the EPM and to the PPM. The resulting models will facilitate our study of the relationship among the C-CPM, the EPM and the PPM.

Section 4.2 CLOSURE UNDER k-LIMITED ERASING OF Λ(EPM) and Λ_o(EPM)

The distinguishing feature of the EPN is the fact that given some arbitrary transition t_k and an inhibitor input place $p_j \in P_k^z$, t_k is enabled in some marking M^i only if $M^i(p_j) = 0$. On the other hand, if $p_r \in P_k^n$, then t_k is enabled in M^i only if $M^i(p_r) \geq I(p_r, t_k)$. The two separate conditions $M^i(p_r) = I(p_r, t_k)$ and $M^i(p_r) > I(p_r, t_k)$ cannot be distinguished by t_k. It is however useful in certain constructs to be able to model the following modified firing rule:

Let t_k be an arbitrary transition and let $p_r \in \mathscr{D}(I_k)$; we want t_k to be enabled in some marking M^i only if $M^i(p_r) = m$, for some given, fixed positive integer m. In particular, we want t_k to be disabled if $M^i(p_r) = m'$, for any $m' > m$.

These concepts are formally expressed below, in the definition of a modified EPM which we shall call EPM(I).

Definition 4.2.1

An Extended Petri Net (I) (EPN(I)) is a modified EPN EN = (T, P, I, O) such that:

1. T is a finite set of transitions.
2. P is a finite set of places.
3. $I: P \times T \to Z^o \cup \{\zeta\} \cup (\{\sigma\} \times Z^+)$ is a total, single-valued mapping called the input incidence function. ζ and σ are special symbols; $\zeta \notin Z^o$ and $\sigma \notin Z^o$.
4. $O: T \times P \to Z^o$ is the output incidence function.

Let $(p_j, t_k) \in P \times T$. If $I(p_j, t_k) = s$, where $s \in Z^+$, then p_j is a

normal input place of t_k. If $I(p_j, t_k) = \zeta$ then p_j is an inhibitor input place of t_k. Finally, if $I(p_j, t_k) = (\sigma, m)$ for some $m \in Z^+$, then p_j is connected to t_k by a special kind of arc, called a positive testing arc. Graphically, a positive testing arc is represented as a directed dotted arc. The integer m is marked adjacent to the corresponding positive testing arc (Figure 4.2.1). Let t_k be an arbitrary transition of some EPN(I) and let P_k^p be the set:

$$P_k^p = \{p_j \mid p_j \in P \text{ and } I(p_j, t_k) = (\sigma, m) \text{ for some } m \in Z^+\}$$

Since I is a single-valued mapping we shall have for each $t_k \in T$, $\mathscr{D}(I_k) = P_k^n \cup P_k^z \cup P_k^p$ where P_k^n, P_k^z and P_k^p are pairwise disjoint. Here, P_k^n and P_k^z have the same meaning as in the case of the EPM.

A marking of an EPN(I) is defined as a total, single-valued mapping from the set of places P into Z^o.

Definition 4.2.2

A transition $t_k \in T$ is enabled in the marking M^i if and only if:

1. $M^i(p_j) \geq I(p_j, t_k)$ for each place $p_j \in P_k^n$.
2. $M^i(p_j) = 0$ for each place $p_j \in P_k^z$.
3. $M^i(p_j) = m$ for each place $p_j \in P_k^p$, where $I(p_j, t_k) = (\sigma, m)$.

As in the case of the EPN, any enabled transition of an EPN(I) may be selected to fire.

Definition 4.2.3

If a transition t_k, enabled in the marking M^i, fires in M^i and $M^i[t_k > M^{i+1}$ then for any $p_j \in P$

$$M^{i+1}(p_j) = \begin{cases} M^i(p_j) - I(p_j, t_k) + O(t_k, p_j) & \text{if } p_j \in P - (P_k^z \cup P_k^p) \\ O(t_k, p_j) & \text{if } p_j \in (P_k^z \cup P_k^p) \end{cases}$$

We notice that there exists a difference between zero testing arcs and positive testing arcs. Thus, a zero testing arc merely checks whether the respective input place is empty in the current marking. On the other hand, a positive testing arc "senses" whether the input place to which it is connected contains in the current marking exactly m tokens, where m is a predefined positive integer associated with the arc and also removes those m tokens in case the transition fires.

Figure 4.2.1 exhibits a sample EPN(I). In the specified marking transition t_1 is enabled while t_2 is disabled.

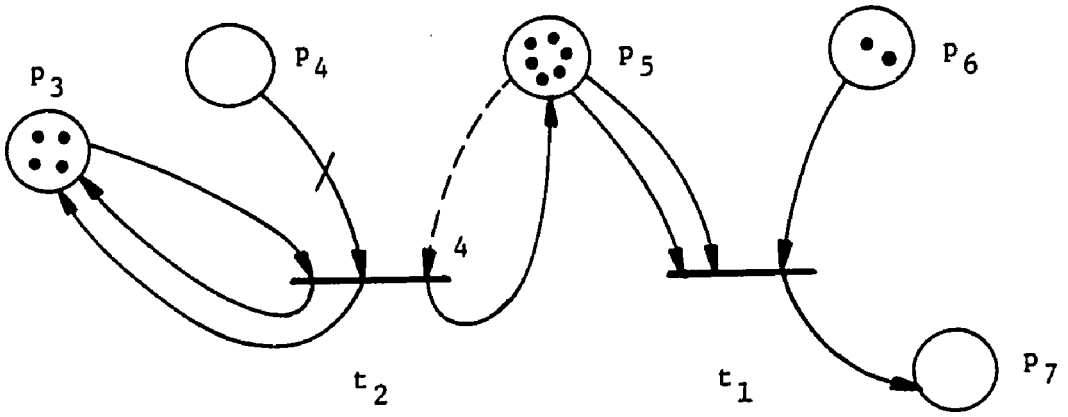


Figure 4.2.1
Sample EPN(I)

Definition 4.2.4

An Extended Petri Model (I) (EPM(I)) is a system $EN = (EN, M^0)$ where:

1. EN = (T, P, I, O) is an EPN(I).
2. M^0 is the initial marking of EN.

A Labelled EPM(I) is defined completely analogous to a Labelled EPM. Similarly, we can define the families of languages Λ(EPM(I)) and Λ_0(EPM(I)) the same way we did in the case of the EPM.

As will be seen later on, the introduction of the positive testing arcs facilitates certain proofs. Nevertheless this feature does not bring any additional power, as far as the representation capabilities of the EPM(I) are concerned, over the EPM or, equivalently, the PPM:

Theorem 4.2.1

Λ(EPM) = Λ(EPM(I)) and Λ_0(EPM) = Λ_0(EPM(I)).

Proof: Obviously, any EPM is an EPM(I) as well. Consequently, Λ(EPM) $\subseteq$ Λ(EPM(I)) and Λ_0(EPM) $\subseteq$ Λ_0(EPM(I)). Therefore, we have to show only that these inclusions are also true in opposite direction.

For this purpose it is advantageous to introduce the concept of a closed subnet. Thus, given a GPN N = (T, P, I, O) a closed subnet of

N is a GPN $N' = (T', P', I', O')$ such that:

1. $P' \subseteq P$.
2. $T' = \{t_k \mid t_k \in T$ and there exists a place $p_j \in P'$ such that $p_j \in \mathcal{D}(I_k) \cup \mathcal{D}(O_k)\}$.
3. I' is the restriction of I to $P' \times T'$.
4. O' is the restriction of O to $T' \times P'$.

The definition of a closed subnet of a GPN can immediately be extended to the case of an EPN(I). In what follows we shall describe the conversion of one-place ($|P'| = 1$) closed subnets of an arbitrary EPN(I) to λ-transition free, language equivalent EPM's. Next, the constructs will be generalized to suit entire EPN(I)'s.

Consider a one-place closed subnet EN' of an arbitrary EPN(I) EN, i.e. $EN' = (T', \{p_j\}, I', O')$. Let

$$T^n = \{t_k \mid t_k \in T' \text{ and } I'(p_j, t_k) = s, s \in Z^o\}$$
$$T^z = \{t_k \mid t_k \in T' \text{ and } I'(p_j, t_k) = \zeta\}$$
$$T^p = \{t_k \mid t_k \in T' \text{ and } I'(p_j, t_k) = (\sigma, m), m \in Z^+\}$$

Since the subnet contains only one place, we have $T' = T^n \cup T^z \cup T^p$ and T^n, T^z, T^p are pairwise disjoint. Let then:

$$n_1 = \max_{t_k \in T^n}\{s \mid s = I'(p_j, t_k)\},\ n_2 = \max_{t_k \in T^p}\{m \mid (\sigma, m) = I'(p_j, t_k)\}$$

and $K = \max\{n_1, n_2\}$.

We shall replace p_j by three places: p_{j1}, p_{j2} and p_{j3}. Our construct will implement the following encoding of any reachable marking $M(p_j)$ in terms of the markings $M(p_{j1})$ and $M(p_{j3})$:

$$M(p_j) = M(p_{j1}) + K \cdot M(p_{j3})$$

where $0 \leq M(p_{j1}) \leq K$ and $M(p_{j1}) = 0$ if and only if $M(p_j) = 0$. Also, $M(p_{j2}) = K - M(p_{j1})$.

For example, if the initial marking $M^o(p_j) = 0$ then $M^o(p_{j1}) = M^o(p_{j3}) = 0$ and $M^o(p_{j2}) = K$. Otherwise, we rewrite $M^o(p_j) = k_1 + k_2 \cdot K$ where $0 < k_1 \leq K$ and k_2 is the largest nonnegative integer which satisfies this equation. We emphasize that if $M^o(p_j)$ is a multiple of K, i.e. $M^o(p_j) = r \cdot K$ for some integer $r \geq 1$, then $k_1 = K$ and $k_2 = r - 1$. Then, we set $M^o(p_{j1}) = k_1$, $M^o(p_{j2}) = K - k_1$

and $M^0(p_{j3}) = k_2$.

Let us now modify the transitions of T' in order to implement the encoding rule discussed above. We have to consider following three cases:

A. $\underline{t_s \in T^n}$. In this case there are three possible interconnections between t_s and p_j:

A1. $\underline{I'(p_j, t_s) > 0 \text{ and } O'(t_s, p_j) > 0}$ (t_s self-loops on p_j). We shall replace t_s by K transitions $t_{s1}, \ldots, t_{sK}$. The input configuration of these transitions is defined as follows:

$$1 \leq i \leq K \begin{cases} I(p_{j1}, t_{si}) = i \\ I(p_{j2}, t_{si}) = K - i \\ I(p_{j3}, t_{si}) = \begin{cases} 1 & \text{if } 1 \leq i < I'(p_j, t_s) \\ 0 & \text{if } I'(p_j, t_s) \leq i \leq K \end{cases} \end{cases}$$

In order to determine the output configurations of these K transitions we shall first write the following K relations, for $1 \leq i \leq K$:

$$i + k_i \cdot K - I'(p_j, t_s) + O'(t_s, p_j) = k_{1i} + k_{2i} \cdot K$$

where

$$k_i = \begin{cases} 1 & \text{if } 1 \leq i < I'(p_j, t_s) \\ 0 & \text{if } I'(p_j, t_s) \leq i \leq K, \end{cases}$$

$0 < k_{1i} \leq K$ and k_{2i} is the largest nonnegative integer which satisfies the i-th equation.

Then:

$$1 \leq i \leq K \begin{cases} O(t_{si}, p_{j1}) = k_{1i} \\ O(t_{si}, p_{j2}) = K - k_{1i} \\ O(t_{si}, p_{j3}) = k_{2i} \end{cases}$$

It can easily be verified that if $M(p_j) = 0$ ($M(p_{j1}) = M(p_{j3}) = 0$, $M(p_{j2}) = K$) or $M(p_j) < I'(p_j, t_s)$ ($M(p_{j1}) = M(p_j)$, $M(p_{j2}) = K - M(p_j)$, $M(p_{j3}) = 0$) then none of the K transitions is enabled. If $M(p_j) \geq I'(p_j, t_s)$ then we can rewrite $M(p_j) = k_1 + k_2 \cdot K$ where $0 < k_1 \leq K$ and $k_2 \geq 0$. In this case $M(p_{j1}) = k_1$, $M(p_{j2}) = K - k_1$, $M(p_{j3}) = k_2$ and consequently exactly one

transition t_{si} is enabled, namely the one for which $I(p_{j1}, t_{si}) = k_i$ and $I(p_{j2}, t_{si}) = K - k_i$. The output configurations of the transitions t_{si}, $1 \leq i \leq K$, has been designed such that the firing of any t_{si} redistributes the tokens of p_{j1}, p_{j2} and p_{j3} according to the encoding rule discussed earlier. In particular, we notice that for any i, $1 \leq i \leq K$, we must have

$$i + k_i \cdot K - I'(p_j, t_s) + O'(t_s, p_j) > 0$$

and therefore the firing of t_{si} will never leave p_{j1} empty. (Remember, we do not want $M(p_{j1}) = 0$ and $M(p_{j3}) > 0$ simultaneously.)

Consequently, the transitions $t_{s1}, \ldots, t_{sK}$ simulate the firing of the original transition t_s while preserving the desired encoding of the marking of p_j in terms of the markings of p_{j1}, p_{j2} and p_{j3}.

A2. $\underline{I'(p_j, t_s) > 0 \text{ but } O'(t_s, p_j) = 0}$

Let $r = I'(p_j, t_s)$, $r \leq K$. We shall replace t_s by K + 1 transitions $t_{s1}, \ldots, t_{sr-1}, t'_{sr}, t''_{sr}, t_{sr+1}, \ldots, t_{sK}$. Let us first consider the K - 1 transitions t_{si}, for $1 \leq i \leq K$ and $i \neq I'(p_j, t_s)$. We set:

$$I(p_{j1}, t_{si}) = i$$
$$I(p_{j2}, t_{si}) = K - i$$
$$I(p_{j3}, t_{si}) = \begin{cases} 1 & \text{if } 1 \leq i < I'(p_j, t_s) \\ 0 & \text{if } I'(p_j, t_s) < i \leq K \end{cases}$$

In order to determine the output configurations of these transitions we shall form following K - 1 relations ($1 \leq i \leq K$ but $i \neq I'(p_j, t_s)$):

$$i + k_i \cdot K - I'(p_j, t_s) = k_{1i} + k_{2i} \cdot K$$

where

$$k_i = \begin{cases} 1 & \text{if } 1 \leq i < I'(p_j, t_s) \\ 0 & \text{if } I'(p_j, t_s) < i \leq K, \end{cases}$$

$0 < k_{1i} \leq K$ and k_{2i} is the largest nonnegative integer which satisfies the i-th relation. We notice that for the K - 1 values of i specified above

$$0 < i + k_i \cdot K - I'(p_j, t_s) < K$$

and therefore $0 < k_{1i} < K$ and $k_{2i} = 0$. Then

$$1 \leq i \leq K \qquad \begin{cases} O(t_{si}, p_{j1}) = k_{1i} \\ O(t_{si}, p_{j2}) = K - k_{1i} \\ O(t_{si}, p_{j3}) = 0 \end{cases}$$
$$i \neq I'(p_j, t_s)$$

The case where $i = r = I'(p_j, t_s)$ requires special treatment. We have introduced the two transitions t'_{sr} and t''_{sr}, where:

$$I(p_{j1}, t'_{sr}) = I(p_{j1}, t''_{sr}) = r = I'(p_j, t_s)$$
$$I(p_{j2}, t'_{sr}) = I(p_{j2}, t''_{sr}) = K - r$$
$$I(p_{j3}, t'_{sr}) = \zeta;\ I(p_{j3}, t''_{sr}) = 1$$

Regarding the output configurations, let:

$$O(t'_{sr}, p_{j1}) = 0;\ O(t''_{sr}, p_{j1}) = K$$
$$O(t'_{sr}, p_{j2}) = K;\ O(t''_{sr}, p_{j2}) = 0$$
$$O(t'_{sr}, p_{j3}) = O(t''_{sr}, p_{j3}) = 0$$

We notice that the transitions $t_{s1}, \ldots, t_{sr-1}, t_{sr+1}, \ldots, t_{sK}$ are defined and operate analogous to the transitions introduced in the case A1. On the other hand, it is easy to see that t'_{sr} and t''_{sr} cannot be simultaneously enabled. Thus, t'_{sr} is enabled if and only if the marking $M(p_j) = r = I'(p_j, t_s)$ or, equivalently, $M(p_{j1}) = r$, $M(p_{j2}) = K - r$ and $M(p_{j3}) = 0$. If t_s fires in M and $M\,[t_s > M'$ then $M'(p_j) = 0$. This fact was taken into consideration in the definition of the output configuration of t'_{sr}. Transition t''_{sr} is enabled if and only if $M(p_{j1}) = r$, $M(p_{j2}) = K - r$ and $M(p_{j3}) \geq 1$ or, equivalently, $M(p_j) = r + n \cdot K$ for some $n \geq 1$. If t_s fires in M and $M\,[t_s > M'$ then $M'(p_j) = n \cdot K$. In our encoding we must then have $M'(p_{j1}) = K$, $M'(p_{j2}) = 0$ and $M'(p_{j3}) = n - 1$. Again, this redistribution of tokens has been considered in the definition of the output configuration of t''_{sr}.

A3. $\underline{I'(p_j, t_s) = 0 \text{ but } O'(t_s, p_j) > 0}$. In this case the firing of t_s does not depend on the marking of p_j. Nevertheless, if t_s fires, tokens will be added to the marking of p_j which requires a redistribution of the tokens in p_{j1}, p_{j2} and p_{j3}. Consider following $K + 1$ relations, for $0 \leq i \leq K$:

$$i + O'(t_s, p_j) = k_{1i} + k_{2i} \cdot K$$

where $0 < k_{1i} \leq K$ and k_{2i} is the largest nonnegative integer which satisfies the i-th equation. We note that for any value of i, $0 \leq i \leq K$, $i + O'(t_s, p_j) > 0$ and thus $k_{1i} > 0$. Consequently, we shall replace t_s by K + 1 transitions whose only function is to redistribute tokens among p_{j1}, p_{j2} and p_{j3}. We shall have:

$$0 \leq i \leq K \begin{cases} I(p_{j1}, t_{si}) = i & ; O(t_{si}, p_{j1}) = k_{1i} \\ I(p_{j2}, t_{si}) = K - i & ; O(t_{si}, p_{j2}) = K - k_{1i} \\ I(p_{j3}, t_{si}) = 0 & ; O(t_{si}, p_{j3}) = k_{2i} \end{cases}$$

B. $t_s \in T^z$, i.e. p_j is an inhibitor input place of t_s. Consequently t_s can fire in some marking M, if and only if $M(p_j) = 0$ or, equivalently, $M(p_{j1}) = 0$, $M(p_{j2}) = K$ and $M(p_{j3}) = 0$. If $M\,[t_s > M'$ then $M'(p_j) = O'(t_s, p_j)$. In case $O'(t_s, p_j) = 0$ then $M'(p_j) = 0$ and we shall replace t_s by a single transition t_s' where:

$$I(p_{j1}, t_s') = \zeta \; ; \quad O(t_s', p_{j1}) = 0$$
$$I(p_{j2}, t_s') = K \; ; \quad O(t_s', p_{j2}) = K$$
$$I(p_{j3}, t_s') = \zeta \; ; \quad O(t_s', p_{j3}) = 0$$

Obviously, t_s' can fire if and only if $M(p_{j1}) = 0$, $M(p_{j2}) = K$ and $M(p_{j3}) = 0$. In fact, if $M(p_{j2}) = K$ we must have $M(p_{j1}) = M(p_{j3}) = 0$ and thus it is not really necessary to connect p_{j1} and p_{j3} to t_s' by zero testing arcs.

On the other hand, if $O'(t_s, p_j) > 0$ then we can rewrite $O'(t_s, p_j)$ as $O'(t_s, p_j) = k_1 + k_2 \cdot K$ where $0 < k_1 \leq K$ and k_2 is the largest nonnegative integer which satisfies this relation. In this case, we shall replace t_s by a single transition t_s'' which has the same input configuration as t_s' but where:

$$O(t_s'', p_{j1}) = k_1$$
$$O(t_s'', p_{j2}) = K - k_1$$
$$O(t_s'', p_{j3}) = k_2$$

C. $t_s \in T^p$, i.e. $I'(p_j, t_s) = (\sigma, m)$ for some $m \in Z^+$. In this case t_s can fire in some marking M if and only if $M(p_j) = m$ or, equivalently, $M(p_{j1}) = m$, $M(p_{j2}) = K - m$ and $M(p_{j3}) = 0$ (remember, $m \leq K$). If t_s fires in M and $M\,[t_s > M'$ then $M'(p_j) = O'(t_s, p_j)$. In case $O'(t_s, p_j) = 0$ then $M'(p_j) = 0$ or, equivalently, $M'(p_{j1}) = 0$, $M'(p_{j2}) = K$,

$M'(p_{j3}) = 0$. Consequently, we shall replace t_s by a single transition t_s' where:

$$I(p_{j1}, t_s') = m \quad ; \quad O(t_s', p_{j1}) = 0$$
$$I(p_{j2}, t_s') = K - m \; ; \quad O(t_s', p_{j2}) = K$$
$$I(p_{j3}, t_s') = \zeta \quad ; \quad O(t_s', p_{j3}) = 0$$

Otherwise, if $O'(t_s, p_j) > 0$ then we can rewrite as in the previous case $O'(t_s, p_j) = k_1 + k_2 \cdot K$ where $0 < k_1 \leq K$ and $k_2 \geq 0$. Then, we shall replace t_s by a single transition t_s'' having the same input configuration as t_s' and:

$$O(t_s'', p_{j1}) = k_1$$
$$O(t_s'', p_{j2}) = K - k_1$$
$$O(t_s'', p_{j3}) = k_2$$

It is obvious that any arbitrary one-place closed subnet of an EPM(I) can be converted into an EPM with three places and no λ-transitions. Figure 4.2.2 exhibits a sample EPM(I) while Figure 4.2.3 displays the two one-place closed subnets (and the corresponding initial markings) which can be formed from the given EPM(I). Applying the transformation procedure to these two closed subnets we obtain the EPM's displayed in Figure 4.2.4 a) and b), respectively.

The conversion procedure discussed above can easily be extended to transform EPM(I)'s with more than one place. Thus, given some arbitrary EPM(I), the conversion procedure is separately applied to all one-place closed subnets which can be formed, or at least to all such subnets which contain positive testing arcs. Given initial and/or final markings are also redistributed according to the encoding rule specified earlier. Next, the resulting EPM's are recombined into a composite EPM by a procedure which can be regarded as the "inverse" of the procedure by means of which the original EPM(I) was divided into one-place closed subnets. Thus, distinct copies introduced for the same transition of the original net (relative to distinct closed subnets) are replaced by a unique transition having the same input and output connections as the eliminated transitions. If a transition t_s is contained in m one-place closed subnets of the original EPM(I) then we shall form all possible distinct combinations of m replacement transitions of t_s, each combination containing exactly one transition

from each of the m distinct EPM's obtained from the transformation of the subnets in which t_s participates. For example, in Figure 4.2.4 we would have to combine each of the transitions t_{21}, t'_{22} and t''_{22} with the transition t'_2. The pair t''_{22}, t'_2 would be replaced by a single transition, let us denote it by t''_2, where $I(p_{61}, t''_2) = 2$, $I(p_{62}, t''_2) = 0$, $I(p_{63}, t''_2) = 1$, $I(p_{71}, t''_2) = 2$, $I(p_{72}, t''_2) = 0$, $I(p_{73}, t''_2) = \zeta$ and $O(t''_2, p_{61}) = 2$, $O(t''_2, p_{62}) = O(t''_2, p_{63}) = 0$, $O(t''_2, p_{71}) = O(t''_2, p_{73}) = 0$, $O(t''_2, p_{72}) = 2$. The pairs t_{21}, t'_2 and t'_{22}, t'_2 are replaced in a similar manner. The complete construct is given in Figure 4.2.5.

Finally, each transition introduced as a replacement transition for some transition of the original EPM(I) is assigned the same label as the respective original transition. The theorem follows immediately.

We shall proceed now to prove the main result of this section, namely the closure under k-limited erasing of the language families Λ(EPM) and Λ_o(EPM). Let Σ be a nonempty, finite alphabet and suppose $c \notin \Sigma$. Given a language $W \subseteq (\Sigma \cup \{c\})^*$ and a homomorphism h defined on $(\Sigma \cup \{c\})^*$, h is said to be a <u>k-limited erasing on W</u> if following conditions are satisfied:

1. $h(c) = \lambda$ and $h(a) = a$ for all $a \in \Sigma$.

2. There exists a nonnegative integer k such that for all strings $w \in W$, h does not erase more than k consecutive symbols of w.

There is no loss in generality in this definition. Thus, consider a language $W \subseteq \Sigma^*$ and let h be a homomorphism from Σ^* into Σ_1^*, where $\Sigma_1 \subseteq \Sigma$, which either erases or maps into itself each symbol of Σ but does not erase more than k consecutive symbols from any string in W, for some nonnegative integer k. Let c be a new symbol, $c \notin \Sigma$, and let h_1 be a homomorphism on Σ^* such that $h_1(a) = c$ if $h(a) = \lambda$ and $h_1(a) = h(a)$ otherwise. Next, let h_2 be the homomorphism on $(\Sigma_1 \cup \{c\})^*$ defined by $h_2(c) = \lambda$ and $h_2(a) = a$ for all $a \in \Sigma_1$. Clearly, h_2 is a k-limited erasing on $h_1(W)$ in the sense of the previous definition and $h_2(h_1(W)) = h(W)$.

Let us consider an arbitrary language W in Λ(EPM) or Λ_o(EPM). We shall assume that $W \subseteq (\Sigma\{\lambda, c, \ldots, c^k\})^*$ where Σ is a nonempty finite alphabet, $c \notin \Sigma$ and k is a nonnegative integer. Let h be a homomorphism on $(\Sigma \cup \{c\})^*$ defined by $h(c) = \lambda$ and $h(a) = a$ for all

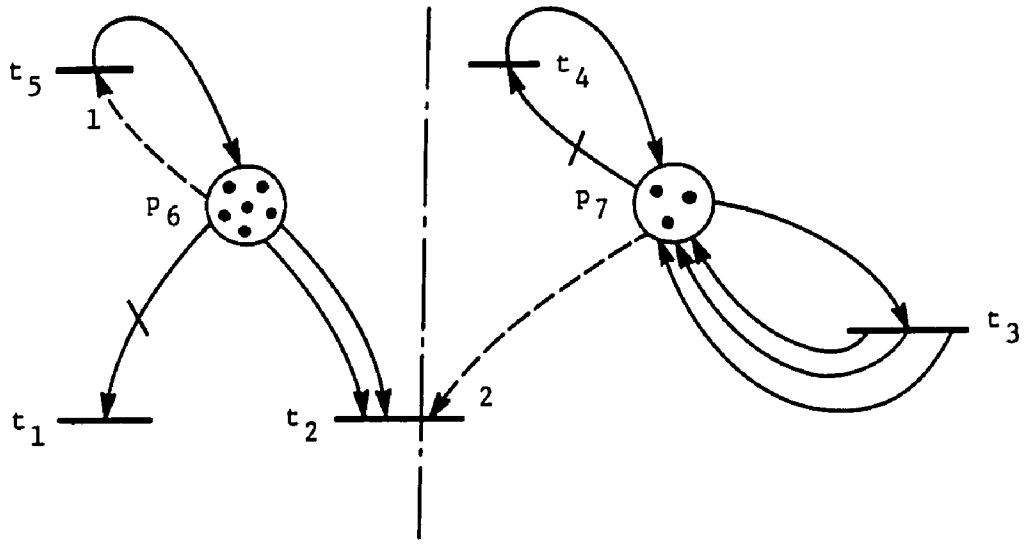
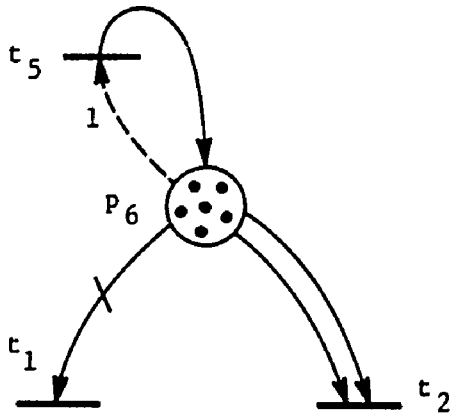


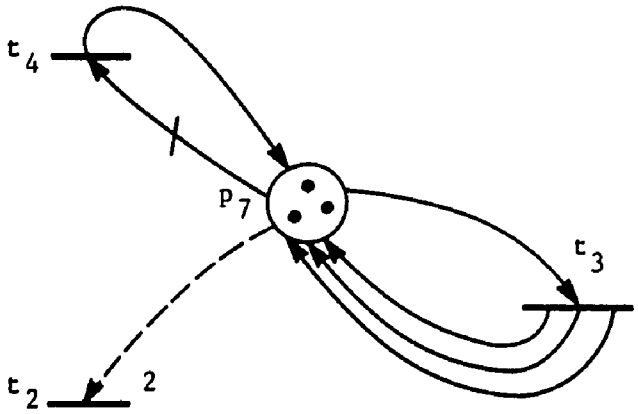
Figure 4.2.2
Sample EPM(I)



$n_1 = 2$

$n_2 = 1$

$K = 2$

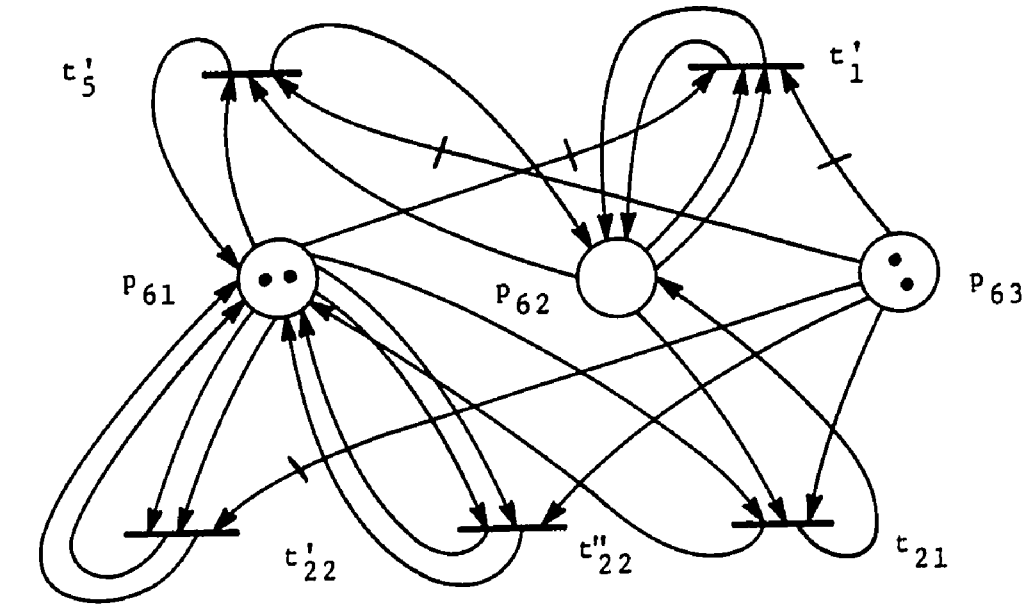


$n_1 = 1$

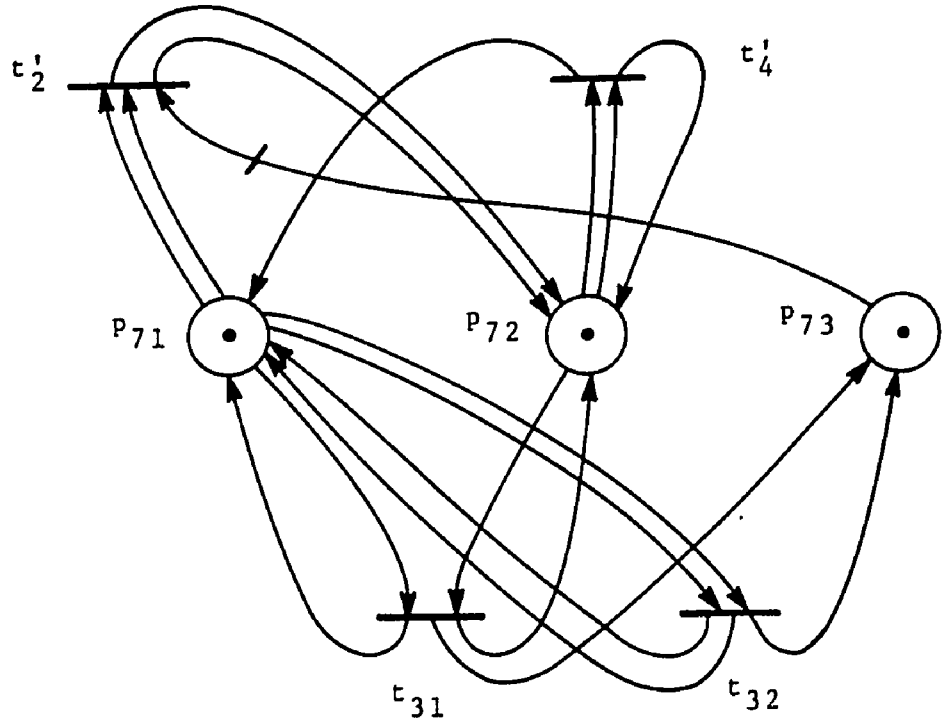
$n_2 = 2$

$K = 2$

Figure 4.2.3
One-Place Closed Subnets of the EPM(I) of Figure 4.2.2



a)



b)

Figure 4.2.4

EPM's Obtained from the One-Place Closed Subnets of Figure 4.2.3

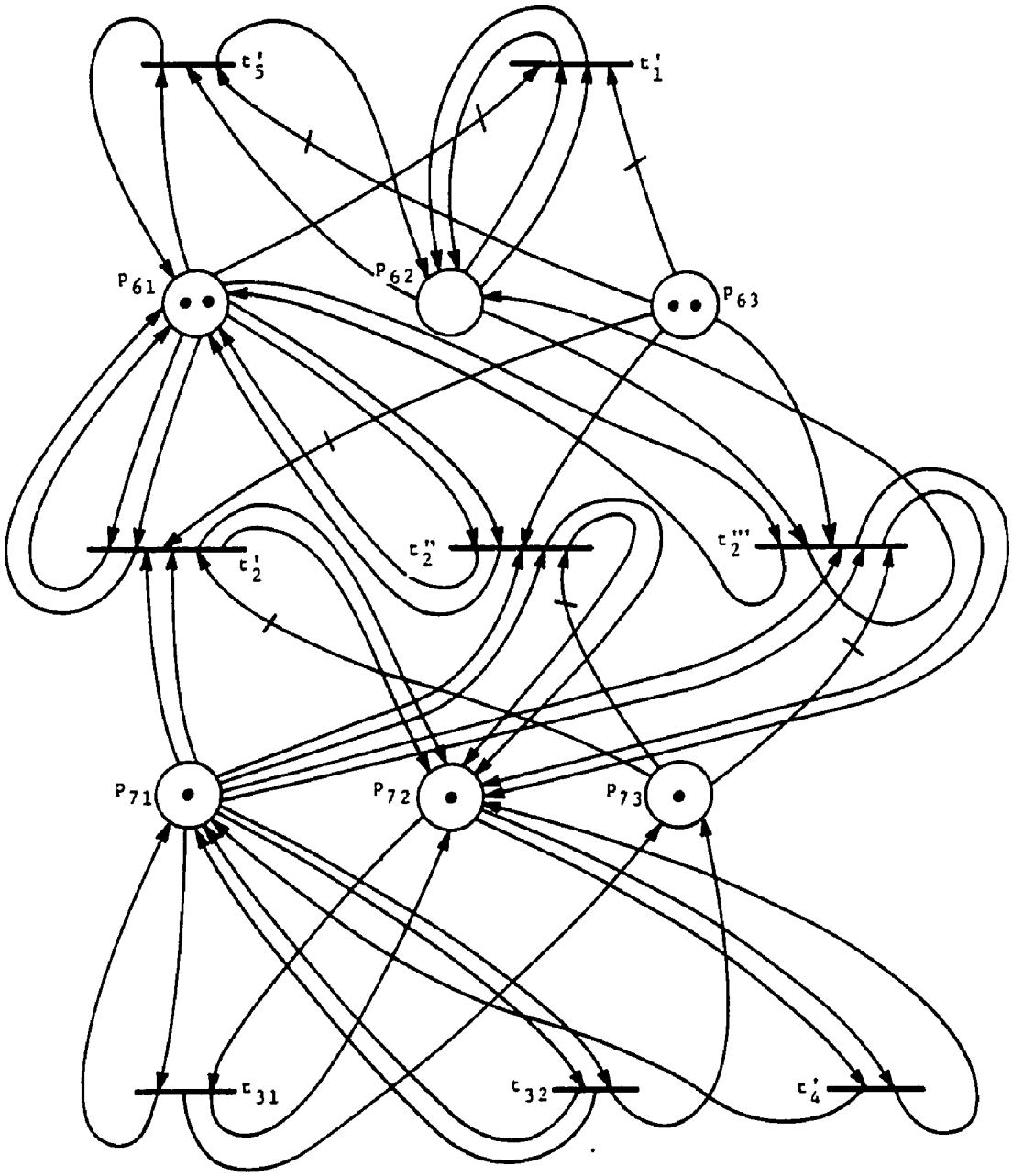


Figure 4.2.5

EPM Obtained from the Sample EPM(I) of Figure 4.2.2

$a \in \Sigma$. Otherwise stated, we assume that there are no strings in W which contain more than k consecutive symbols c; such strings would not be in the domain of the k-limited erasing h. Let us define the finite set $W' = \{c^i a c^j \mid a \in \Sigma \text{ and } 0 \leq i, j \leq k\}$. A string of W' may be generated by several distinct firing sequences of the Labelled EPM that generates W. Since for any Labelled EPM, the labelling function L is a λ-free renaming, if γ is a firing sequence such that $L(\gamma) \in W'$ then $|\gamma| = |L(\gamma)|$ and therefore for each such firing sequence γ, $0 < |\gamma| \leq 2 \cdot k + 1$. Thus, there are only a finite number of firing sequences which can produce label sequences in W'.

Theorem 4.2.2

The language families Λ(EPM) and Λ_o(EPM) are closed under k-limited erasing.

Proof: The general idea behind the construct which we shall employ is to modify the given EPM such that during the execution of the resulting Labelled EPM(I), each possible firing sequence γ of the original net for which $L(\gamma) \in W'$ is substituted by the firing of a single transition τ, where:

1. τ is enabled in some marking M^i if and only if the entire firing sequence γ could have been fired from M^i.
2. $M^i[\tau > M^{i+1}$ if and only if $M^i[\gamma > M^{i+1}$.
3. $L(\tau) = h(L(\gamma))$ (We note that for all strings $w \in W'$, $h(w) \in \Sigma$).

Since the set W' contains all the strings of the form $c^i a c^j$, $0 \leq i, j \leq k$, the construct applies to both language families Λ(EPM) and Λ_o(EPM).

We shall first investigate the conditions on the marking and the structure of a net under which a finite-length firing sequence γ is firable.

Let $EN_{\Sigma'} = (EN, \Sigma', L)$ be the Labelled EPM which generates the language W, where $\Sigma' = \Sigma \cup \{c\}$, $EN = (EN, M^o)$ and $EN = (T, P, I, O)$. Suppose $\gamma_j = t_{j1} \ldots t_{jm}$ is a finite length firing sequence in T^+. Let us assume that p_r is some place of the net which is not an inhibitor input place for any transition in γ_j. We shall denote by $Min(p_r, \gamma_j)$ the minimum marking required in p_r in order to fire the

entire firing sequence γ_j. $Min(p_r, \gamma_j)$ can be determined by the following straightforward algorithm which simply simulates the effect of the firing sequence γ_j on the marking of p_r:

```
initial  Min(p_r, γ_j), M(p_r) = I(p_r, t_j1)
while  1 ≤ k ≤ m  do
  if  I(p_r, t_jk) = 0
    then  M(p_r) = M(p_r) + O(t_jk, p_r)
    else if  M(p_r) ≥ I(p_r, t_jk)
           then  M(p_r) = M(p_r) - I(p_r, t_jk) + O(t_jk, p_r)
           else  Min(p_r, γ_j) = Min(p_r, γ_j) + I(p_r, t_jk) - M(p_r)
                 M(p_r) = O(t_jk, p_r)
end
```

$M(p_r)$ denotes the "current" marking of p_r along the firing sequence γ_j. We shall define $Min(p_r, \lambda) = 0$.

Thus, γ_j can be fired from some marking M only if $M(p_r) \geq Min(p_r, \gamma_j)$.

Let us now suppose that $\gamma_j = t_{j1} \cdots t_{jk-1} t_{jk} t_{jk+1} \cdots t_{js-1} t_{js} t_{js+1} \cdots t_{jm}$ and let us denote $\delta'_j = t_{j1} \cdots t_{jk-1}$, $\delta''_j = t_{jk+1} \cdots t_{js-1}$. Also, let us now assume that p_r is an inhibitor input place of t_{jk} but p_r is not an inhibitor input place for any transition in δ'_j. Then, γ_j is a valid firing sequence of EN (i.e., γ_j can be fired from some marking M) only if:

$$Min(p_r, \delta'_j) = \sum_{n=1}^{k-1} [I(p_r, t_{jn}) - O(t_{jn}, p_r)]$$

If the above condition is satisfied then γ_j can only be fired from those markings M for which $M(p_r) = Min(p_r, \delta'_j)$. In particular, if $\delta'_j = \lambda$ or if $Min(p_r, \delta'_j) = 0$ then γ_j can only be fired from those markings M for which $M(p_r) = 0$.

If

$$M(p_r) = Min(p_r, \delta'_j) = \sum_{n=1}^{k-1} [I(p_r, t_{jn}) - O(t_{jn}, p_r)]$$

then the marking $M''(p_r)$ obtained at the end of the firing sequence $\delta'_j t_{jk} \delta''_j$ is:

$$M'(p_r) = O(t_{jk}, p_r) - \sum_{n = k + 1}^{s - 1} [I(p_r, t_{jn}) - O(t_{jn}, p_r)]$$

We can see that the firability of the transition t_{js} from the point of view of the input place p_r depends only on the particular connections to p_r of the transitions in the subsequence $t_{jk}\delta_j''$ and not on the marking M from which γ_j is fired.

With the above preparation we are now ready to modify the original Labelled EPM. First, we eliminate all transitions labelled c and the corresponding input and/or output arcs. Further, for every valid firing sequence γ_j of EN such that $L(\gamma_j) \in W'$ we introduce a transition τ_j. The input and the output connections of the new transition τ_j to an arbitrary place p_r are determined as follows. We first consider the input arcs. There are two cases to be considered:

1. p_r is not an inhibitor input place for any transition in γ_j. In this case $I'(p_r, \tau_j) = \text{Min}(p_r, \gamma_j)$ where I' denotes the input incidence function of the modified net.

2. p_r is an inhibitor input place for at least one transition in γ_j. Suppose t_{jk} is the first transition in the sequence γ_j such that $p_r \in P^z_{jk}$, i.e. p_r is not an inhibitor input place for any transition in the prefix firing sequence $\delta_j' = t_{j1}\cdots t_{jk-1}$. Then

$$I'(p_r, \tau_j) = \begin{cases} \zeta & \text{if } \text{Min}(p_r, \delta_j') = 0 \\ & \text{(zero testing arc)} \\ (\sigma, m) & \text{if } \text{Min}(p_r, \delta_j') = m > 0 \\ & \text{(positive testing arc)} \end{cases}$$

Let us now consider the output arcs of the transition τ_j:

$$O'(\tau_j, p_r) = m - \sum_{t_{jk} \in \gamma_j} [I(p_r, t_{jk}) - O(t_{jk}, p_r)]$$

where

$$m = \begin{cases} \text{Min}(p_r, \gamma_j) & \text{if } I'(p_r, \tau_j) \geq 0 \\ 0 & \text{if } I'(p_r, \tau_j) = \zeta \\ m' & \text{if } I'(p_r, \tau_j) = (\sigma, m') \\ & \text{for some integer } m' > 0. \end{cases}$$

For the purpose of this summation $I(p_r, t_{jk}) = \zeta$ has been replaced by 0.

Our construct preserves the places of the original Labelled EPM as well as the transitions whose label is different from the erased symbol c.

Let us now show that the resulting Labelled EPM(I) generates exactly the language h(W). From the above discussion it should be clear that for any two markings M and M', $M[\gamma_j > M'$ in the original net if and only if $M[\tau_j > M'$ in the modified net. Let T' denote the set of transitions of the original Labelled EPM whose label is different from c; T' contains the transitions common to the original net and to the EPM(I) resulting from our construct.

Consider an arbitrary firing sequence of the original EPM:

$$\gamma = \delta_1\gamma_1\delta_2\ldots\delta_k\gamma_k\delta_{k+1}\ldots\delta_m\gamma_m\delta_{m+1}$$

where $\delta_i \in T'^*$, for $1 \leq i \leq m+1$, and γ_i is a firing sequence such that $L(\gamma_i) \in W'$, for $1 \leq i \leq m$. Let us suppose that:

$$M^0[\delta_1 > M^1[\gamma_1 > M^{1'}[\delta_2 > M^2 \ldots M^{k-1'}[\delta_k > M^k[\gamma_k > M^{k'}[\delta_{k+1} > M^{k+1} \ldots M^{m-1'}[\delta_m > M^m[\gamma_m > M^{m'}[\delta_{m+1} > M^{m+1}$$

Since $\delta_i \in T'^*$, for $1 \leq i \leq m+1$, if $M[\delta_i > M'$ in the original net then $M[\delta_i > M'$ in the modified net as well. Consequently, there exists the firing sequence in the modified net:

$$\gamma' = \delta_1\tau_1\delta_2\ldots\delta_k\tau_k\delta_{k+1}\ldots\delta_m\tau_m\delta_{m+1}$$

such that

$$M^0[\delta_1 > M^1[\tau_1 > M^{1'}[\delta_2 > M^2 \ldots M^{k-1'}[\delta_k > M^k[\tau_k > M^{k'}[\delta_{k+1} > M^{k+1} \ldots M^{m-1'}[\delta_m > M^m[\tau_m > M^{m'}[\delta_{m+1} > M^{m+1}$$

The firing sequence γ' generates the string:

$$L(\delta_1)h(L(\gamma_1))L(\delta_2)\ldots L(\delta_k)h(L(\gamma_k))L(\delta_{k+1}) \ldots \ldots L(\delta_m)h(L(\gamma_m))L(\delta_{m+1}),$$

i.e. $h(L(\gamma))$.

For the sake of clarity we emphasize that given such a firing sequence γ there may exist more than one distinct firing sequences γ' which generate the same string $h(L(\gamma))$. For example, consider the following subsequence of γ and the corresponding label string:

$$\ldots\ t_a t_{c1} t_{c2} t_{c3} t_b\ \ldots$$
$$\ldots\ a\ c\ \ c\ \ c\ \ b\ \ \ldots$$

Obviously, if $t_a t_{c1} t_{c2} t_{c3} t_b$ is a valid firing sequence of the original net then so are the firing sequences given in the leftmost column of the following table:

$t_a t_{c1}$	τ_1	a
$t_a t_{c1} t_{c2}$	τ_2	a
$t_a t_{c1} t_{c2} t_{c3}$	τ_3	a
$t_{c1} t_{c2} t_{c3} t_b$	τ_4	b
$t_{c2} t_{c3} t_b$	τ_5	b
$t_{c3} t_b$	τ_6	b

All these firing sequences generate a string in W'. The middle column of the table indicates the corresponding replacement transitions τ_i while the rightmost column indicates the label of each of these replacement transitions. Then, the following firing sequences of the modified net simulate the given firing sequence ... $t_a t_{c1} t_{c2} t_{c3} t_b$ ... :

... $\tau_1 \tau_5$ ...
... $\tau_2 \tau_6$ ...
... $\tau_3 t_b$ ...
... $t_a \tau_4$ ...

Each of these firing sequences generates the label sequence ... ab ... . Note that the firing sequence ... $\tau_1 \tau_6$ ... for example, can be executed in the modified net only if the firing sequence ... $t_a t_{c1} t_{c3} t_b$ ... can be executed in the original net.

Conversely, let γ' be some arbitrary firing sequence of the modified net:

$$\gamma' = \delta_1 \tau_1 \delta_2 \cdots \delta_k \tau_k \delta_{k+1} \cdots \delta_m \tau_m \delta_{m+1}$$

where $\delta_i \in T'^*$, for $1 \leq i \leq m+1$, and τ_i, for $1 \leq i \leq m$, denotes the occurrence of a replacement transition introduced by our construct.

Then, there exists the firing sequence of the original Labelled EPM:

$$\gamma = \delta_1 \gamma_1 \delta_2 \cdots \delta_k \gamma_k \delta_{k+1} \cdots \delta_m \gamma_m \delta_{m+1}$$

such that $M^o [\gamma > M^{m+1}$ if $M^o [\gamma' > M^{m+1}$ and $h(L(\gamma))$ is the string generated by γ'.

Thus, the Labelled EPM(I) resulting from our construct can gener-

ate all the label sequences in h(W) and no other label sequences. By virtue of Theorem 4.2.1 the theorem follows.

□

Section 4.3 THE PRIORITY PETRI MODEL (I)

In Section 4.1 we have defined the Priority Petri Model. We have seen that given some Priority Petri Net PN = (N, Pr, Y) and a marking M^i of PN, a transition t_k of N is firable in M^i if and only if t_k is enabled in M^i and no transition with higher priority than t_k is also enabled in M^i. In the present section we shall modify this rule for the selection of a transition to be fired. The resulting model, which we shall call Priority Petri Model (I), will be useful in our study of the relationship between the PPM and the C-CPM.

Let PN = (N, Pr, Y) be a PPN in some marking M^i, where N = (T, P, I, O) is a GPN, Pr = (Q, $\underline{\alpha}$) is a finite partially ordered set and Y is a total, single-valued mapping from T into Q. Let us define for each place $p_j \in P$ the set:

$$T(p_j) = \{ t_k \mid t_k \in T \text{ and } I(p_j, t_k) > 0 \}$$

Definition 4.3.1

There exists a conflict at the place p_j in the marking M^i if and only if

$$\sum_{t_k \in E^i \cap T(p_j)} I(p_j, t_k) > M^i(p_j)$$

Thus, two transitions $t_k \in T$ and $t_s \in T$ are said to conflict (denoted by $t_k \circ t_s$) in the marking M^i if:

1. $t_k \in E^i$ and $t_s \in E^i$.
2. There exists at least one place $p_j \in I_k \cap I_s$ such that there is a conflict at p_j in M^i.

We can extend the conflict relation in the case of the PPN the same way we did in Section 2.3. Thus, let CONF be the relation on E^i such that for any $t_k \in E^i$ and $t_s \in E^i$, $(t_k, t_s) \in$ CONF if there exists a sequence of transitions $t_k = t_{r1}, \ldots, t_{rn} = t_s$ such that $t_{rj} \circ t_{rj+1}$ for each j, $1 \leq j < n$. By convention we assume that t_k CONF t_k for all $t_k \in E^i$. Using the same argument as given in Lemma 2.3.1 one can easily show (proof omitted) that CONF is an equivalence relation on E^i

in the case of the PPN as well. We shall continue to call an equivalence class of CONF a conflict cluster. The set of all conflict clusters in some marking M^i will be denoted by CF^i. Obviously, CF^i is a partition of E^i.

Let us suppose that there are m conflict clusters in the marking M^i, i.e. $CF^i = \{ CT^i_1, \ldots , CT^i_m\}$. For each conflict cluster CT^i_r, a conflict subcluster of CT^i_r is a maximal subset SCT^i_{rk} of CT^i_r such that all transitions in SCT^i_{rk} are pairwise in conflict in M^i.

Definition 4.3.2

A transition $t_k \in T$ is said to cover another transition $t_s \in T$ in the marking M^i (denoted by t_k/t_s) if:

1. $t_k \in E^i$ and $t_s \in E^i$.
2. $Y(t_s) \alpha Y(t_k)$.

The set of majorants of a conflict subcluster SCT^i_{rk} is defined in the case of the PPN analogous to Definition 2.3.6 and is denoted by $maj\{SCT^i_{rk}\}$. Each transition of $maj\{SCT^i_{rk}\}$ is called a majorant of the conflict subcluster SCT^i_{rk}. Due to the fact that the priorities of the transitions contained in a conflict subcluster form a nonempty subset of the set Q (where $Pr = (Q, \underline{\alpha})$) there may exist more than one majorants of a conflict subcluster.

We notice that the set SCF^i_r of all conflict subclusters of some conflict cluster CT^i_r is not necessarily a partition of CT^i_r. Also, two transitions t_k and t_s may belong to CT^i_r and still have $I_k \cap I_s = \phi$. On the other hand, if t_k and t_s are in the same conflict subcluster of CT^i_r then $I_k \cap I_s \neq \phi$ since $t_k \circ t_s$. Given some marking M^i, let us define for each transition $t_s \in T$ the set:

$$Y^i_s = \{SCT^i_{rk} \mid t_s \in SCT^i_{rk}\}$$

According to the previous remarks, we must have for any transitions $t_m \in T$ and $t_s \in T$:

$$t_m \in \bigcup_{SCT^i_{rk} \in Y^i_s} SCT^i_{rk} \qquad \text{only if } I_m \cap I_s \neq \phi$$

Consequently, in any marking M^i, a transition t_s can participate in at most m_s distinct conflict subclusters, where $m_s = |\mathcal{D}(I_s)|$, a finite

nonnegative integer, fixed by the structure of the respective GPN. Therefore we must have:

$$\bigcup_{SCT^i_{rk} \in Y^i_s} SCT^i_{rk} \subseteq E^i \cap \left[\bigcup_{p_j \in \mathcal{D}(I_s)} T(p_j) \right]$$

Using the concepts of conflict cluster, conflict subcluster and majorant of a conflict subcluster one can immediately translate the execution rule given for the CPM in Section 2.3 for the case of the PPN. Accordingly, the new definition of a firable transition is:

Definition 4.3.3

A transition t_k is firable in some marking M^i if and only if t_k is a majorant of all the conflict subclusters in which it participates in M^i.

With these introductory definitions we can now define a Priority Petri Net (I) (PPN(I)) as a PPN PN = (N, Pr, Y) such that in each reachable marking of PN the transition to be fired is selected according to Definition 4.3.3. The effect produced by the firing of a transition of a PPN(I) is as described in Definition 4.1.5.

Definition 4.3.4

A Priority Petri Model (I) (PPM(I)) is a system $\mathcal{PN}$ = (PN, M^o) where:

1. PN = (N, Pr, Y) is a PPN(I).
2. M^o is the initial marking of PN.

A Labelled PPM(I) is defined analogous to Definition 4.1.3. Similarly, the language families Λ(PPM(I)) and Λ_o(PPM(I)) are defined completely analogous to Λ(PPM) and Λ_o(PPM).

In the remainder of this section we shall show that these modified execution rules do not alter the representation capabilities of the Priority Petri Model.

Lemma 4.3.1

Λ(PPM(I)) $\subseteq$ Λ(EPM(I)) and

Λ_o(PPM(I)) $\subseteq$ Λ_o(EPM(I)).

Proof: We begin by introducing a few notations. Let PN = (N, Pr, Y)

be a PPN(I) where $N = (T, P, I, O)$ is the underlying GPN. Suppose PN is in some marking M^i. Rephrasing Definition 4.3.3, a transition $t_s \in T$ is firable in the marking M^i if and only if $Y(t_s)$ is a maximal element of the set:

$$\{Y(t_m) \mid t_m \in SCT^i_{rk}, \text{ for some conflict subcluster } SCT^i_{rk} \in Y^i_s\}$$

Let us define for each transition $t_s \in T$ the sets:

$$T(t_s) = \bigcup_{p_j \in \mathcal{D}(I_s)} T(p_j)$$

$$H(t_s) = \{t_m \mid t_m \in T(t_s) \text{ and } Y(t_s)\ \alpha\ Y(t_m)\}$$

Then, t_s can be selected to fire in the marking M^i if and only if $t_s \in E^i$ and

$$H(t_s) \cap \{t_m \mid t_s \circ t_m \text{ in } M^i\} = \phi \qquad (*)$$

Obviously, if $H(t_s) = \phi$ the relation above is satisfied and t_s can be selected to fire in the marking M^i if it is enabled in M^i, regardless of the transitions with which it may conflict in M^i.

Let us now assume that $H(t_s) \neq \phi$ and suppose $M^i(p_j) < M_j$ for some $p_j \in \mathcal{D}(I_s)$, where

$$M_j = \sum_{t_m \in T(p_j)} I(p_j, t_m)$$

In this case there may exist a conflict at p_j in the marking M^i, depending on the status of the transitions in $T(p_j)$. Consequently, the firability of t_s may depend on the priorities of the transitions with which it conflicts. If $t_s \circ t_m$ in M^i and p_j is a place in $I_s \cap I_m$ such that there is a conflict at p_j in M^i, then we shall say that t_s and t_m "conflict at p_j". We shall also say that t_s, and for that matter t_m, "participates in a conflict at p_j". Let us denote for each $t_s \in T$:

$$\pi'(t_s) = \bigcup_{t_m \in T(t_s)} [\mathcal{D}(I_m) - \mathcal{D}(I_s)]$$

$$\pi(t_s) = \mathcal{D}(I_s) \cup \pi'(t_s).$$

In order to simplify notations let us suppose that $\pi(t_s) = \{p_{j1}, \cdots, p_{jq}, p_{jq+1}, \cdots, p_{jn}\}$ such that $p_{jk} \in \mathcal{D}(I_s)$ for $1 \leq k \leq q$ and $p_{jk} \in \pi'(t_s)$ for $q+1 \leq k \leq n$. Let us also define for

each $t_s \in T$ and all $p_{jk} \in \pi'(t_s)$:

$$m^s_{jk} = \max\{I(p_{jk}, t_m) \mid t_m \in T(t_s) \cap T(p_{jk})\}$$

At this point it is advantageous to view the ranges of the markings of a PPN(I) as vectors of nonnegative integers. Thus, the range of the restriction of any marking M^i of PN to $\pi(t_s)$ can be represented as a vector with n coordinates, where the k-th coordinate equals $M^i(p_{jk})$, $1 \leq k \leq n$.

In this spirit let us form the finite set of vectors:

$$M(t_s) = \{(\beta_1, \ldots, \beta_q, \beta_{q+1}, \ldots, \beta_n) \mid I(p_{jk}, t_s) \leq \beta_k \leq M_{jk} \text{ for } 1 \leq k \leq q \text{ and } 0 \leq \beta_k \leq m^s_{jk} \text{ for } q+1 \leq k \leq n\} -$$

$$- \{(M_{j1}, \ldots, M_{jq}, \beta_{q+1}, \ldots, \beta_n) \mid 0 \leq \beta_k \leq m^s_{jk} \text{ for } q+1 \leq k \leq n\}$$

From the analysis of the vectors in the set $M(t_s)$ we shall determine the set of markings of PN in which the transition t_s is firable. First, we explain the particular selection of the vectors in $M(t_s)$.

We notice that for any transition $t_s \in T$, the set $H(t_s)$ is fixed by the definition of the respective PPN(I). On the other hand, in any marking M^i

$$\{t_m \mid t_s \circ t_m \text{ in } M^i\} \subseteq E^i \cap T(t_s)$$

and moreover, t_s can participate at the most in conflicts at places $p_j \in \mathscr{D}(I_s)$. Therefore, in order to determine whether t_s may be selected to fire in the marking M^i, it is sufficient to examine the restriction of M^i to $\pi(t_s)$.

Consider a transition $t_m \in T(t_s)$, $t_m \neq t_s$, and let $p_{jk} \in \pi'(t_s) \cap \mathscr{D}(I_m)$, i.e. $p_{jk} \in \mathscr{D}(I_m)$ but $p_{jk} \notin \mathscr{D}(I_s)$. Obviously, t_s and t_m cannot conflict at p_{jk} but $t_s \circ t_m$ in some marking M^i only if $M^i(p_{jk}) \geq I(p_{jk}, t_m)$. Suppose now that M^i is a marking of PN such that the restriction of M^i to $\pi(t_s)$ is a vector in $M(t_s)$ and $M^i(p_{jk}) = m^s_{jk}$. Let M' be another marking of PN such that for any $p_{jr} \in \pi(t_s)$, $r \neq k$, $M'(p_{jr}) = M^i(p_{jr})$ and $M'(p_{jk}) > M^i(p_{jk})$. Since $m^s_{jk} \geq I(p_{jk}, t_m)$ for all $t_m \in T(p_{jk}) \cap T(t_s)$, $t_s \circ t_m$ in M' if and only if $t_s \circ t_m$ in M^i.

Otherwise viewed, t_s is firable in any such marking M' if and only if t_s is firable in M^i. Therefore, it is sufficient to consider only vectors β for which $0 \leq \beta_k \leq m^s_{jk}$, for $q+1 \leq k \leq n$.

Suppose now that $p_{jk} \in \mathscr{D}(I_s)$. Then, $t_s \circ t_m$ in some marking M^i, where $t_m \in T(p_{jk})$, $t_m \neq t_s$, only if $M^i(p_{jk}) < M_{jk}$. Suppose on the other hand that M^i is a marking of PN whose restriction to $\pi(t_s)$ is a vector in $M(t_s)$ and $M^i(p_{jk}) = M_{jk}$. Let M' be another marking of PN such that $M'(p_{jr}) = M^i(p_{jr})$ for all $p_{jr} \in \pi(t_s)$, $r \neq k$, and $M'(p_{jk}) > M_{jk}$. Then there is no conflict at p_{jk} either in M^i or in M' and t_s is firable in any such marking M' if and only if t_s is firable in M^i.

In particular, if in some marking M^i, $M^i(p_{jk}) \geq M_{jk}$ for all places $p_{jk} \in \mathscr{D}(I_s)$ then t_s is firable in M^i independent of the status of any other transition of the net. Since we are interested at this point in markings M^i such that for at least one place $p_{jk} \in \mathscr{D}(I_s)$, $M^i(p_{jk}) < M_{jk}$, we have excluded from $M(t_s)$ the vectors $(\beta_1, \ldots, \beta_q, \beta_{q+1}, \ldots, \beta_n)$ for which $\beta_k = M_{jk}$ for each k, $1 \leq k \leq q$.

Let $\bar{M}(t_s)$ denote the subset of $M(t_s)$ such that if $\beta \in \bar{M}(t_s)$ then for any marking M^i of PN whose restriction to $\pi(t_s)$ is β, t_s is firable in M^i ($\bar{M}(t_s)$ is determined by eliminating from $M(t_s)$ the vectors which do not satisfy the condition (*) given at the beginning of this proof). Suppose $\beta = (\beta_1, \ldots, \beta_q, \beta_{q+1}, \ldots, \beta_n)$ is a vector in $\bar{M}(t_s)$. A vector $\beta' = (\beta'_1, \ldots, \beta'_q, \beta'_{q+1}, \ldots, \beta'_n)$ is called an <u>extension</u> of β if for each k, $1 \leq k \leq q$, either $\beta'_k = \beta_k$ or $\beta'_k > M_{jk}$ if $\beta_k = M_{jk}$ and for each k, $q+1 \leq k \leq n$, either $\beta'_k = \beta_k$ or $\beta'_k > m^s_{jk}$ if $\beta_k = m^s_{jk}$.

According to the previous discussion, t_s is firable in some marking M^i if and only if the restriction of M^i to $\pi(t_s)$ is a vector in $\bar{M}(t_s)$ or an extension of a vector in $\bar{M}(t_s)$ or if $M^i(p_{jk}) \geq M_{jk}$ for all places $p_{jk} \in \mathscr{D}(I_s)$.

With these remarks we can now present a conversion procedure by means of which a given Labelled PPM(I) can be converted into a λ-transition free, language equivalent Labelled EPM(I). The essential factor in our procedure is the fact that the set $\bar{M}(t_s)$ is finite for each transition t_s of the original PPN(I). Our construct will preserve the places of the original net. If t_s is a transition such that $H(t_s) = \phi$ then we shall introduce in the EPN(I) EN = (T', P, I', O') a

transition t'_s which is the exact copy of the original transition t_s, i.e. t'_s has the same input and output connections as t_s.

Let now t_s be a transition such that $\bar{H}(t_s) \neq \phi$. For each vector $\beta_v = (\beta_{v1}, \ldots, \beta_{vq}, \beta_{vq+1}, \ldots, \beta_{vn})$ in $\bar{M}(t_s)$ we shall introduce a transition t'_{sv} in T' such that:

$$I'(p_{jk}, t'_{sv}) = \begin{cases} \zeta & \text{if } \beta_{vk} = 0 \\ (\sigma, \beta_{vk}) & \text{if } 0 < \beta_{vk} < M_{jk} \\ \beta_{vk} & \text{if } \beta_{vk} = M_{jk} \end{cases} \qquad 1 \leq k \leq q \quad (p_{jk} \in \mathcal{D}(I_s))$$

$$I'(p_{jk}, t'_{sv}) = \begin{cases} \zeta & \text{if } \beta_{vk} = 0 \\ (\sigma, \beta_{vk}) & \text{if } 0 < \beta_{vk} < m^s_{jk} \\ \beta_{vk} & \text{if } \beta_{vk} = m^s_{jk} \end{cases} \qquad q+1 \leq k \leq n \quad (p_{jk} \in \pi'(t_s))$$

$$I'(p_j, t'_{sv}) = 0 \quad \text{for any } p_j \in P - \pi(t_s).$$

$$O'(t'_{sv}, p_{jk}) = \beta_{vk} - I(p_{jk}, t_s) + O(t_s, p_{jk}) \text{ for all } p_{jk} \in \pi(t_s)$$

$$O'(t'_{sv}, p_j) = O(t_s, p_j) \text{ for any } p_j \in P - \pi(t_s).$$

Finally, we shall introduce for t_s one more replacement transition, let it be t'_s, such that:

$$I'(p_j, t'_s) = \begin{cases} M_j & \text{if } p_j \in \mathcal{D}(I_s) \\ 0 & \text{if } p_j \in P - \mathcal{D}(I_s) \end{cases}$$

$$O'(t'_s, p_j) = \begin{cases} M_j - I(p_j, t_s) + O(t_s, p_j) & \text{if } p_j \in \mathcal{D}(I_s) \\ O(t_s, p_j) & \text{if } p_j \in P - \mathcal{D}(I_s) \end{cases}$$

Our construct preserves the places and the set of reachable markings of the original PPN(I). In any marking at most one replacement transition introduced for a transition of the original net can be enabled (and consequently firable). The input connections of each replacement transition were determined such that the transition can be enabled only in a marking in which the corresponding transition of the original net is also firable. The output connections of each replacement transition ensure that the firing of that transition in some marking M^i produces the same effect on M^i as the corresponding original transition. It can be verified that the resulting EPM(I) generates the same language as the given PPM(I) if all replacement transitions carry the same label as the original transition to which they correspond.

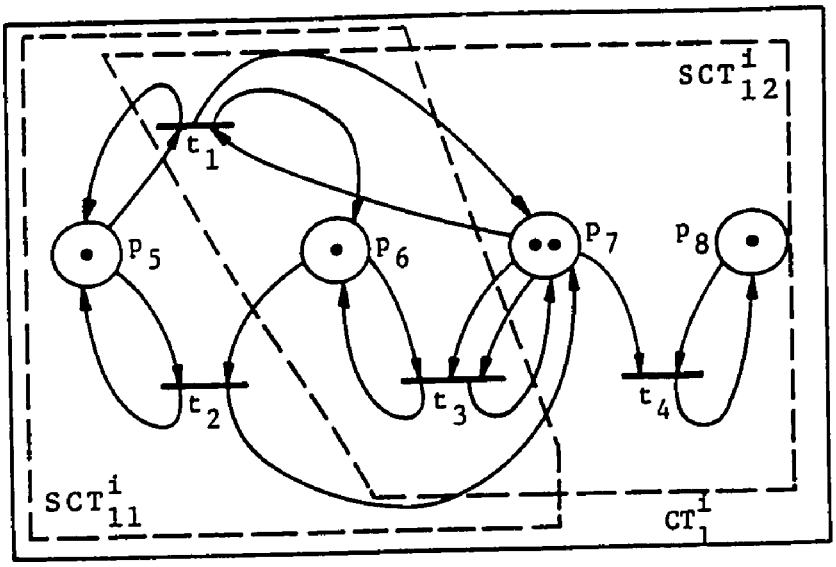
Figure 4.3.1 exhibits a sample PPN(I). The associated partial ordering Pr = (Q, $\underline{\alpha}$) is given in its Hasse diagram representation. In the given marking M^i, all transitions are enabled and form a unique conflict cluster CT_1^i (marked in the picture with solid line). CT_1^i in turn contains two conflict subclusters, SCT_{11}^i and SCT_{12}^i (marked in the picture with dotted line). It can easily be seen that t_1 and t_3 are firable in M^i while t_2 and t_4 are not. Figure 4.3.2 depicts the EPN(I) obtained by applying the construct of Lemma 4.3.1 to the PPN(I) of Figure 4.3.1. Since the output arcs of a transition are irrelevant from the point of view of the firability of that transition in any marking, we have deliberately omitted from Figure 4.3.2 all output arcs in order to simplify the picture. In the given marking, t_1' and t_3' are enabled and therefore, firable. It can be verified that none of the replacement transitions introduced for the transitions t_2 and t_4 of the original PPN(I) is enabled.

Let us examine for a moment the differences between the Priority Petri Model and the Priority Petri Model (I). Clearly, there are no "structural" differences, the only distinction consists in the rule applied for the selection of a transition to be fired. Figure 4.3.3 depicts a Priority Petri Net. According to Definition 4.1.6, in the given marking, let it be M^i, only transition t_2 is firable. If we consider the same net as a Priority Petri Net (I) (Figure 4.3.4) then it can be seen that in the same marking M^i the transitions of the net are partitioned into two conflict clusters CT_1^i and CT_2^i, which in turn contain three conflict subclusters SCT_{11}^i, SCT_{12}^i and SCT_{21}^i, respectively. According to Definition 4.3.3, in the given marking, both t_2 and t_4 are firable. Obviously, the same net may generate distinct firing sequences (and for that matter distinct label sequences if, for example, t_2 and t_4 carry different labels) depending on which execution rule is used. Nevertheless, one can show that:

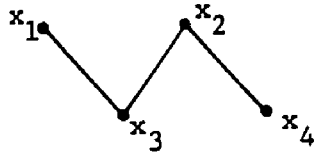
Lemma 4.3.2

$\Lambda(PPM) \subseteq \Lambda(PPM(I))$ and $\Lambda_o(PPM) \subseteq \Lambda_o(PPM(I))$.

Proof. Let $PN_\Sigma = (PN, \Sigma, L)$ be a Labelled PPM where $PN = (PN, M^o)$, PN = (N, Pr, Y) and N = (T, P, I, O). Suppose PN is in some marking M^i. According to Definition 4.1.6, a transition $t_k \in T$ is firable in M^i if and only if $t_k \in E^i$ and $T_k \cap E^i = \phi$ where



$P_r = (Q, \underline{\alpha})$:



t_s	$Y(t_s)$
t_1	x_1
t_2	x_3
t_3	x_2
t_4	x_4

t_s	$T(t_s)$	$H(t_s)$
t_1	t_1, t_2, t_3, t_4	ϕ
t_2	t_1, t_2, t_3	t_1, t_3
t_3	t_1, t_2, t_3, t_4	ϕ
t_4	t_1, t_3, t_4	t_3

$M_5 = 2$
$M_6 = 2$
$M_7 = 4$
$M_8 = 1$

$\boxed{t_2}$ $\pi'(t_2) = \{p_7\}$ $\pi(t_2) = \{p_5, p_6, p_7\}$ $m^2_7 = 2$

$\bar{M}(t_2)$:

p_5	p_6	p_7
1	1	0
1	2	0
2	1	0
2	1	1

$\boxed{t_4}$ $\pi'(t_4) = \{p_5, p_6\}$ $\pi(t_4) = \{p_5, p_6, p_7, p_8\}$ $m^4_5 = 1$, $m^4_6 = 1$

$\bar{M}(t_4)$:

p_5	p_6	p_7	p_8	p_5	p_6	p_7	p_8
0	0	1	1	0	0	3	1
1	0	1	1	1	0	3	1
0	1	1	1	0	1	3	1
1	1	1	1				
0	0	2	1				
1	0	2	1				

Figure 4.3.1
Sample PPN(I)

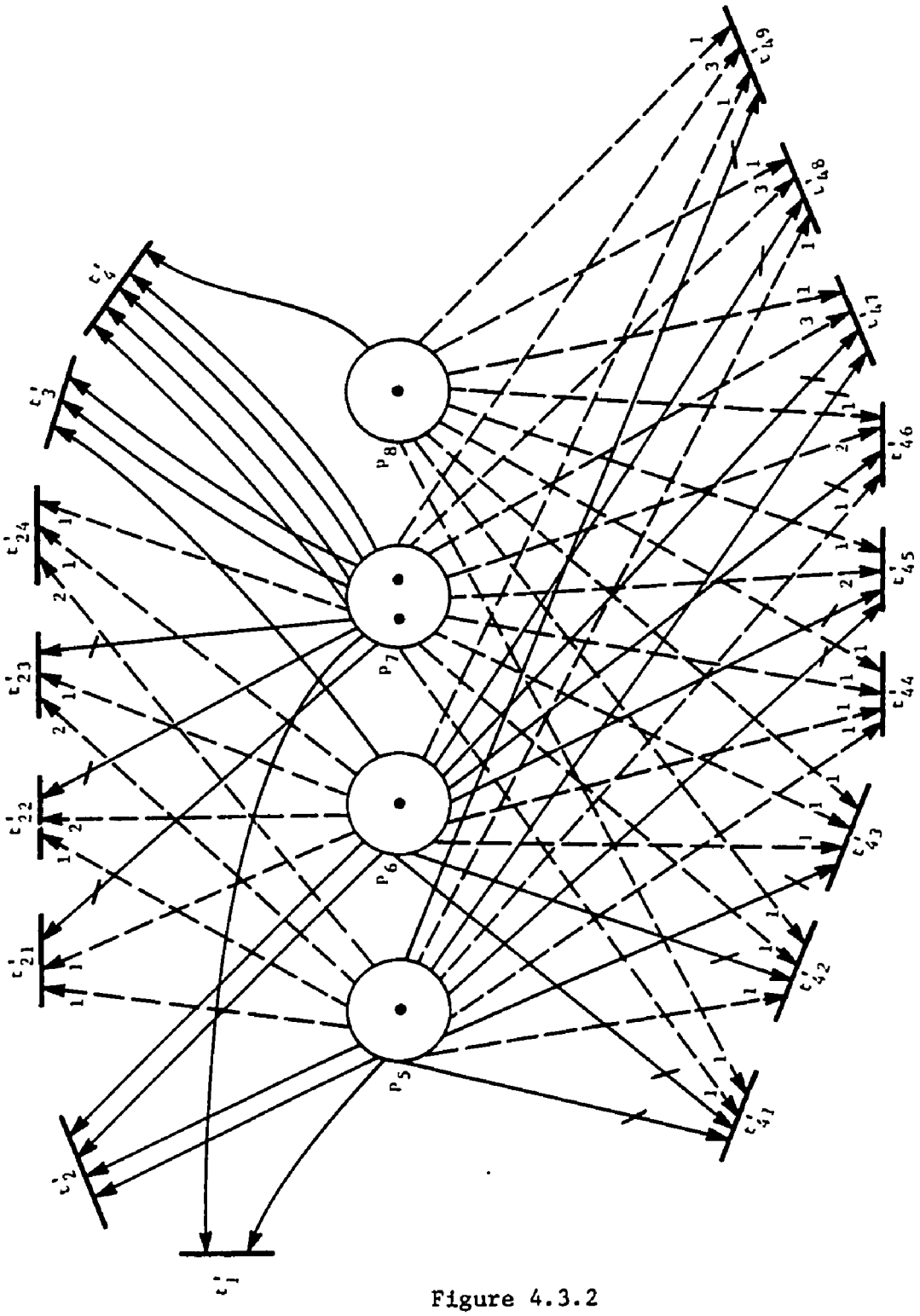
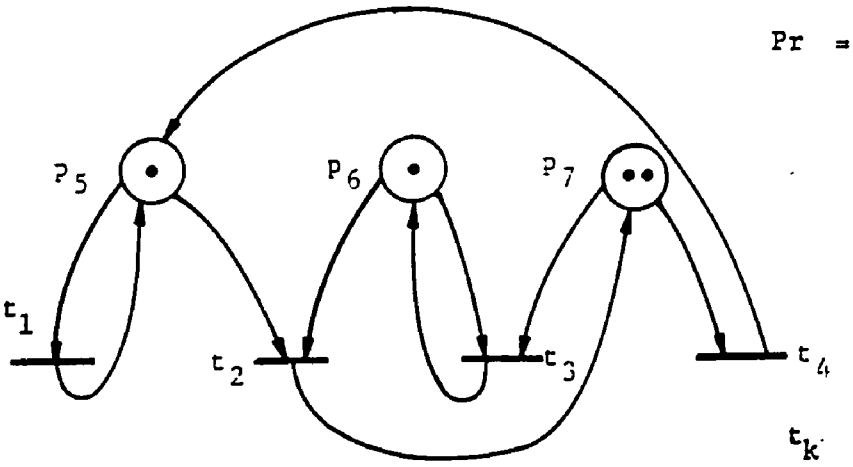
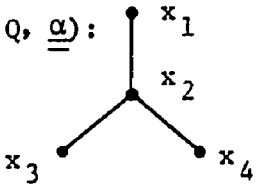


Figure 4.3.2

EPN(I) Obtained from the PPN(I) of Figure 4.3.1



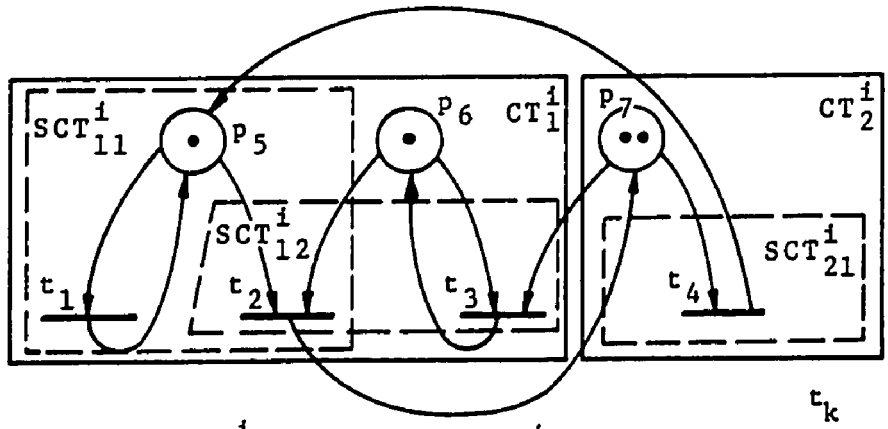
$Pr = (Q, \underline{\alpha})$:

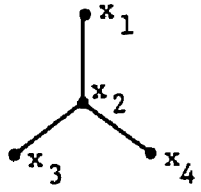


t_k	t_1	t_2	t_3	t_4
$Y(t_k)$	x_3	x_1	x_2	x_4

Figure 4.3.3
Sample PPN

$Pr = (Q, \underline{\alpha})$:





$maj\{SCT^i_{11}\} = t_2$; $maj\{SCT^i_{12}\} = t_2$
$maj\{SCT^i_{21}\} = t_4$

t_k	t_1	t_2	t_3	t_4
$Y(t_k)$	x_3	x_1	x_2	x_4

Figure 4.3.4
Sample PPN(I)

$$T_k = \{t_s \mid t_s \in T \text{ and } Y(t_k) \alpha Y(t_s)\}$$

Let us form a Labelled PPM(I) $\mathcal{PN}'_\Sigma = (PN', \Sigma, L)$ where $\mathcal{PN}' = (PN', M^{o'})$, $PN' = (N', Pr, Y)$ and $N' = (T, P', I', O')$. We shall have $P' = P \cup \{p_o\}$, where $p_o \notin P$, and for each $t_k \in T$:

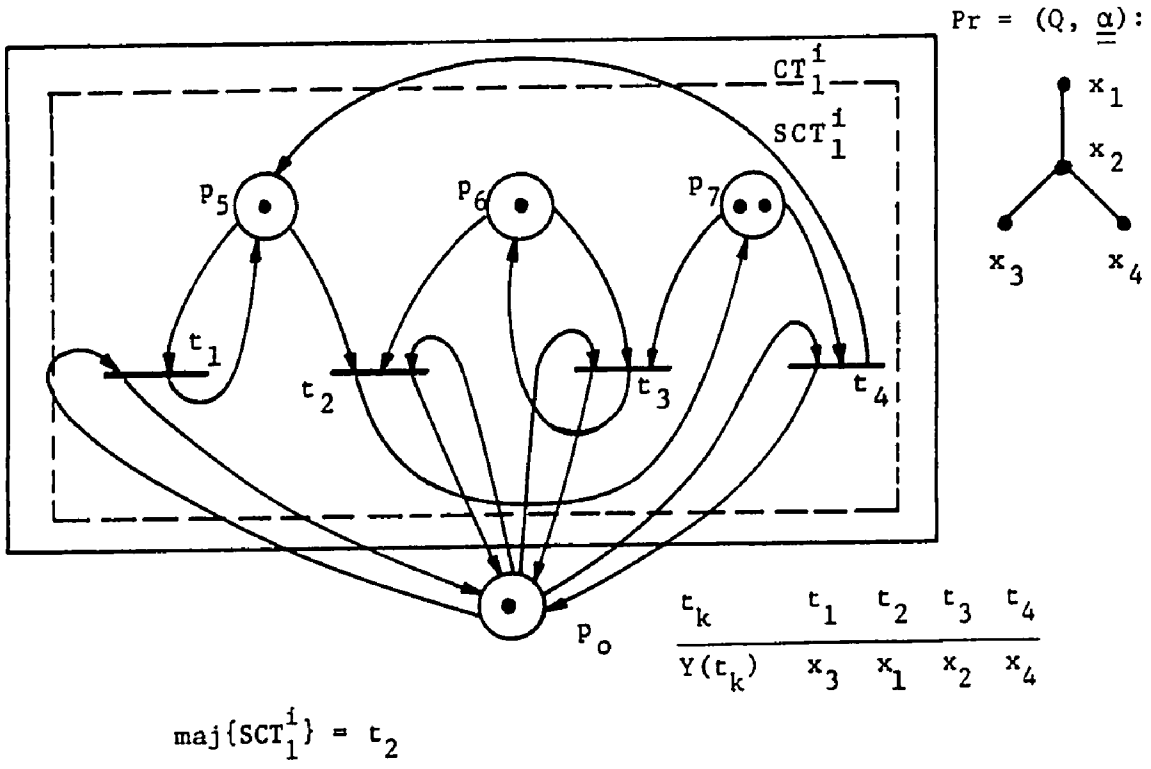


Figure 4.3.5

PPN(I) Obtained from the PPN of Figure 4.3.3

$$I'(p_j, t_k) = \begin{cases} I(p_j, t_k) & \text{if } p_j \in P \\ 1 & \text{if } p_j = p_o \end{cases}$$

$$O'(t_k, p_j) = \begin{cases} O(t_k, p_j) & \text{if } p_j \in P \\ 1 & \text{if } p_j = p_o \end{cases}$$

Moreover, $M^{o'}(p_j) = M^o(p_j)$ for each $p_j \in P$ and $M^{o'}(p_o) = 1$. In other words, we have introduced in the GPN N of PN a control place p_o on which all transitions of PN' self-loop. Obviously, the control place does not alter the conditions under which a particular transition $t_k \in T$ is enabled but, as we shall see, p_o drastically changes the structure of the conflict clusters and subclusters which can be formed in any marking M^i.

Since all transitions $t_k \in E^i$ conflict at p_o we shall have in any marking M^i for which $E^i \neq \phi$ a unique conflict cluster, let it be CT_1^i, such that $CT_1^i = E^i$. From the definition of a conflict subcluster follows that we shall also have only one conflict subcluster, let us denote it by SCT_{11}^i, such that $SCT_{11}^i = E^i$. Therefore, in the case of PN', Definition 4.3.3 becomes: a transition $t_k \in T$ is firable in M^i if and only if t_k is a majorant of E^i. Equivalently, $t_k \in T$ is firable in the marking M^i if and only if $t_k \in E^i$ and $T_k \cap E^i = \phi$, where T_k has the same meaning as in the case of the PPM. This is, however, exactly the rule for the selection of a firable transition in the PPM. Consequently the Labelled PPM(I) PN'_Σ generates the same firing sequences, and therefore the same label sequences, as the Labelled PPM PN_Σ. □

The PPN(I) of Figure 4.3.5 examplifies the construct of Lemma 4.3.2 as applied to the PPN of Figure 4.3.3. We have seen (Lemma 4.3.1) that $\Lambda(PPM(I)) \subseteq \Lambda(EPM(I))$ and $\Lambda_o(PPM(I)) \subseteq \Lambda_o(EPM(I))$. From this and Theorems 4.2.1 and 4.1.1(a) we then have $\Lambda(PPM(I)) \subseteq \Lambda(PPM)$ and $\Lambda_o(PPM(I)) \subseteq \Lambda_o(PPM)$. Lemma 4.3.2 proves the containment in reverse direction. Consequently, we can now state the main result of this section:

Theorem 4.3.1

$\Lambda(PPM) = \Lambda(PPM(I))$ and $\Lambda_o(PPM) = \Lambda_o(PPM(I))$.

Section 4.4 THE C-COLORED PETRI MODEL (I)

In this section we shall introduce another class of the CPM, closely related to the C-CPM, which we shall call the C-Colored Petri Model (I) (C-CPM(I)). The C-CPM(I) will be used in our study of the relationship between the C-CPM and the PPM.

Definition 4.4.1

A C-Colored Petri Net (I) (C-CPN(I)) is a Colored Petri Net $CN = (N, C, F, R, Q)$ where:

1. N, C, F, Q are as in Definition 2.4.1.
2. $R = R_E$ is the identity relation on the set X.

Since all models in the C-CPM(I) class will employ the relation R_E for the selection of enabling colors, we shall omit it when defining a C-CPN(I).

Clearly, the C-CPN(I) resembles closely the C-CPN, the only distinction consisting in the rule used for the selection of enabling colors. Thus, let $CN = (N, C, F)$ be a C-CPN(I) in some color marking CM^i. Let $N = (T, P, I, O)$ be the underlying GPN and suppose $t_k \in T$, $p_j \in P$ and p_j is an input place of t_k. Let us assume that $I(p_j, t_k) = m$, for some integer $m > 0$, and consider an arbitrary input arc a^r_{jk}, $1 \leq r \leq m$. Let us denote by f^r_{jk} the color threshold function associated with a^r_{jk} where $f^r_{jk}: \mathscr{P}(X) \rightarrow c^r_{jk}$, for some color $c^r_{jk} \in X$. Definition 2.2.2 becomes:

Definition 4.4.2

A color $c \in \mathscr{D}(C^i_j)$ is the enabling color of transition t_k on the input arc a^r_{jk} if and only if $c = c^r_{jk}$.

As we shall see, this rule for the selection of enabling colors will influence the execution rules of the C-CPM(I). We can already notice that irrespective of the color marking CM^i, there cannot exist a set of distinct, incomparable colors which are all eligible to become the enabling color of t_k on the input arc a^r_{jk}. Thus, in the case of the C-CPN(I), unlike the case of the C-CPN, there does not exist a choice in the selection of enabling colors. Consequently, the bag of enabling colors of t_k from the input place p_j becomes:

$$\theta^i_{jk} = \bigcup_{r=1}^{I(p_j,\, t_k)} c^r_{jk}$$

where $f^r_{jk} : \mathscr{P}(X) \to c^r_{jk}$, $c^r_{jk} \in X$, for each r, $1 \le r \le I(p_j, t_k)$. Hence, the bag of enabling colors of t_k is:

$$\theta^i_k = \bigcup_{p_j \in \mathscr{D}(I_k)} \left[\bigcup_{r=1}^{I(p_j,\, t_k)} c^r_{jk} \right]$$

We notice that in the case of the C-CPN(I), as opposed to the C-CPN, neither θ^i_{jk} (for each $p_j \in \mathscr{D}(I_k)$) nor θ^i_k depends on the color marking CM^i; therefore we shall drop the superscript i. We can now rewrite the condition for a transition $t_k \in T$ to be enabled as follows:

Definition 4.4.3

A transition t_k of the C-CPN(I) CN is enabled in some color marking CM^i if and only if for each $p_j \in \mathscr{D}(I_k)$:

$$\theta_{jk} = \bigcup_{r=1}^{I(p_j,\, t_k)} c^r_{jk} \subseteq c^i_j$$

Let us denote for each place $p_j \in \mathscr{D}(O_k)$

$$\alpha_{kj} = \bigcup_{r=1}^{O(t_k,\, p_j)} c^r_{kj} \qquad \text{(bag union)}$$

where f^r_{kj}: $\mathscr{P}(X) \times \mathscr{P}(X) \to c^r_{kj}$, $c^r_{kj} \in X$, for each r, $1 \le r \le O(t_k, p_j)$. Let us also make the convention that $\theta_{jk} = \phi$ if $p_j \notin \mathscr{D}(I_k)$ and

$\alpha_{kj} = \phi$ if $p_j \notin \mathcal{D}(O_k)$. Then:

Definition 4.4.4

If a transition t_k is enabled in the color marking CM^i and $CM^i[t_k > CM^{i+1}$ then for each $p_j \in P$

$$CM^{i+1}(p_j) = (CM^i(p_j) - \theta_{jk}) \cup \alpha_{kj}.$$

We notice that in the case of the C-CPN(I) the effect of the firing of a transition in any color marking CM^i is fixed and therefore if $CM^i[t_k > CM^{i+1}$ then the color marking CM^{i+1} is uniquely determined by CM^i and t_k.

Regarding the dynamic behavior of a C-CPN(I), we notice that the priority of a transition t_k is $g.\ell.b(\mathcal{D}(\theta_k))$ and therefore fixed and independent of the color marking of the net (we remember that in the case of the C-CPN the priority of a transition depends on the particular selection of enabling colors and therefore may vary from one color marking to another and even for the same color marking).

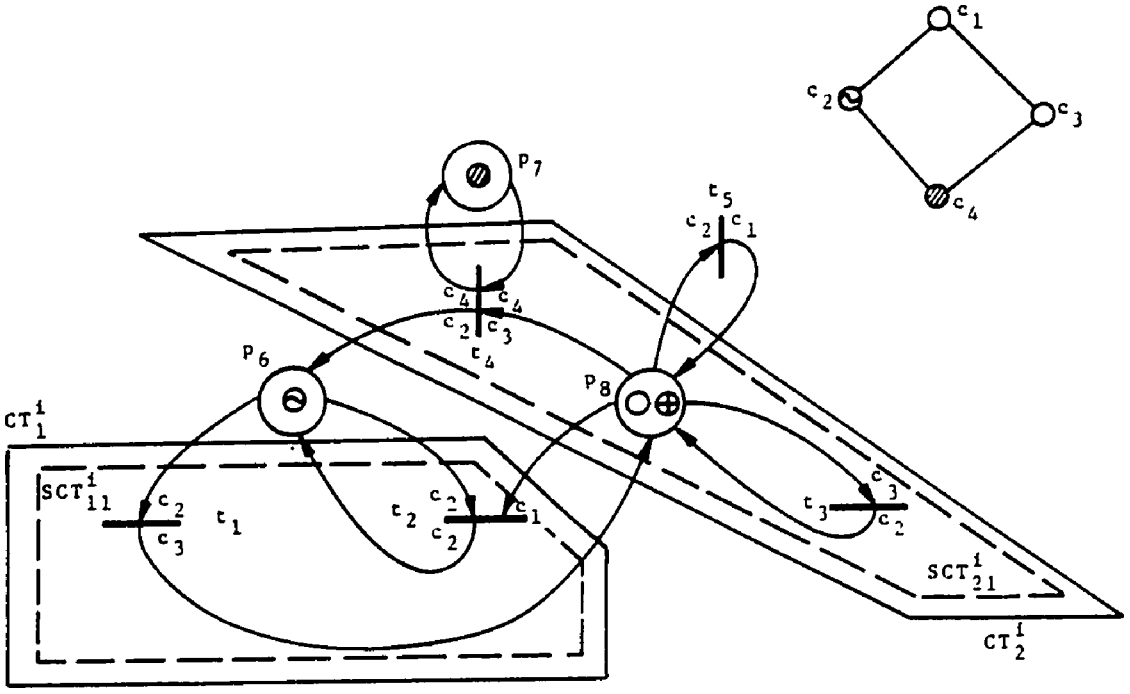
The execution of a C-CPN(I) follows the execution rule given for the CPM in Section 2.3, hence an enabled transition is firable in some color marking CM^i if and only if it is a majorant of all the conflict subclusters in which it is contained. The definitions of Section 2.3 can immediately be particularized for the case of the C-CPN(I). Finally, a C-CPM(I) is defined analogous to Definition 2.4.2.

Figure 4.4.1 exhibits a sample C-CPN(I). All transitions, with the exception of t_5, are enabled in the given color marking, let us denote it by CM^i. There are two conflict clusters in CM^i, which we have denoted by CT_1^i and CT_2^i, each of which contains a conflict subcluster, SCT_{11}^i and SCT_{21}^i, respectively. It is easy to see that transitions t_1, t_2 and t_3 are firable while t_4 is not.

A Labelled C-CPM(I) is defined completely analogous to a Labelled C-CPM. Similarly, the language families Λ(C-CPM(I)) and Λ_o(C-CPM(I)) are defined the same way as Λ(C-CPM) and Λ_o(C-CPM), respectively.

Lemma 4.4.1

$\Lambda(\text{C-CPM(I)}) \subseteq \Lambda(\text{PPM(I)})$ and $\Lambda_o(\text{C-CPM(I)}) \subseteq \Lambda_o(\text{PPM(I)})$.



t_k	θ_k	g.l.b.($\mathscr{D}(\theta_k)$)	
t_1	$<c_2>$	c_2	conflict at the place p_6 for the color c_2
t_2	$<c_2, c_1>$	c_2	
t_3	$<c_3>$	c_3	conflict at the place p_8 for the color c_3
t_4	$<c_3, c_4>$	c_4	
t_5	$<c_2>$	c_2	

Figure 4.4.1
Sample C-CPN(I)

Proof: Let $\mathcal{CP}_\Sigma = (\mathcal{CP}, \Sigma, L)$ be an arbitrary Labelled C-CPM(I) where $\mathcal{CP} = (CN, CM^o)$, $CN = (N, C, F)$, $N = (T, P, I, O)$ is the underlying GPN and $C = (X, \leq)$ is the set of colors associated with N. We want to construct a language-equivalent, λ-transition free Labelled PPM(I) $\mathcal{PN}_\Sigma = (\mathcal{PN}, \Sigma, L')$. Let $\mathcal{PN} = (PN, M^o)$ where $PN = (N', Pr, Y)$ and $N' = (T', P', I', O')$.

We shall first define the structure of the GPN N'. Let $|T'| = |T|$, i.e. for each transition $t_k \in T$ we shall introduce a transition t'_k in T' (t'_k will be called "the transition of T' corresponding to $t_k \in T$"). Next, for each place $p_j \in P$ we shall introduce in P' a set of places $\{p_{j1}, \ldots, p_{j|X|}\}$. In order to simplify the notations we shall make use of the following mappings:

a) $\mathcal{S}: P \to \mathcal{P}(P')$ ($\mathcal{P}(P')$ denotes the power set of the set P') such that for each place $p_j \in P$ we have $\mathcal{S}(p_j) = \{p_{j1}, \ldots, p_{j|X|}\}$

b) For each color $c_r \in X$ let $\mathcal{S}_r : P \to P'$ be the function defined by $\mathcal{S}_r(p_j) = p_{jr}$, for all $p_j \in P$.

c) $\mathcal{F}: T \to T'$ such that $\mathcal{F}(t_k) = t_k'$ for each $t_k \in T$, and t'_k is the transition of T' corresponding to t_k.

The mapping $\mathcal{F}$ is both one-to-one and onto while the mappings $\mathcal{S}_r$ (for each $c_r \in X$) and $\mathcal{S}$ are only one-to-one.

d) Let $R(CM^o)$ and $R(M^o)$ denote the set of reachable color markings and the set of reachable markings of $\mathcal{CP}$ and $\mathcal{PN}$, respectively. Let $\mathcal{C}: R(CM^o) \to R(M^o)$ be a mapping such that given some marking $M^i \in R(M^o)$ and some color marking $CM^s \in R(CM^o)$ then $M^i = \mathcal{C}(CM^s)$ if and only if for each $p_j \in P$ and any $c_r \in X$

$$M^i(\mathcal{S}_r(p_j)) = \#(c_r, C_j^s)$$

We can proceed now to define the input and output incidence functions of N'. Suppose $t_k \in T$ and let us assume that $p_j \in \mathcal{D}(I_k)$. We shall denote for each color $c_r \in X$, $\#(c_r, \theta_{jk}) = m_{jk}^r$; certainly $\sum_{c_r \in X} m_{jk}^r = I(p_j, t_k)$. For each $c_r \in X$, let:

$$I'(\mathcal{S}_r(p_j), \mathcal{F}(t_k)) = m_{jk}^r.$$

This rule is repeated for all $p_j \in \mathscr{D}(I_k)$ and all transitions $t_k \in T$.

Suppose now that $p_s \in \mathscr{D}(O_k)$ and let $m_{ks}^r = \#(c_r, \alpha_{ks})$. In PN we shall then have:

$$O'(\mathscr{F}(t_k), \mathscr{S}_r(p_s)) = m_{ks}^r \text{ , for each } c_r \in X .$$

This rule is repeated for all $p_s \in \mathscr{D}(O_k)$ and all $t_k \in T$. The definition of the GPN $N' = (T', P', I', O')$ is now complete. Let us next set $P_r = C$ and define the priority function Y by

$$Y(t_k') = \text{g.l.b.}(\mathscr{D}(\theta_{\mathscr{F}^{-1}(t_k')})),$$

for each transition $t_k' \in T'$.

Figure 4.4.2 displays a typical transition of a C-CPN(I) while Figure 4.4.3 illustrates the result of the construct given above, as applied to the transition of Figure 4.4.2.

Suppose now that the C-CPN(I) CN is in some color marking CM^i and let $t_k \in T$ be a transition enabled in CM^i. Hence, for each place $p_j \in \mathscr{D}(I_k)$ and for all colors $c_r \in X$, we must have:

$$\#(c_r, C_j^i) \geq \#(c_r, \theta_{jk})$$

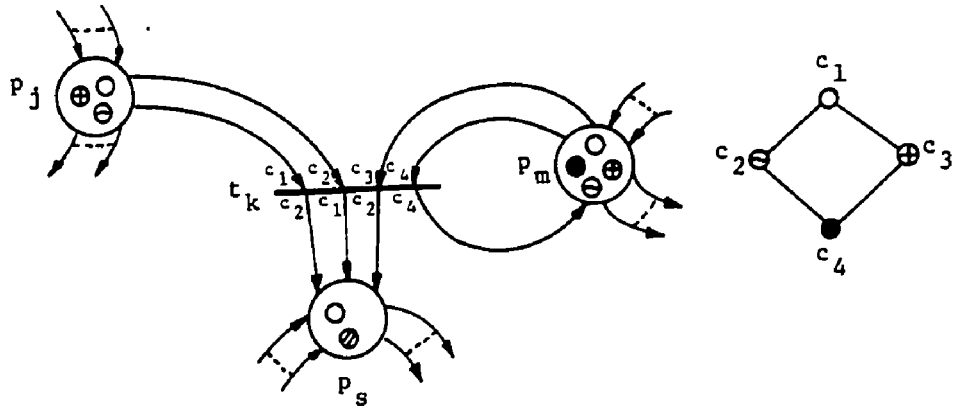
Let M^s be the marking of PN such that $M^s = \mathscr{C}(CM^i)$. Then, for each place $p_{jr} = \mathscr{S}_r(p_j)$ in $\mathscr{S}(p_j)$, for all $p_j \in \mathscr{D}(I_k)$, we must have:

$$M^s(\mathscr{S}_r(p_j)) = \#(c_r, C_j^i) \geq \#(c_r, \theta_{jk}) = m_{jk}^r$$

$$= I'(\mathscr{S}_r(p_j), \mathscr{F}(t_k))$$

Therefore, $\mathscr{F}(t_k)$ is enabled in the marking M^s and consequently the set of transitions of PN enabled in M^s is $\mathscr{F}(E^i)$, where E^i denotes the set of transitions of CN enabled in CM^i and the mapping $\mathscr{F}$ has been naturally extended to a set.[‡]

[‡] If $V = \{t_{k1}, \ldots, t_{kn}\}$ is an arbitrary subset of T then $\mathscr{F}(V) = \{\mathscr{F}(t_{k1}), \ldots, \mathscr{F}(t_{kn})\} = \{t_{k1}', \ldots, t_{kn}'\}$; $\mathscr{F}(V) \subseteq T'$.



$\theta_k = < c_1, c_2, c_3, c_4 > \Rightarrow$ g.l.b.($\mathscr{D}(\theta_k)$) $= c_4$

Figure 4.4.2

Typical Transition of a C-CPN(I)

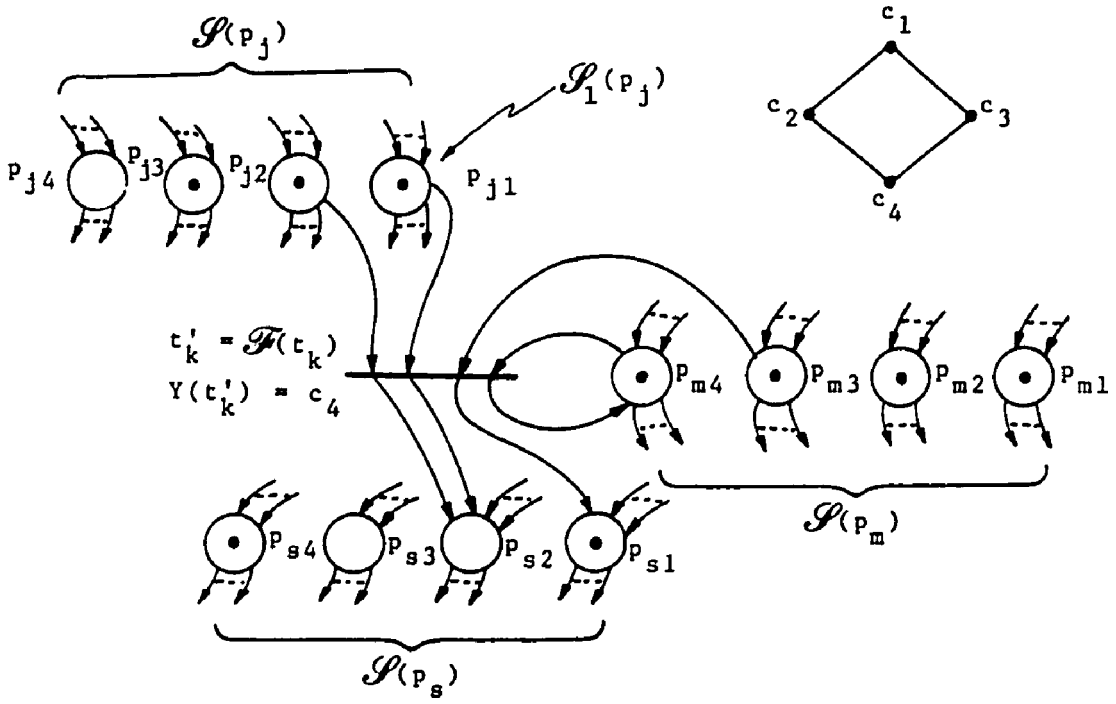


Figure 4.4.3

Transition of a PPN(I) Obtained from the Transition of Figure 4.4.2

Let us now suppose that $p_j \in P$ is a place of CN such that there exists a conflict at p_j for the color c_r in the color marking CM^i. Let us define for each $p_j \in P$ and for all $c_r \in X$ the sets:

$$T^r(p_j) = \{t_k \mid t_k \in T \text{ and } \#(c_r, \theta_{jk}) > 0\}$$

Certainly, $T^r(p_j) \subseteq T(p_j)$. Then:

$$\sum_{t_k \in T(p_j) \cap E^i} \#(c_r, \theta_{jk}) = \sum_{t_k \in T^r(p_j) \cap E^i} \#(c_r, \theta_{jk}) > \#(c_r, C_j^i)$$

For each t_k and t_s in $T^r(p_j) \cap E^i$ we shall say that "t_k and t_s conflict at the place p_j for the color c_r in the color marking CM^i."

Since $\mathcal{F}$ is an isomorphism we must have:

$$\mathcal{F}(T^r(p_j) \cap E^i) = \mathcal{F}(T^r(p_j)) \cap \mathcal{F}(E^i) = T(\mathcal{S}_r(p_j)) \cap \mathcal{F}(E^i)$$

Hence

$$\sum_{t_k' \in T(\mathcal{S}_r(p_j)) \cap \mathcal{F}(E^i)} I'(\mathcal{S}_r(p_j), t_k') = \sum_{t_k \in T(p_j) \cap E^i} \#(c_r, \theta_{jk})$$

Since $M^s = \mathcal{C}(CM^i)$ we must also have

$$\sum_{t_k' \in T(\mathcal{S}_r(p_j)) \cap \mathcal{F}(E^i)} I'(\mathcal{S}_r(p_j), t_k') > \#(c_r, C_j^i) = M^s(\mathcal{S}_r(p_j))$$

Hence there is a conflict at $\mathcal{S}_r(p_j)$ in the marking M^s. Using a similar argument it can be shown that t_k and t_m conflict at p_j for the color c_r in the color marking CM^i if and only if $\mathcal{F}(t_k)$ and

$\mathcal{F}(t_m)$ conflict at $\mathcal{S}_r(p_{j_i})$ in the marking M^s. It is therefore easy to see that $t_k \circ t_m$ in CM^i if and only if $\mathcal{F}(t_k) \circ \mathcal{F}(t_m)$ in M^s. It follows that SCT^i_{rn} is a conflict subcluster of CN in the color marking CM^i if and only if $\mathcal{F}(SCT^i_{rn})$ is a conflict subcluster of PN in the marking M^s. Conversely, t_k participates in CM^i in a conflict subcluster SCT^i_{rn} if and only if $\mathcal{F}(t_k)$ participates in M^s in the conflict subcluster $\mathcal{F}(SCT^i_{rn})$. According to the definition of the priority function Y, $\mathcal{F}(t_k)$ can be a majorant of all conflict subclusters in which it particpates in M^s if and only if t_k is a majorant of all conflict subclusters in which it participates in CM^i. We can conclude that $\mathcal{F}(t_k)$ is firable in M^s if and only if t_k is firable in CM^i, where $M^s = \mathcal{C}(CM^i)$.

From the definition of the functions I' and O' follows that if $CM^i[t_k > CM^{i+1}$ then $\mathcal{C}(CM^i)[\mathcal{F}(t_k) > \mathcal{C}(CM^{i+1})$. Thus, if $M^o = \mathcal{C}(CM^o)$ and $L'(\mathcal{F}(t_k)) = L(t_k)$ for each $t_k \in T$ then PN_Σ generates the same language as CP_Σ (if a final color marking CM^f is given then, obviously, $M^f = \mathcal{C}(CM^f)$).

□

Note: The number of places of PN is considerably larger than the number of places of CP, i.e. $|P'| = |X| \cdot |P|$. We can, however, reduce the number of places used in our construct by introducing in P' places p_{jr} only for the colors $c_r \in X$ which may occur in p_j (in any reachable color marking), for all $p_j \in P$. Thus, suppose $p_j \in P$ and let

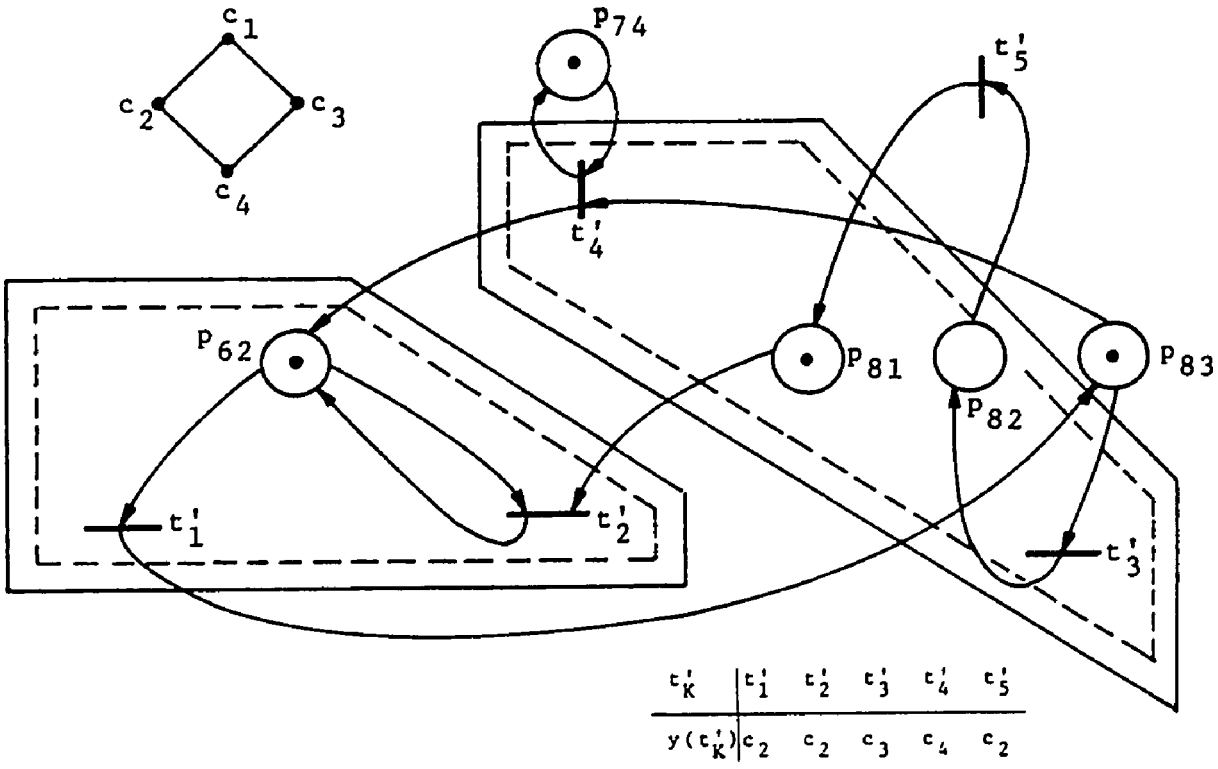
$$X_j = \{c_r \mid c_r \in X \text{ and } \#(c_r, C^o_j) > 0\} \cup \left[\bigcup_{t_k \in T} \mathcal{D}(\alpha_{kj})\right]$$

(remember that $\alpha_{kj} = \phi$ if $p_j \notin \mathcal{D}(O_k)$). Then, the mapping $\mathcal{S}$ is redefined as follows:

$$\mathcal{S}(p_j) = \{p_{jr} \mid c_r \in X_j\} \quad \text{for each } p_j \in P.$$

Also, $\mathcal{S}_r(p_j)$ is undefined if $c_r \notin X_j$.

The complete construct employed in Lemma 4.4.1 is examplified in Figure 4.4.4 as applied to the sample C-CPN(I) of Figure 4.4.1. It is easy to verify that only transitions t'_1, t'_2 and t'_3 are firable in the given marking.



t'_K	t'_1	t'_2	t'_3	t'_4	t'_5
$y(t'_K)$	c_2	c_2	c_3	c_4	c_2

Figure 4.4.4

PPN(I) Obtained from the C-CPN(I) of Figure 4.4.1

Lemma 4.4.2

$\Lambda(\text{EPM}) \subseteq \Lambda(\text{C-CPM})$ and $\Lambda_o(\text{EPM}) \subseteq \Lambda_o(\text{C-CPM})$

Proof: Let $EN_\Sigma = (EN, \Sigma, L)$ be an arbitrary Labelled EPM where $EN = (EN, M^o)$ and $EN = (T, P, I, O)$ is the underlying EPN. We want to build a language-equivalent, λ-transition free Labelled C-CPM $CP_\Sigma = (CP, \Sigma, L')$. Let $CP = (CN, CM^o)$ where $CN = (N, C, F)$, $N = (T', P', I', O')$ is the underlying GPN and $C = (X, \leq)$ is the color set associated with N.

We shall introduce for each place $p_j \in P$ a distinct place p_j' in P'. Moreover, the set of colors X will contain two distinguished colors c_f and c_e by means of which any reachable marking $M^i(p_j)$ will be encoded in terms of the color marking $CM^i(p_j')$ as follows:

i) if $M^i(p_j) = 0$ (i.e. p_j is "empty") then $CM^i(p_j') = < c_e >$. *)

ii) if $M^i(p_j) > 0$ (i.e. p_j is "full") then $CM^i(p_j') = < c_f^{M^i(p_j)} >$.

Our construct will ensure that no tokens of other colors will ever occur in any such place p_j'.

Let us now define for each transition $t_k \in T$ the following sets:

$$P_k^+ = \mathcal{D}(O_k) \cap P_k^n$$

$$P_k^- = \mathcal{D}(O_k) \cap P_k^z$$

Certainly, $P_k^+ \cap P_k^- = \phi$. Consider an arbitrary transition t_k of EN and let us briefly examine the effect of the firing of t_k in some marking M^i with respect to the places to which t_k is connected:

a) $p_j \in P_k^n - P_k^+$. Thus, $I(p_j, t_k) > 0$ and $O(t_k, p_j) = 0$, i.e. p_j is only a normal input place of t_k. If $M^i(p_j) = I(p_j, t_k)$ then the firing of t_k will change the status of p_j from "full" to "empty." If $M^i(p_j) > I(p_j, t_k)$ then p_j will remain "full" even after the firing of t_k.

b) $p_j \in P_k^+$. In this case both $I(p_j, t_k) > 0$ and $O(t_k, p_j) > 0$. Consequently, the firing of t_k maintains the "full" status of p_j.

c) $p_j \in P_k^z - P_k^-$. Thus, $I(p_j, t_k) = \zeta$ and $O(t_k, p_j) = 0$, i.e. p_j is only an inhibitor input place of t_k. Therefore, t_k can fire only if p_j is "empty" and the firing of t_k will not alter this status of p_j.

d) $p_j \in P_k^-$. In this case $I(p_j, t_k) = \zeta$ and $O(t_k, p_j) > 0$. Thus, t_k can fire only if p_j is "empty" but the firing of t_k will always switch the status of p_j to "full".

e) $p_j \in \mathcal{D}(O_k) - (P_k^+ \cup P_k^-)$. Therefore, $I(p_j, t_k) = 0$ but $O(t_k, p_j) > 0$. After the firing of t_k the status of p_j will always be "full" but the firing of t_k may cause a change in the status of p_j, depending on $M^i(p_j)$.

We shall introduce for each transition $t_k \in T$ a set of replacement transitions; each replacement transition will simulate the firing of t_k with respect to a certain distinct category of markings of EN. Of primary importance to our construct will be to properly detect the changes in status which the firing of t_k may cause for the places to which it is connected.

In order to simplify notations let us suppose that $P_k^n - P_k^+ = \{p_{j1}, \ldots, p_{js}\}$. Obviously, t_k is enabled (and therefore firable) in some marking M^i only if $M^i(p_{jr}) \geq I(p_{jr}, t_k)$ for each r, $1 \leq r \leq s$. The restriction of any such marking M^i to $P_k^n - P_k^+$ can be represented by an s-tuple of binary digits $\beta = (\beta_1, \ldots, \beta_s)$ where:

$$\beta_r = \begin{cases} 1 & \text{if } M^i(p_{jr}) > I(p_{jr}, t_k) \\ & \qquad 1 \leq r \leq s \\ 0 & \text{if } M^i(p_{jr}) = I(p_{jr}, t_k) \end{cases}$$

Let us call β the "binary representation of the restriction of M^i to $\{p_{j1}, \ldots, p_{js}\}$." We note that if $\beta_r = 0$ for some r, $1 \leq r \leq s$, then the firing of t_k in M^i will produce a change in the status of p_{jr}, namely from "full" to "empty". On the other hand, if $\beta_r = 1$ then the status of p_{jr} will remain "full" after the firing of t_k. A similar binary representation of the restriction of the color markings CM^i of CN to $\{p'_{j1}, \ldots, p'_{js}\}$ can be defined (remember that we shall have $CM^i(p'_{jr}) = < c_f^{M^i(p_{jr})} >$ for each r, $1 \leq r \leq s$).

For each s-tuple of binary digits $\beta_v = (\beta_{v1}, \ldots, \beta_{vs})$ we shall introduce in T' a replacement transition t'_{kv} of t_k. For each r, $1 \leq r \leq s$, let:

$$I'(p'_{jr}, t'_{kv}) = \begin{cases} I(p_{jr}, t_k) & \text{if } \beta_{vr} = 0 \\ I(p_{jr}, t_k) + 1 & \text{if } \beta_{vr} = 1 \end{cases}$$

$$f^{\,n}_{jr\,kv} : \mathcal{P}(X) \to c_f \qquad \text{for } 1 \leq n \leq I'(p'_{jr}, t'_{kv})$$

$$O'(t'_{kv}, p'_{jr}) = 1$$

$$f^{\,1}_{kv\,jr} : \mathcal{P}(X) \times \mathcal{P}(X) \to c_e \qquad \text{if } \beta_{vr} = 0$$

$$f^{\,1}_{kv\,jr} : \mathcal{P}(X) \times \mathcal{P}(X) \to c_f \qquad \text{if } \beta_{vr} = 1$$

Obviously, we have introduced 2^s distinct replacement transitions t'_{kv}. We notice that the 2^s tuples β_v can be grouped into exactly $s + 1$ groups, all tuples in a group having the same number of digits 1. If β_v belongs to group q, $1 \leq q \leq s + 1$, and t'_{kv} is the replacement transition corresponding to β_v then:

i) t'_{kv} can be enabled in a color marking for which the binary representation of its restriction to $\{p'_{j1}, \ldots, p'_{js}\}$ belongs to some group r, $r > q$.

ii) t'_{kv} cannot be enabled in any color marking whose restriction to $\{p'_{j1}, \ldots, p'_{js}\}$ has a binary representation which is a tuple in groups 1, ..., q but different from β_v.

Nevertheless, in order to correctly implement the encoding rule *) given at the beginning of this proof, we want each replacement transition t'_{kv} to be firable only in those color markings whose restriction to $\{p'_{j1}, \ldots, p'_{js}\}$ has β_v as binary representation and in no other color markings. We shall achieve this by manipulating the priorities of these replacement transitions. Thus, all transitions t'_{kv} corresponding to tuples β_v belonging to the same group q, $1 \leq q < s + 1$, will have the same priority to fire in any color marking (in which they are enabled) and their priority will be strictly less than the priority of any enabled replacement transition corresponding to some tuple in the group $q + 1$. Let us therefore introduce the colors $c_{k1}, \ldots, c_{ks+1}$ in X and let us (temporarily) set $c_{kq} < c_{kq+1}$ for $1 \leq q < s + 1$. Next, we shall introduce a place p'_{ok} in P' on which each replacement transition t'_{kv} self-loops:

$$I'(p'_{ok}, t'_{kv}) = O'(t'_{kv}, p'_{ok}) = 1$$

$f_{ok\ kv}^{\ 1}$: $\mathcal{P}(X) \rightarrow c_{kq}$; $f_{kv\ ok}^{\ 1}$: $\mathcal{P}(X) \times \mathcal{P}(X) \rightarrow c_{kq}$

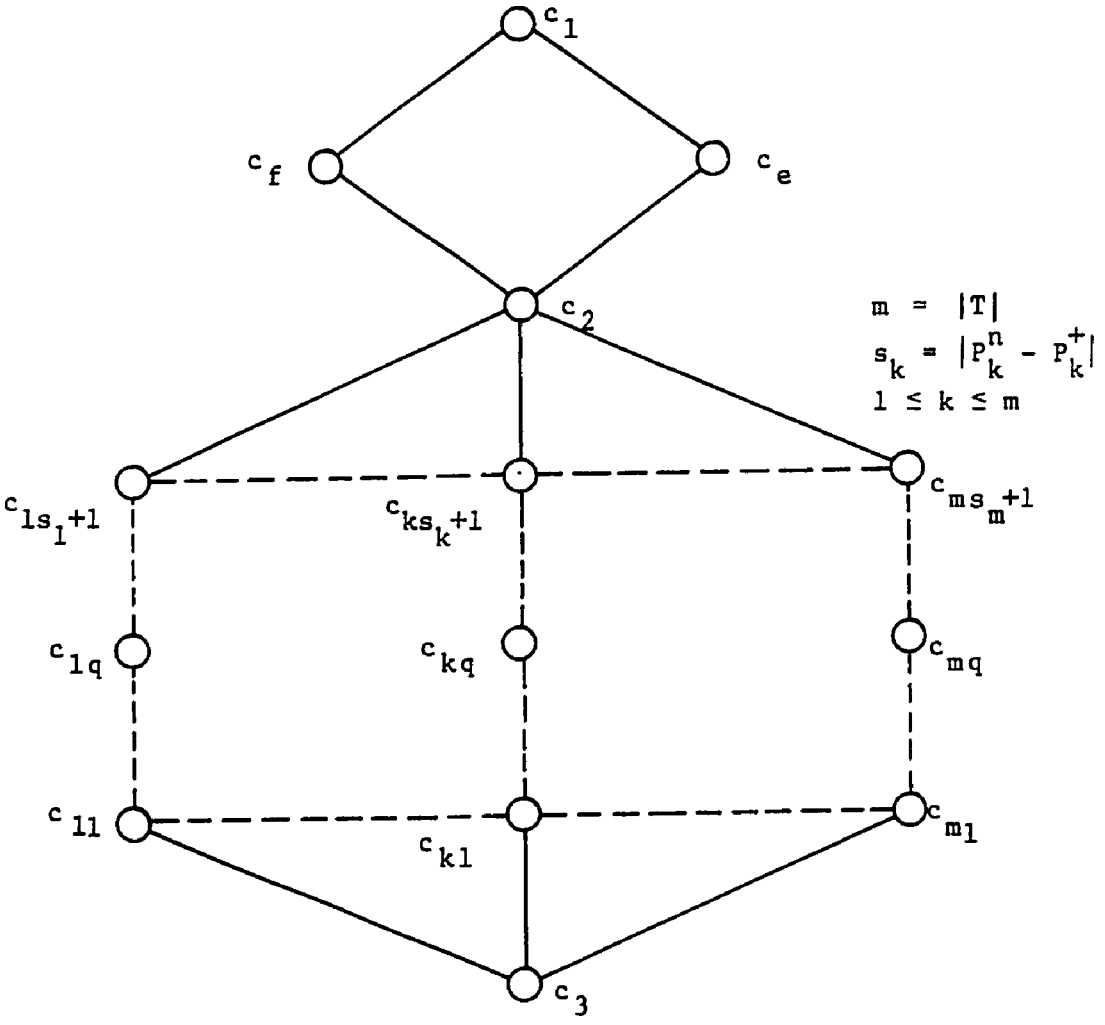
where q is the group of binary s-tuples to which β_v belongs. Finally, we shall introduce a new color c_1 in X and another place p'_{ko} in P', on which all replacement transitions t'_{kv} self-loop:

$$I'(p'_{ko}, t'_{kv}) = O'(t'_{kv}, p'_{ko}) = 1$$

$f_{ko\ kv}^{\ 1}$: $\mathcal{P}(X) \rightarrow c_1$; $f_{kv\ ko}^{\ 1}$: $\mathcal{P}(X) \times \mathcal{P}(X) \rightarrow c_1$

If $CM^o(p'_{ok}) = < c_{k1}, \ldots, c_{ks+1} >$ and $CM^o(p'_{ko}) = < c_1 >$ then these color bags of p'_{ok} and p'_{ko}, respectively will be preserved unaltered in any other reachable color marking of CN. As already mentioned, the sole purpose of the places p'_{ok} and p'_{ko} is to enforce the priority rule among the replacement transitions t'_{kv} discussed earlier. In any reachable color marking p'_{ko} creates a conflict for the (unique) token of color c_1 among all replacement transitions t'_{kv} enabled in that color marking. Therefore, in any reachable color marking all enabled transitions t'_{kv} are in the same conflict subcluster.

This construct is applied to all transitions t_k of EN for which $p_k^n - p_k^+ \neq \phi$. Note, however, that we want to enforce a priority hierarchy only among the replacement transitions introduced for the same transition t_k of the original net but not among replacement transitions introduced for different transitions of EN (remember that in EN any enabled transition can be selected to fire). Therefore, we shall impose the following partial ordering among the colors of the set X:


c_1
c_f
c_e
c_2
$m = |T|$
$s_k = |P_k^n - P_k^+|$
$1 \le k \le m$
c_{1s_1+1}
c_{ks_k+1}
c_{ms_m+1}
c_{1q}
c_{kq}
c_{mq}
c_{11}
c_{k1}
c_{m1}
c_3

The colors c_2 and c_3 were introduced in order to obtain a lattice, as required by the definition of the C-CPM. From the construct presented so far it follows that X will be a finite set. For each of the replacement transitions t'_{kv}, due to the particular structure of the lattice $C = (X, \leq)$ and to the corresponding color threshold functions, $g.\ell.b.(\mathcal{D}(\theta^i_{kv})) = g.\ell.b.(\{c_1, c_f, c_{kq}\}) = c_{kq}$ in any reachable color marking CM^i in which t'_{kv} is enabled, where q is the group of binary s-tuples to which the tuple β_v (corresponding to t'_{kv}) belongs.

In case $P^n_k - P^+_k = \phi$ for some transition t_k of EN, we shall introduce a single replacement transition, say t'_{k1}, in T' and one color, say c_{k1}, in X. The transition t'_{k1} will self-loop on the place p'_{ok} as described earlier but will not have any other input or output connections. The place p'_{ko} is certainly not needed any more.

In what follows, we shall "refine" the replacement transitions t'_{kv} introduced so far in order to simulate the firing of t_k with respect to the other categories of places to which t_k is connected.

Let $p_j \in P^+_k$ and suppose p'_j is the place of CN introduced for p_j. For each replacement transition t'_{kv} introduced at the previous step of our construct we shall in addition set:

$$I'(p'_j, t'_{kv}) = I(p_j, t_k);\ O'(t'_{kv}, p'_j) = O(t_k, p_j)$$

$$f_j{}^n{}_{kv}:\ \mathcal{P}(X) \to c_f \qquad \text{for } 1 \leq n \leq I'(p'_j, t'_{kv})$$

$$f_{kv}{}^n{}_j:\ \mathcal{P}(X) \times \mathcal{P}(X) \to c_f \qquad \text{for } 1 \leq n \leq O'(t'_{kv}, p'_j)$$

Suppose now that $p_j \in P^z_k - P^-_k$. For each replacement transition t'_{kv} we shall set:

$$I'(p'_j, t'_{kv}) = O'(t'_{kv}, p'_j) = 1$$

$$f_j{}^1{}_{kv}:\ \mathcal{P}(X) \to c_e;\ f_{kv}{}^1{}_j:\ \mathcal{P}(X) \times \mathcal{P}(X) \to c_e$$

Let now $p_j \in P^-_k$. Then, for each replacement transition t'_{kv} we shall have:

$$I'(p'_j, t'_{kv}) = 1;\ f_j{}^1{}_{kv}:\ \mathcal{P}(X) \to c_e$$

$$O'(t'_{kv}, p'_j) = O(t_k, p_j)$$

$$f_{kv}{}^{n}{}_{j}: \mathcal{P}(X) \times \mathcal{P}(X) \to c_f \text{ for } 1 \le n \le O'(t'_{kv}, p'_j).$$

Let us now consider the last category of places connected to t_k. Suppose first that there exists only one place p_j in $\mathcal{D}(O_k) - (P_k^+ \cup P_k^-)$. In order to simulate correctly the status change which the firing of t_k may cause for p_j, we shall replace each transition t'_{kv} by two copies, t'_{kv1} and t'_{kv2}, which will have exactly the same input and output connections as t'_{kv}. In addition:

$$I'(p'_j, t'_{kv1}) = I'(p'_j, t'_{kv2}) = 1$$

$$f_j{}^{1}{}_{kv1}: \mathcal{P}(X) \to c_f; \; f_j{}^{1}{}_{kv2}: \mathcal{P}(X) \to c_e$$

$$O'(t'_{kv1}, p'_j) = O(t_k, p_j) + 1; \; O'(t'_{kv2}, p'_j) = O(t_k, p_j)$$

$$f_{kv1}{}^{n}{}_{j}: \mathcal{P}(X) \times \mathcal{P}(X) \to c_f \text{ for } 1 \le n \le O'(t'_{kv1}, p'_j)$$

$$f_{kv2}{}^{n}{}_{j}: \mathcal{P}(X) \times \mathcal{P}(X) \to c_f \text{ for } 1 \le n \le O'(t'_{kv2}, p'_j)$$

Obviously, t'_{kv1} and t'_{kv2} cannot be simultaneously enabled in any reachable color marking of CN. If there are more than one places in $\mathcal{D}(O_k) - (P_k^+ \cup P_k^-)$ then the resulting transitions are further refined according to the above construct for each such place in turn.

We note that the refinements which we have presented above cannot change the priority of any replacement transition, in any reachable color marking of CN. If each replacement transition introduced for some transition t_k of the original EPN EN carries the same label as t_k then the resulting Labelled C-CPM CP_Σ will generate the same language as EN_Σ. □

Figure 4.4.5 illustrates a sample EPN. In the given marking only transition t_2 is enabled. Figure 4.4.6 displays the C-CPN obtained by applying the construct of Lemma 4.4.2 to the EPN of Figure 4.4.5. It is easy to see that both replacement transitions t'_{11} and t'_{12} introduced for the transition t_1 of the original EPN are disabled. On the other hand, both replacement transitions introduced for transition t_2 of the original net are enabled. Due to the conflict at the place p'_{20}, only t'_{22} is firable.

We notice that, in general, if the Labelled C-CPM CP_Σ produced by the construct of Lemma 4.4.2 is executed according to the execution

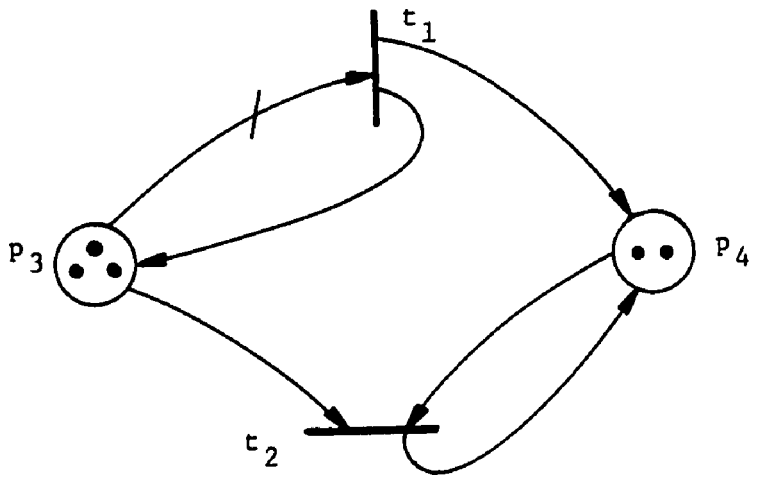


Figure 4.4.5

Sample EPN

rules of a C-CPM(I) (i.e., we use the relation R_E rather than R_{LE} for the selection of enabling colors) CP_Σ will perform the same firing sequences, and therefore generate the same label sequences, as before. Due to the particular structure of the color set $C = (X,\leq)$ and to the way in which the color threshold and color output functions of the particular transitions were defined, in any reachable color marking of CP_Σ there does not exist any "choice" in the selection of enabling colors. Thus, each transition is forced to select as enabling colors exactly the colors of its own (constant) color threshold functions which, in fact, is the rule governing the selection of enabling colors in a C-CPM(I). Consequently, we can state:

$$\Lambda(\text{EPM}) \subseteq \Lambda(\text{C-CPM(I)})$$

and

$$\Lambda_o(\text{EPM}) \subseteq \Lambda_o(\text{C-CPM(I)})$$

Combining this result with Lemma 4.4.1, Theorem 4.3.1 and Theorem 4.1.1(a) we can conclude:

Theorem 4.4.1

$\Lambda(\text{PPM}) = \Lambda(\text{C-CPM(I)})$ and $\Lambda_o(\text{PPM}) = \Lambda_o(\text{C-CPM(I)})$ □

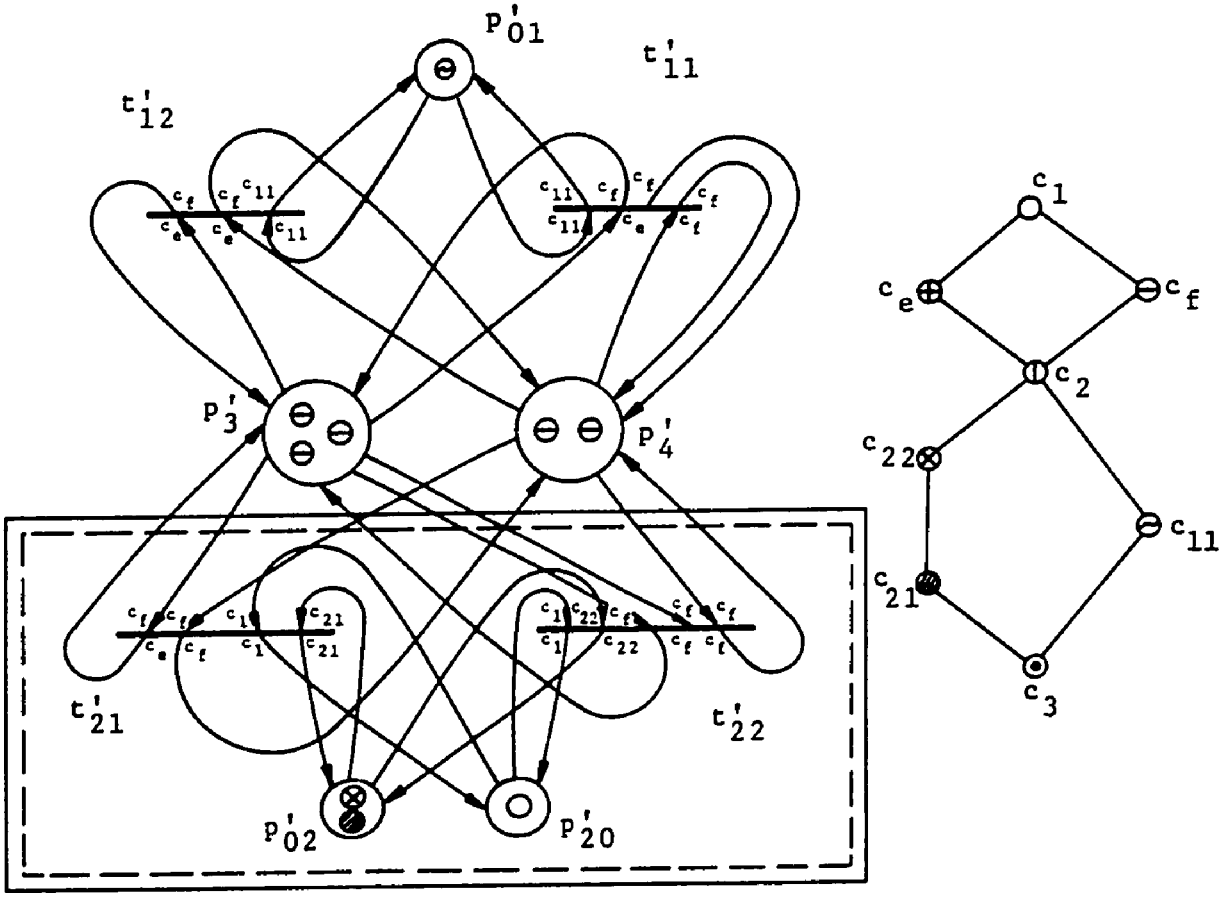


Figure 4.4.6

C-CPN Obtained from the EPN of Figure 4.4.5

Section 4.5 THE RELATIONSHIP BETWEEN THE C-CPM AND THE PPM

Based on the results presented in the previous sections of this chapter we can finally proceed now to investigate the relationship between the C-CPM and the PPM. First, let us however introduce the following composition operation on lattices we shall make use of later in this section.

It is known that any nonvoid, finite lattice $C = (X,\leq)$ has a least element (denoted by 0) and a greatest element (denoted by I) ([BIRK-67]). Even though we shall employ similar notations for the least and greatest elements of a lattice and for the output and input incidence functions of a GPN, respectively, the distinction will be clear from the context.

Let $C_1 = (X_1,\leq_1)$ and $C_2 = (X_2,\leq_2)$ be two arbitrary nonvoid, finite lattices such that $X_1 \cap X_2 = \phi$ (this condition can easily be met through appropriate renaming). We shall define the concatenation of C_1 and C_2, denoted by $C_1 \uparrow C_2$, to be the partially ordered set $C = (X,\leq)$ where:

1. $X = X_1 \cup X_2$
2. $\leq = \leq_1 \cup \leq_2 \cup \{(I_2, O_1)\}$

The concatenation of two (finite) lattices can be schematically represented in terms of their Hasse diagrams as in Figure 4.5.1.

It can easily be verified that $C = C_1 \uparrow C_2$ is a lattice itself and that $I = I_1$ and $O = O_2$, where I_1 denotes the greatest element of the lattice C_1 and O_2 denotes the least element of the lattice C_2. We notice that in general $C_1 \uparrow C_2 \neq C_2 \uparrow C_1$, i.e. the concatenation operation on lattices is not commutative. On the other hand, the concatenation operation $\uparrow$ can easily be seen to be associative, i.e. $(C_1 \uparrow C_2) \uparrow C_3 = C_1 \uparrow (C_2 \uparrow C_3) = C_1 \uparrow C_2 \uparrow C_3$ for any arbitrary finite lattices C_1, C_2 and C_3.

Lemma 4.5.1

$\Lambda(\text{C-CPM}) \leq \Lambda(\text{C-CPM(I)})$ and $\Lambda_o(\text{C-CPM}) \leq \Lambda_o(\text{C-CPM(I)})$

Proof: Consider an arbitrary Labelled C-CPM $CP_\Sigma = (CP, \Sigma, L)$ where $CP = (CN, CM^o)$, $CN = (N, C, F)$, $N = (T, P, I. O)$ is the underlying GPN and $C = (X,\leq)$ is the set of colors associated with CN. We shall first

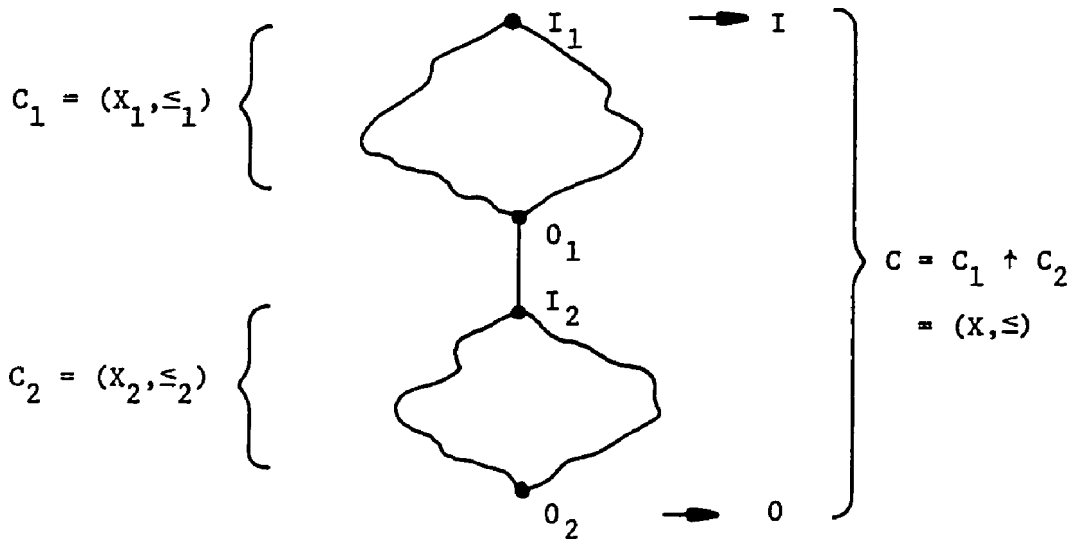


Figure 4.5.1
The Concatenation of Two Finite Lattices

construct a Labelled C-CPM(I) CP'_Σ, $=(CP', \Sigma', L')$ which generates through erasure the language of CP_Σ. Let $CP' = (CN', CM^{O'})$ where $CN' = (N', C_I, F')$ and $N' = (T', P', I', O')$.

We shall first build the set of colors $C_I = (X_I, \leq_I)$ of CN'. Consider the set of colors $C = (X, \leq)$ of CN. C is by definition a finite lattice. Let then $C' = (X', \leq')$ be a lattice isomorphic to C; hence, there exists an order preserving isomorphism $h : X \to X'$. Since C' is a finite lattice, there exists the dual of C', let us denote it by $C'' = (X', \leq'')$. Obviously, C'' must be a lattice too ([BIRK-67]). We note that the bijection $h : X \to X'$ is a dual isomorphism between C and C''. Thus, for each $x \in X$ and $y \in X$ we shall have $x \leq y$ if and only if $h(x) \leq' h(y)$ if and only if $h(y) \leq'' h(x)$.

Suppose now that A is the set of directed arcs associated with the GPN N. Let $A^i \subseteq A$ be the set of input arcs of the GPN N. Let h' be a bijection from A^i onto A', $A' \cap A^i = \phi$, defined by $h'(a^r_{jk}) = x^r_{jk}$, $x^r_{jk} \in A'$, for each arc $a^r_{jk} \in A^i$ and let X''' be the set.

$$X''' = A' \cup \{I''', O''', x\}$$

We shall define the following finite lattice $C''' = (X''', \leq''')$:

1. For each $a^r_{jk} \in A^i$ and $a^q_{mn} \in A^i$, $x^r_{jk} \not\leq''' x^q_{mn}$ and $x^q_{mn} \not\leq''' x^r_{jk}$, where $x^r_{jk} = h'(a^r_{jk})$ and $x^q_{mn} = h'(a^q_{mn})$.

2. For each arc $a^r_{jk} \in A^i$, $x^r_{jk} \not\leq''' x$ and $x \not\leq''' x^r_{jk}$, where $x^r_{jk} = h'(a^r_{jk})$.

3. I''' and O''' are the greatest and the least elements of C''', respectively.

Finally, let $C^{IV} = (\{c^+\}, \leq^{IV})$ and $C^V = (\{c^-\}, \leq^V)$ be two lattices, where the partial ordering relations $\leq^{IV}$ and $\leq^V$ are trivially defined on the singleton sets $\{c^+\}$ and $\{c^-\}$, respectively.

Then, $C_I = C^{IV} + C + C''' + C'' + C^V$ (Figure 4.5.2 displays the finite lattice C_I schematically).

Consider now a transition t_k of the original C-CPM CN and suppose $p_j \in \mathscr{D}(I_k)$. Let a^r_{jk} be an arbitrary input arc of t_k from p_j, $1 \leq r \leq I(p_j, t_k)$. Let us also assume that CN is in some color marking CM^i. We note that if $f^r_{jk} : \mathscr{P}(X) \to c^r_{jk}$ is the color threshold function associated with the arc a^r_{jk} then the color threshold imposed by f^r_{jk} for the selection of the enabling color is in fact independent of the color bag C^i_j. Moreover a color $c \in \mathscr{D}(C^i_j)$ can be selected as enabling color of t_k on the input arc a^r_{jk} only if $c \geq c^r_{jk}$. Let then

$$X^r_{jk} = \{c \mid c \in X \text{ and } c \geq c^r_{jk}\}.$$

The set X^r_{jk} contains all the nodes of the Hasse diagram of $C = (X, \leq)$ which lie in the paths between c^r_{jk} and I. We note that the Hasse diagram of C is an acyclic finite graph and thus there are only a finite number of distinct paths between any element $c^r_{jk} \in X$ and I. Since $c^r_{jk} \in X^r_{jk}$ we must have $g.\ell.b.(X^r_{jk}) = c^r_{jk}$.

Let us extend the bijection $h : X \to X'$ to a set. Thus, if $X_i \subseteq X$ then:

$$h(X_i) = \{h(c) \mid c \in X_i\}$$

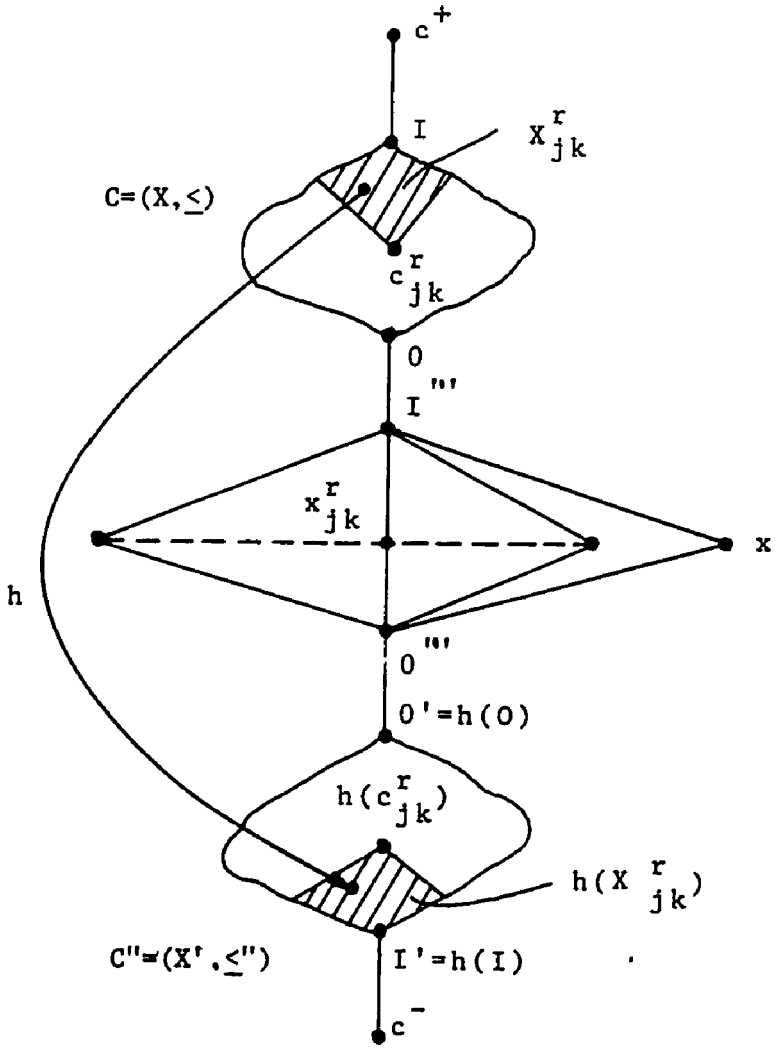


Figure 4.5.2

The Set of Colors $C_I = (X_I, \leq_I)$

Since h is the dual isomorphism between C and C'' we must have:

$$h(X^r_{jk}) = \{c' \mid c' \in X' \text{ and } c' \leq'' h(c^r_{jk})\}$$

and $\ell.u.b.(h(X^r_{jk})) = h(c^r_{jk})$ (see Figure 4.5.2).

Our construct will preserve the places of the original C-CPN CN. Thus, for each $p_j \in P$ we shall introduce a corresponding place p_j' in P'. Our construct will also ensure that the restrictions of the

reachable color markings of CP'_Σ, to the set of these places p'_j are exactly the set of reachable color markings of the original net CP_Σ. The initial and final color marking of the C-CPM(I) CP'_Σ, are defined accordingly.

Consider the input arc a^r_{jk} of the transition $t_k \in T$. For each color $c \in X^r_{jk}$ we shall introduce in T' a distinct transition t'_{rc}. These transitions will simulate the enabling color selection function of the arc a^r_{jk}. Therefore, we shall set:

$$I'(p'_j, t'_{rc}) = O'(t'_{rc}, p'_j) = 1$$

$$f_{j\,rc}^{\;1}: \mathscr{P}(X_I) \to c; \; f_{rc\,j}^{\;1}: \mathscr{P}(X_I) \times \mathscr{P}(X_I) \to c$$

Thus, all transitions t'_{rc} self-loop on the place p'_j. Since we are dealing with a C-CPM(I), in any color marking $CM^{i'}$ each t'_{rc} must select as enabling color from p'_j exactly c, i.e. the color for which t'_{rc} has been introduced.

Next we shall introduce a place p'_e in P' on which all transitions t'_{rc}, introduced for all input arcs a^r_{jk} of the original net, self-loop. We shall set:

$$f_{e\,rc}^{\;1}: \mathscr{P}(X_I) \to x^r_{jk}; \; f_{rc\,e}^{\;1}: \mathscr{P}(X_I) \times \mathscr{P}(X_I) \to x^{r+1}_{jk}$$

where $x^r_{jk} = h'(a^r_{jk})$. The presence of the colors x^r_{jk} and x^{r+1}_{jk} in the above functions implies an ordering of the input arcs of the original net which will be explained later on. We note that if p'_e contains a single token of color x^r_{jk} then at the most the transitions t'_{rc} introduced for the arc a^r_{jk} can be enabled and all such enabled transitions conflict at p'_e for the color x^r_{jk}. The role of the place p'_e will also be further discussed later on.

In addition, we shall make each transition t'_{rc} introduced above self-loop on a separate place of P', let us denote it by p'_c. Let:

$$f_{c\,rc}^{\;1}: \mathscr{P}(X_I) \to h(c); \; f_{rc\,c}^{\;1}: \mathscr{P}(X_I) \times \mathscr{P}(X_I) \to h(c).$$

In the initial color marking the place p'_c contains a single token, of color $h(c)$. It is easy to see that $\theta_{rc} = \langle c, x^r_{jk}, h(c) \rangle$ (remember that in a C-CPM(I) the bag of enabling colors of a transition does not depend on the current color marking of the respective net). Hence, g.l.b. $(\mathscr{D}(\theta_{rc})) = h(c)$ and thus the place p'_c determines the

priority of t'_{rc}. We can see now that in any color marking $CM^{i'}$, at most one transition t'_{rc} introduced for some arc a^r_{jk} can be fired (assuming that the place p'_e contains only one token, of color x^r_{jk}) namely the one which corresponds to a color c such that $c \in \mathcal{D}(C^{i'}_j)$, $c \geq c^r_{jk}$ in C and there exists no other color $c' \in \mathcal{D}(C^{i'}_j)$, $c > c' \geq c^r_{jk}$ in C. This is so because if there exists such a color c' in p'_j in the color marking $CM^{i'}$, then $h(c) <_I h(c') \leq_I h(c^r_{jk})$ and correspondingly the transition $t'_{rc'}$, introduced for c', is also enabled and has higher priority than t'_{rc}. Nevertheless, if there exist the distinct colors $c \in \mathcal{D}(C^{i'}_j)$ and $c' \in \mathcal{D}(C^{i'}_j)$ such that $c \not\geq c'$ and $c' \not\geq c$ in C then by the same argument both transitions t'_{rc} and $t'_{rc'}$ are enabled and have the same priority to fire in $CM^{i'}$. Thus, the transitions t'_{rc} truly simulate the enabling color selection function of the arc a^r_{jk}.

For each input arc a^r_{jk} of the original C-CPN we shall introduce a distinct place, let us denote it by p'_{jkr}, in P'. Whenever any of the transitions t'_{rc} introduced for a^r_{jk} fires it will:

1. put a token of the (selected enabling) color c in p'_{jkr}. The place p'_{jkr} thus "remembers" the enabling color selected by the transition t'_{rc}.

2. restore the token of color c in p'_j. Consequently, the process of selecting an enabling color for some other input arc is not impaired.

3. put a token of color x^{r+1}_{jk} in the place p'_e, thus disabling all transitions t'_{rc} introduced for the arc a^r_{jk} and, at the same time, enabling the transitions t'_{r+1c} introduced for some other arc a^{r+1}_{jk}. The selection of the particular color x^{r+1}_{jk} will be discussed in more detail later on.

The construct presented so far is exemplified in Figure 4.5.3. This construct will be performed for all input arcs a^r_{jk} of A.

We note that there exists the possibility that in the given color marking $CM^{i'}$, $C^{i'}_j \cap X^r_{jk} = \phi$, i.e. no color in $\mathcal{D}(C^{i'}_j)$ is eligible to become the enabling color on a^r_{jk}. In this situation none of the transitions t'_{rc} discussed above can be enabled. We shall introduce an additional transition t'_{rs} in T' (called the "service transition" corresponding to the arc a^r_{jk}). This transition is connected as shown

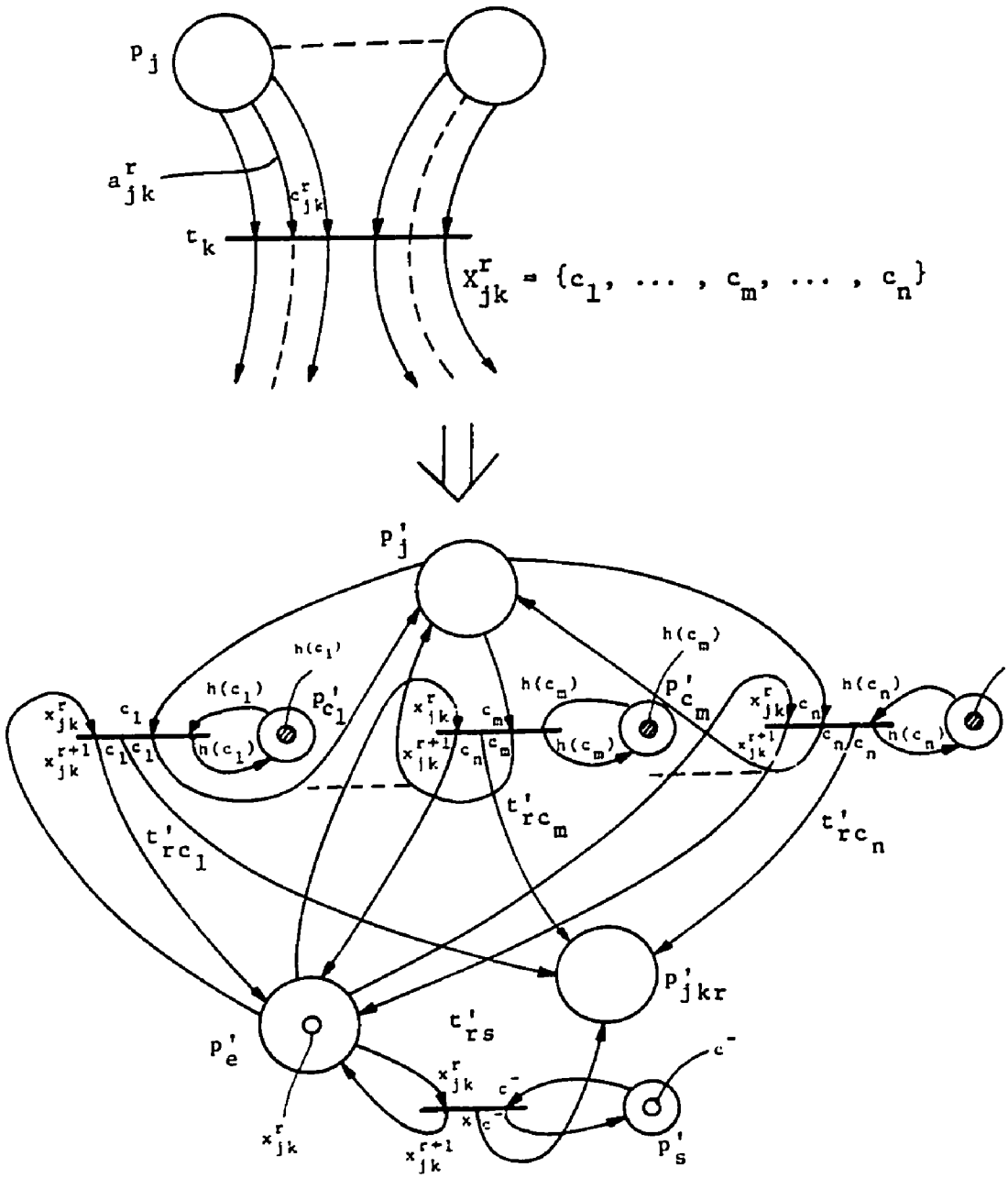


Figure 4.5.3

Simulation of the Enabling Color Selection Function of the Arc a^r_{jk}

in Figure 4.5.3. Let p'_s contain in the initial color marking of CN' only one token, of color c^-. Hence, in any reachable color marking $g.\ell.b.(\mathscr{D}(\theta_{rs})) = g.\ell.b.(\{x^r_{jk}, c^-\}) = c^-$. But $c^- \leq_I c$, for any color $c \in X_I$ and therefore t'_{rs} has lower priority than any enabled transition t'_{rc} introduced for the arc a^r_{jk}. Therefore, t'_{rs} can fire if and only if all the other transitions t'_{rc} introduced for the arc a^r_{jk} are disabled. If t'_{rs} fires it will put in p'_{jkr} a token of color x (we shall call x the "neutral color"). Thus p'_{jkr} will "remember" that there exists no enabling color on the arc a^r_{jk} in the given color marking.

Informally, we shall refer to the transitions t'_{rc} and t'_{rs}, and the places p'_c, p'_s and p'_{jkr} which we have introduced in order to simulate the enabling color selection function of the input arc a^r_{jk} as to the "enabling color selection subnet" corresponding to the arc a^r_{jk}.

We have seen that in any reachable color marking CM^i of the original C-CPN CN, the enabling color of a transition t_k of CN on some input arc a^r_{jk} must be selected from the set X^r_{jk}. At this point it is advantageous to represent the bag of enabling colors of t_k from the input place p_j, i.e. θ^i_{jk}, as an m_j-tuple of colors, where $m_j = I(p_j, t_k)$ and the r-th coordinate represents the enabling color selected by a^r_{jk}, $1 \leq r \leq I(p_j, t_k)$. Then, in any reachable color marking CM^i, we must have:

$$\theta^i_{jk} \in \bigtimes_{r=1}^{I(p_j, t_k)} X^r_{jk}$$

Consequently, the bag of enabling colors of t_k can be viewed as an n-tuple of m_j-tuples of colors, where $n = |\mathscr{D}(I_k)|$ and some total ordering has been imposed on the set of input places of t_k. Hence, in any reachable color marking CM^i :

$$\theta^i_k \in \bigtimes_{p_j \in \mathscr{D}(I_k)} \left[\bigtimes_{r=1}^{I(p_j, t_k)} X^r_{jk} \right]$$

Let

$$K_k = \left| \bigtimes_{p_j \in \mathcal{D}(I_k)} \left[\bigtimes_{r=1}^{I(p_j, t_k)} X_{jk}^r \right] \right|$$

Certainly K_k is a finite number. In the C-CPN(I) CN' we shall introduce in T' K_k replacement transitions for $t_k \in T$, which will simulate the firing of t_k (for different bags θ_k^i of enabling colors).

Let $\beta_v = (\beta_{v1}, \ldots, \beta_{vn})$ be one of the K_k combinations of colors from X mentioned above and suppose $\beta_{vj} = (c_{vj1}, \ldots, c_{vjm})$, for some j, $1 \leq j \leq n$, where $m = I(p_j, t_k)$. Let us denote by t'_{kv} the transition introduced in T' for the tuple β_v. We shall set for each place $p'_j \in P'$ corresponding to some place $p_j \in P$:

$$I'(p'_j, t'_{kv}) = I(p_j, t_k) = m; \; O'(t'_{kv}, p'_j) = O(t_k, p_j)$$

$$f_{j\,kv}^{\;r}: \mathcal{P}(X_I) \to c_{vjr} \quad \text{for} \quad 1 \leq r \leq m$$

$$f_{kv\;j}^{\;r}: \mathcal{P}(X_I) \times \mathcal{P}(X_I) \to c_{kj}^r$$

where, in CN,

$$f_{kj}^r: \mathcal{P}(X) \times \mathcal{P}(X) \to c_{kj}^r, \; c_{kj}^r \in X, \text{ for } 1 \leq r \leq O(t_k, p_j)$$

Consequently, $\theta_{jkv} = \beta_{vj}$. We remember that we have introduced for each input arc a_{jk}^r of t_k a distinct place p'_{jkr} in P'. Each transition t'_{kv} will self-loop on each such place p'_{jkr} and:

$$\left.\begin{aligned} f_{jkr\;kv}^{\;1}: \; & \mathcal{P}(X_I) \to c_{vjr} \\ f_{kv\;jkr}^{\;1}: \; & \mathcal{P}(X_I) \times \mathcal{P}(X_I) \to c_{vjr} \end{aligned}\right\} \text{for} \quad 1 \leq r \leq m$$

This part of the construct is exhibited in Figure 4.5.5 for the sample C-CPN of Figure 4.5.4.

Let us denote by θ'_{kv} the bag of enabling colors of t'_{kv} from all places p'_j such that p'_j is a place of P' corresponding to some place $p_j \in \mathcal{D}(I_k)$. Also, let θ''_{kv} be the bag of enabling colors of t'_{kv} from the places p'_{jkr} introduced in P' for the input arcs a_{jk}^r of the original transition t_k. By construct, we have $\theta'_{kv} = \theta''_{kv} = \beta_v$ and the

bag of enabling colors of t'_{kv} is $\theta_{kv} = \theta'_{kv} \cup \theta''_{kv}$ (bag union).

Since each replacement transition t'_{kv} has been introduced for a distinct combination of colors β_v and since we are dealing with a C-CPM(I), at most one transition t'_{kv} can be enabled in any reachable color marking of CN'. Let $CM^{i'}$ be a color marking of CN' and let p_j' be an input place of t'_{kv} corresponding to some input place p_j of the original transition t_k. Then, t'_{kv} is enabled in $CM^{i'}$ only if the bag of colors deposited by the corresponding enabling color selection subnets in the places p'_{jkr} corresponding to all input arcs of t_k from p_j is β_{vj} and β_{vj} is also contained in the color bag $C_j^{i'}$ of p'_j. This condition must be satisfied for all such places p'_j to which t'_{kv} is connected and the corresponding places p'_{jkr}. In particular, we notice that if some enabling color selection subnet could not select an enabling color, i.e. the corresponding place p'_{jkr} contains a token of the neutral color x, then none of the replacement transitions t'_{kv} can be enabled.

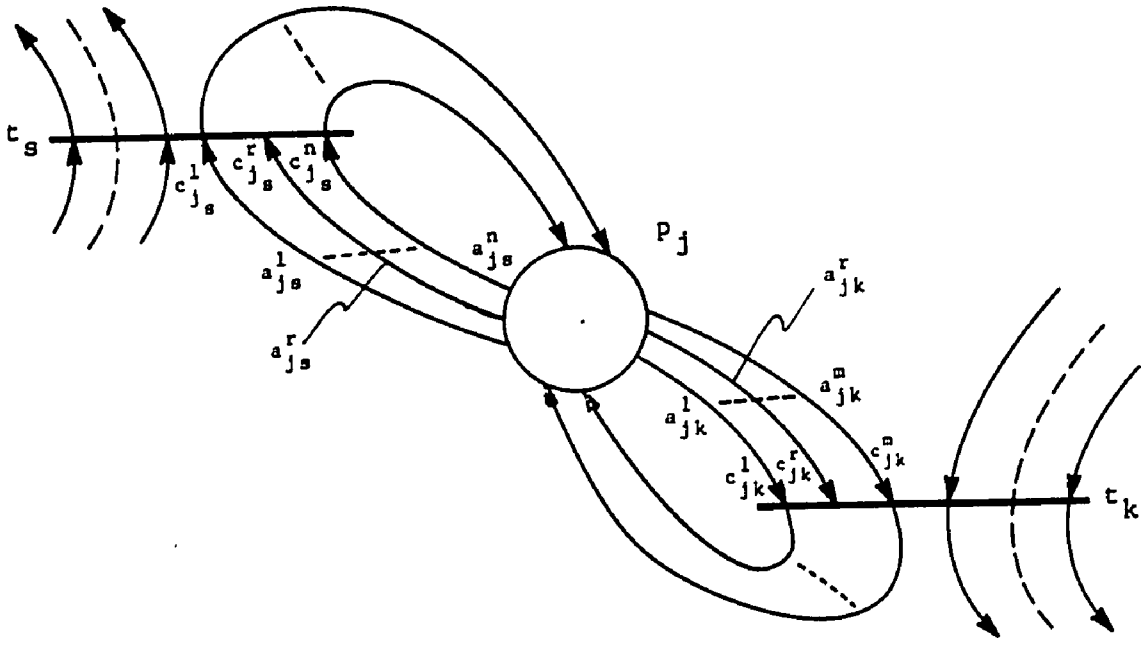
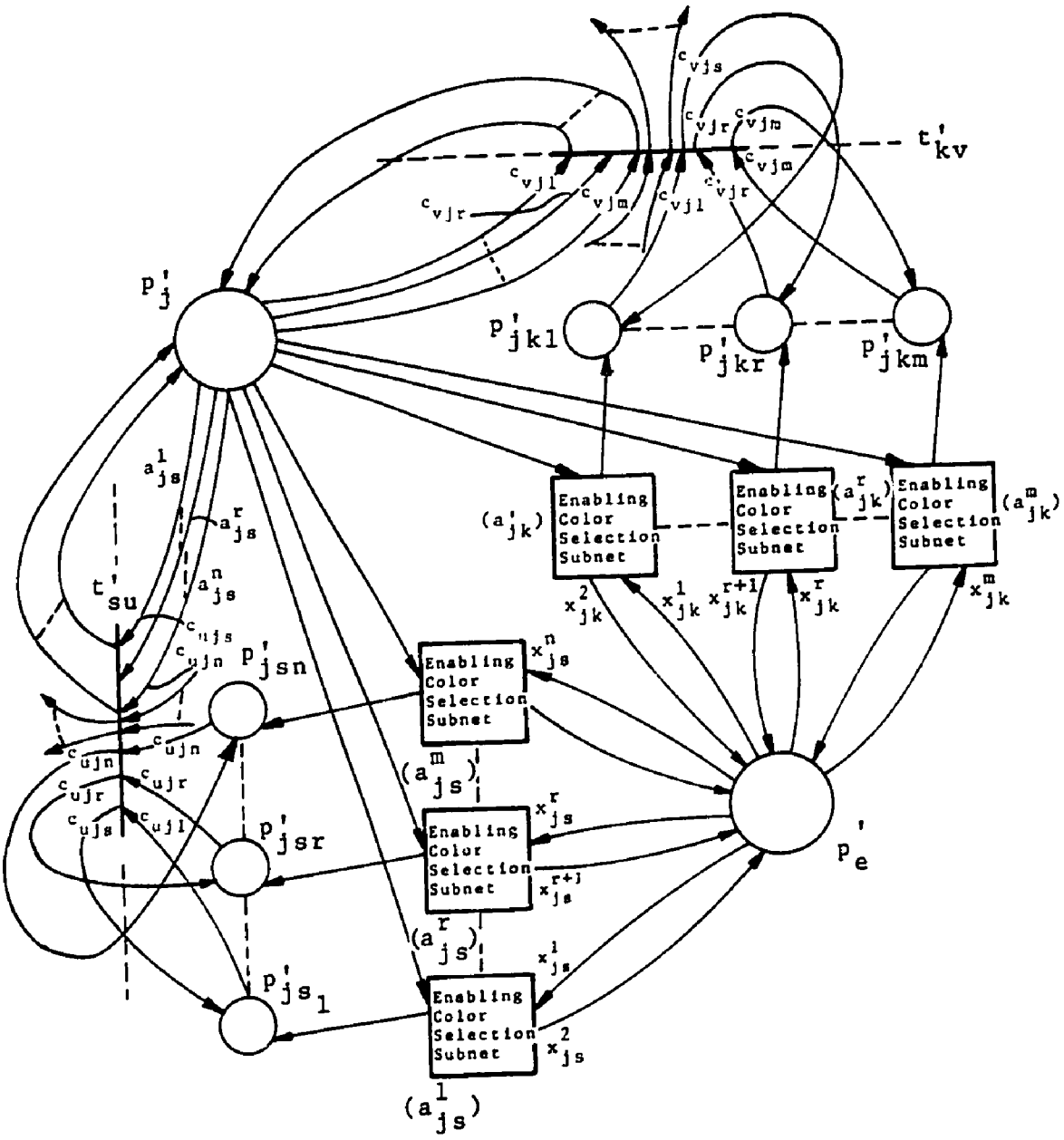


Figure 4.5.4
Sample C-CPN



$$\beta_{vj} = (c_{vj1}, \ldots, c_{vjr}, \ldots, c_{vjm})$$

$$\beta_{uj} = (c_{uj1}, \ldots, c_{ujr}, \ldots, c_{ujm})$$

$$c_{vjq} \geq x^q_{jk} \ , \ 1 \leq q \leq m$$

$$c_{ujq} \geq x^q_{js} \ , \ 1 \leq q \leq n$$

Figure 4.5.5

Sample Replacement Transitions

For each t'_{kv} we shall have g.ℓ.b.($\mathcal{D}(\theta_{kv})$) = g.ℓ.b.($\mathcal{D}(\theta'_{kv} \cup \theta''_{kv})$) = g.ℓ.b.($\mathcal{D}(\beta_v)$). By construction, the transition t'_{kv} can conflict with some other replacement transition t'_{sv} (introduced for some other transition t_s of the original net) only at some place $p'_j \in P'$ such that the corresponding place p_j of P is an element of $\mathcal{D}(I_k) \cap \mathcal{D}(I_s)$. Moreover the priority of t'_{kv} is the same as the priority of the original transition t_k if t_k would have selected β_v as its bag of enabling colors. Since our construct imposes that the set of reachable color markings of each place $p'_j \in P'$ is the same as the set of reachable color markings of the corresponding place $p_j \in P$, t'_{kv} is firable if and only if t_k with the bag of enabling colors β_v is firable. The output connections and the respective output color functions of each replacement transition t'_{kv} with respect to the places $p'_j \in P'$ are the same as those of the original transition t_k with respect to the corresponding places $p_j \in P$. Consequently, the firing of t'_{kv} produces the same effect as the firing of t_k with β_v as its bag of enabling colors.

In the sequel we shall refer to the set of replacement transitions t'_{kv} introduced for a transition t_k of the original net as to the "firing subnet" corresponding to the transition t_k.

Let us now put together the constructs presented so far and explain the mechanism for the simulation of the execution of the original C-CPM. The simulation of the firing of a transition of CP_Σ is performed in three phases.

Phase I In this phase, an enabling color (if any) is selected for each input arc a^r_{jk} of CN. This operation is implemented by the enabling color selection subnet corresponding to each such arc a^r_{jk}. We note that according to the execution rules of the CPM given in Chapter II, the enabling color has to be individually selected for each input arc a^r_{jk} of each transition $t_k \in T$, in particular, independently of the other input arcs of t_k from the same place p_j. Moreover, the selection procedure does not discriminate among tokens of the same color present in p_j in the respective color marking CM^i. Consequently, it is possible for two distinct input arcs a^r_{jk} and a^s_{jk}, where $1 \leq r, s \leq I(p_j, t_k)$, to select as enabling color from p_j the same color c when in fact $\#(c, C^i_j) = 1$. Evidently, the order in which the

particular input arcs select their enabling colors is not important (i.e. does not affect the particular selections). We can therefore define a total ordering on the set of arcs A^i of CN, for the purpose of selecting the enabling colors. If a^r_{jk} is the first arc to select its enabling color according to our ordering then we shall have p'_e contain a single token of color x^r_{jk} in the initial color marking $CM^{o'}$, where $x^r_{jk} = h'(a^r_{jk})$. As we have already explained, after its enabling color selection subnet in CN' has selected an enabling color it will put in p'_e a token of color x^{r+1}_{jk}, where $x^{r+1}_{jk} = h'(a^{r+1}_{jk})$ and a^{r+1}_{jk} is the arc following a^r_{jk} in the process of selecting the enabling colors, according to our ordering. The last enabling color selection subnet to "fire" will place in p'_e a token of color x thus disabling all enabling color selection subnets, including itself.

We shall introduce in T' a new transition, let it be t'_I, and we shall append to P' a distinct place p'_{okv} for each replacement transition t'_{kv} of the firing subnet corresponding to the transition t_k, for all $t_k \in T$. The transition t'_I will collect the token of color x from p'_e and will place a token of color c^+ in each place p'_{okv}. Thus:

$$I'(p'_e, t'_I) = 1 \; ; \; O'(t'_I, p'_e) = 0 \; ; \; f^1_{eI} : \mathscr{P}(X_I) \to x$$

$$I'(p'_{okv}, t'_I) = 0; \; O'(t'_I, p'_{okv}) = 1$$

$$f^1_{I\,okv} : \mathscr{P}(X_I) \times \mathscr{P}(X_I) \to c^+$$

Each replacement transition t'_{kv} will self-loop on the corresponding place p'_{okv} and:

$$f^1_{okv\,kv} : \mathscr{P}(X_I) \to c^+ \; ; \; f^1_{kv\,okv} : \mathscr{P}(X_I) \times \mathscr{P}(X_I) \to c^+$$

Clearly, after the last enabling color selection subnet has "fired", the only enabled transition is t'_I. By firing, t'_I disables itself (note that t'_I does not restore the token of color x in p'_e) and enables all the transitions t'_{kv} of the firing subnets introduced for the transitions $t_k \in T$ from the point of view of the corresponding places p'_{okv} (which were empty so far). Of course, the enabled status of such a replacement transition t'_{kv} depends also on the other conditions

mentioned earlier. Note that the place p'_{okv} does not affect the firability of the respective replacement transition t'_{kv} since $c^+_I \geq c$, for all $c \in X_I$ and moreover, t'_{kv} does not conflict at this stage with any other transition at p'_{okv}.

The firing of t'_I marks the end of Phase I and the beginning of Phase II. At this stage each place p'_{jkr} must contain a token, possibly of the neutral color x, as produced by the respective enabling color selection subnet.

Phase II In this phase, the firing of some transition t_k of the original C-CPN CN is simulated. One of the replacement transitions t'_{kv} belonging to some firing subnet is selected to fire according to the execution rules of the C-CPM(I) (note that at this stage no transitions other than those belonging to firing subnets can be enabled).

Since we want to fire exactly one such transition t'_{kv} at a time, we shall introduce in T' a new transition, t'_{II}, and a new place p'_{oII} in P'. Each replacement transition t'_{kv} of the firing subnet corresponding to transition t_k, for all $t_k \in T$, will deposit upon firing a token of color c^+ in p'_{oII}. On the other hand, t'_{II} will have as input places all the places p'_{okv} as well as p'_{oII}. Formally:

$$I'(p'_{oII}, t'_{II}) = 1 \; ; \; O'(t'_{II}, p'_{oII}) = 0 \; ; \; f_{oII\,1\,II} : \mathscr{P}(X_I) \to c^+$$

and for each place p'_{okv}:

$$I'(p'_{okv}, t'_{II}) = 1 \; ; \; O'(t'_{II}, p'_{okv}) = 0 \; ; \; f_{okv\,1\,II} : \mathscr{P}(X_I) \to c^+.$$

Also, for each replacement transition t'_{kv} we shall have:

$$I'(p'_{oII}, t'_{kv}) = 0 \; ; \; O'(t'_{kv}, p'_{oII}) = 1 \; ; \; f_{kv\,1\,oII} : \mathscr{P}(X_I) \times \mathscr{P}(X_I) \to c^+$$

Thus, after any replacement transition t'_{kv} has fired, t'_{II} becomes enabled (p'_{oII} was empty so far) and has higher priority than any other enabled transition. This is so because c^+ is the enabling color of t'_{II} from each of its input places. Moreover, t'_{II} conflicts with all enabled transitions t'_{kv} at the corresponding input places p'_{okv} for the color c^+. Hence, the firing of any replacement transition t'_{kv} is necessarily followed (in case of a language from the Λ_o family) by the firing of t'_{II}. By firing, t'_{II} disables all the replacement

transitions t'_{kv} (t'_{II} consumes the token of color c^+ from the input place p'_{okv} of each transition t'_{kv}) and also disables itself. We shall introduce yet another place, denoted by p'_{oc}, in P' and we shall set:

$$I'(p'_{oc}, t'_{II}) = 0 \; ; \; O'(t'_{II}, p'_{oc}) = 1 \; ; \; f_{II\,oc}^{\;1} : \mathcal{P}(X_I) \times \mathcal{P}(X_I) \rightarrow x^r_{jk}$$

where $x^r_{jk} = h'(a^r_{jk})$ and a^r_{jk} is the first arc of CN to select its enabling color according to our ordering of the arcs in A^i. The firing of t'_{II} marks the end of Phase II and the beginning of Phase III.

Phase III The purpose of Phase III is to "clean up" the net, that is to empty the places p'_{jkr} corresponding to each enabling color selection subnet.

Consider the enabling color selection subnet introduced for some arc $a^r_{jk} \in A^i$. We shall introduce q transitions in T', where $q = |X^r_{jk}| + 1$. Each such transition t'_c will have:

$$I'(p'_{jkr}, t'_c) = 1 \; ; \; O'(t'_c, p'_{jkr}) = 0 \; ; \; f_{jkr\,c}^{\;1} : \mathcal{P}(X_I) \rightarrow c$$

where $c \in X^r_{jk} \cup \{x\}$ is the color for which t'_c has been introduced. Also, the transition t'_c self-loops on p'_{oc} and:

$$f_{oc\,c}^{\;1} : \mathcal{P}(X_I) \rightarrow x^r_{jk} \; ; \; f_{c\,oc}^{\;1} : \mathcal{P}(X_I) \times \mathcal{P}(X_I) \rightarrow x^{r+1}_{jk}$$

where $x^{r+1}_{jk} = h'(a^{r+1}_{jk})$, $x^r_{jk} = h'(a^r_{jk})$ and a^{r+1}_{jk} is the arc of A^i following a^r_{jk} in our ordering of the arcs in A^i.

This construct is performed for each distinct enabling color selection subnet contained in CN', i.e., for each arc $a^r_{jk} \in A^i$. We shall refer to the set of transitions t'_c introduced for some enabling color selection subnet as to a "clean-up subnet". Figure 4.5.6 displays a sample clean-up subnet.

We remember that after Phase I each place p'_{jkr} contains a single token, possibly of color x. Phase II preserves these tokens. In Phase III, for each place p'_{jkr}, only one transition t'_c of the corresponding clean-up subnet can be enabled, namely the one corresponding to the color c of the token in p'_{jkr}. By firing, t'_c empties the respective place p'_{jkr} and also enables the transitions of the clean-up subnet corresponding to the "next" place p'_{jkr+1}. The place p'_{oc} plays the same role as p'_e in this respect. The transitions of the "last" clean-up subnet are slightly different in the sense that they

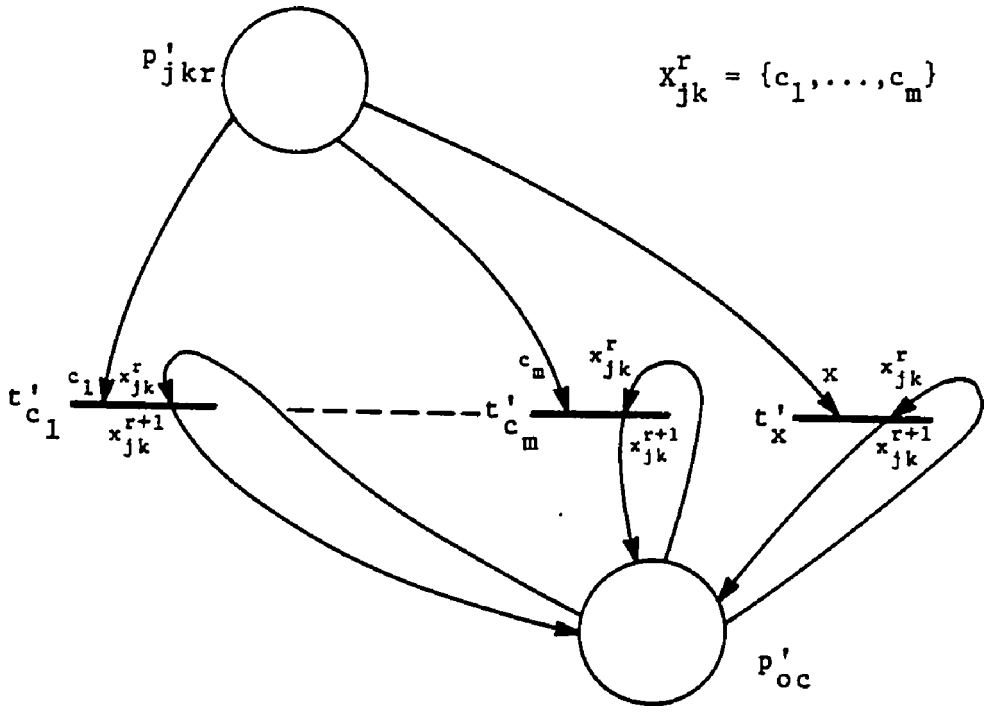


Figure 4.5.6

Sample Clean-Up Subnet

will place a token in p'_e rather than p'_{oc}. This token will enable the transitions of the enabling color selection subnet corresponding to the "first" arc a^r_{jk} in our ordering. Thus, Phase III terminates and Phase I is restarted. Figure 4.5.7 depicts schematically the simulation of the firing of a transition of CN by CN'.

It can be verified that the construct of this proof works both in the case when CP_Σ generates a language in the family Λ and when the language generated by CP_Σ belongs to the Λ_o family. The initial and the final color marking of $CP'_{\Sigma'}$ can be derived from the above discussion. Let us now define the Labelling function L' of $CP'_{\Sigma'}$.

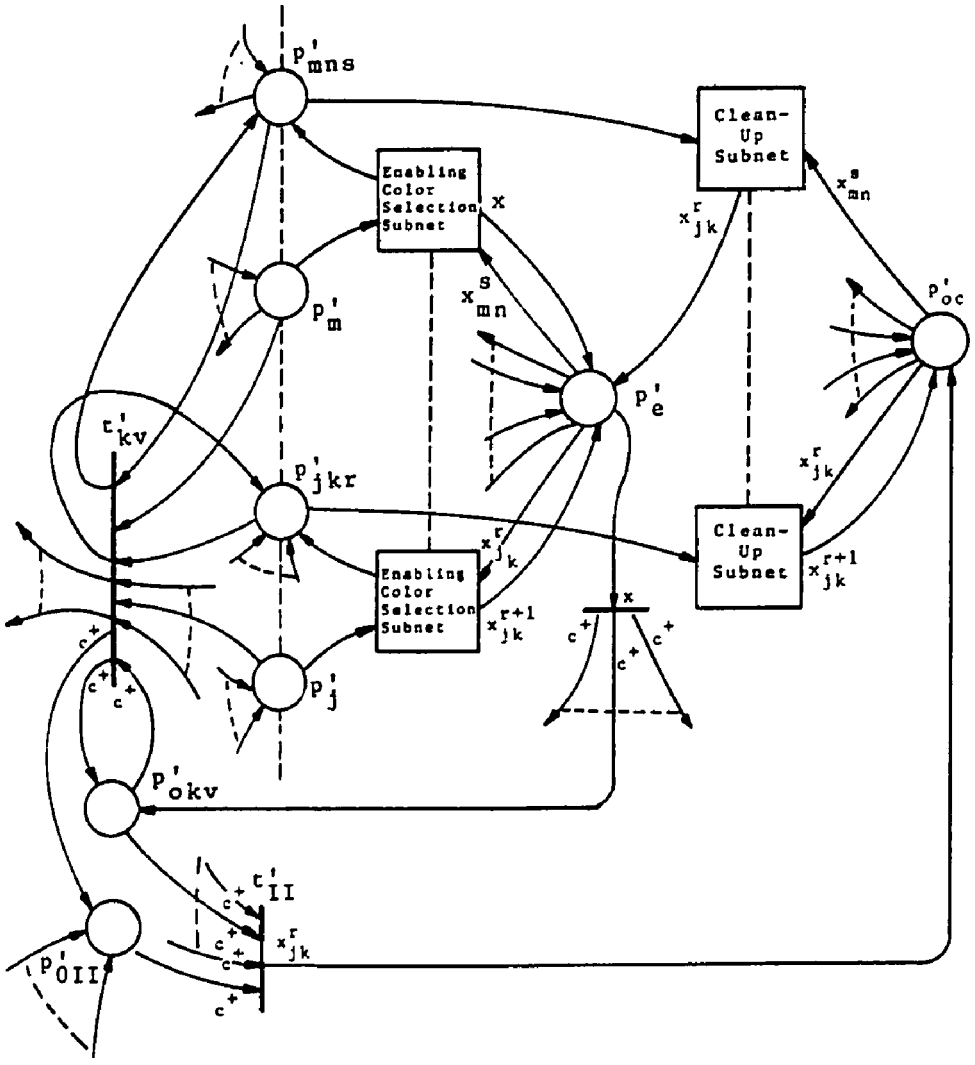


Figure 4.5.7

Simulation of the Firing of a Transition by CN'

Let $\Sigma' = \Sigma \cup \{d\}$ where $d \notin \Sigma$. The transitions contained in all enabling color selection subnets, t'_I , t'_{II} and the transitions contained in all clean-up subnets will receive the label d. All replacement transitions t'_{kv} belonging to the firing subnet corresponding to transition t_k will receive the label $L(t_k)$, for all $t_k \in T$. Obviously, the C-CPM(I) which we have built generates a language over $(\Sigma \cup \{d\})^*$, let us denote it by W'. In Phase I, we fire one transition from each enabling color selection subnet followed by one firing of t'_I, thus $|A^i| + 1$ firings of transitions labelled d. In Phase II we fire exactly one transition t'_{kv} (whose label is in Σ) followed by a firing of t'_{II}, thus only one synbol d can be generated. In Phase III, we fire exactly one transition from each clean-up subnet, thus a total of $|A^i|$ symbols d are generated during Phase III. Evidently, between the firing of two transitions labelled by symbols from Σ we generate (at the most) $2 \cdot |A^i| + 2$ symbols d (depending on whether W' is in Λ or Λ_o).

Let $W \in \Sigma^*$ be the language generated by the original Labelled C-CPM CP_Σ and let $h'' : (\Sigma \cup \{d\})^* \to \Sigma^*$ be a homomorphism defined by $h(d) = \lambda$ and $h(a) = a$, for each $a \in \Sigma$. Obviously $h''(W') = W$ and h is a k-limited erasing on W', where $k = 2 \cdot |A^i| + 2$.

By Theorems 4.4.1, 4.4.1(a) and 4.2.2 the lemma follows.

□

By Lemma 4.5.1 and Theorem 4.4.1 we have:

$$\Lambda(\text{C-CPM}) \subseteq \Lambda(\text{PPM})$$

and

$$\Lambda_o(\text{C-CPM}) \subseteq \Lambda_o(\text{PPM})$$

From Lemma 4.4.2 and Theorem 4.1.1(a) follows that the above containment relations hold true in opposite direction also. Hence, we can state:

Theorem 4.5.1

$\Lambda(\text{C-CPM}) = \Lambda(\text{PPM})$ and $\Lambda_o(\text{C-CPM}) = \Lambda_o(\text{PPM})$.

□

Some thought will show that part b) of Theorem 4.1.1 is true in the case of the C-CPM as well. We can therefore state:

$$H_\lambda(\Lambda_o(\text{C-CPM})) = \mathcal{RE}.$$

Evidently, the same statement also applies to the other formal models we have defined so far, i.e. the PPM(I), the EPM(I) and the C-CPM(I).

The above construct can be extended for a more general class of the CPM. Let $CP_\Sigma = (CP, \Sigma, L)$ be an arbitrary Labelled C-CPM where $CP = (CN, CM^0)$, $CN = (N, C, F)$, $N = (T, P, I, O)$ is the underlying GPN and $C = (X, \leq)$ is the set of colors associated with N. Let A be the set of arcs of N. Suppose the definition of the C-CPN has been extended in order to accept the following type of color threshold and color output functions:

- if $a^r_{jk} \in A$ is an input arc and $F(a^r_{jk}) = f^r_{jk}$ then $f^r_{jk}: \mathcal{P}(X) \to X$ and in any color marking CM^i of CN, the color threshold for the selection of the enabling color on the arc a^r_{jk} is $f^r_{jk}(\mathcal{D}(C^i_j))$.

- if $a^r_{ks} \in A$ is an output arc and $F(a^r_{ks}) = f^r_{ks}$ then $f^r_{ks}: \mathcal{P}(X) \times \mathcal{P}(X) \to X$ and in any color marking CM^i, the color of the token produced on the arc a^r_{ks} in case t_k fires in CM^i is $f^r_{ks}(\mathcal{D}(\theta^i_k))$.

We are interested to know to what extent are the language families Λ(C-CPM) and Λ_o(C-CPM) affected by this modification.

We shall construct, as in Lemma 4.5.1, a Labelled C-CPM(I) $CP'_{\Sigma'} = (CP', \Sigma', L')$ which will simulate the execution of CP_Σ. Let $CP' = (CN', CM^{0'})$ where $CN' = (N', C_I, F')$, $N' = (T', P', I', O')$ and $C_I = (X_I, \leq_I)$.

We shall first define the set of colors C_I. Let $X^{VI} = \{y_0, \ldots, y_{|X|}\}$ and suppose h''' is a mapping from $\mathcal{P}(X)$ onto X^{VI} defined by $h'''(\beta) = y_{|\beta|}$ for each set $\beta \in \mathcal{P}(X)$. We shall define the finite lattice $C^{VI} = (X^{VI}, \leq^{VI})$ such that $\leq^{VI}$ is a total ordering on the set X^{VI}, i.e. $y_r <^{VI} y_{r+1}$ for each r, $0 \leq r < |X|$.

Suppose now that V is a set such that there exists a bijection $h^{IV}: \mathcal{P}(X) \to V$ and $V \cap \mathcal{P}(X) = \phi$. Let then $C^{VII} = (X^{VII}, \leq^{VII})$ be the finite lattice defined by:

1. $X^{VII} = V \cup \{I^{VII}, O^{VII}\}$; $I^{VII} \notin V$ and $O^{VII} \notin V$.
2. For each $v_1 \in V$ and $v_2 \in V$, $v_1 \not\leq^{VII} v_2$ and $v_1 \not\geq^{VII} v_2$.
3. I^{VII} and O^{VII} are the greatest and the least element of C^{VII}, respectively.

Consider now the set P of places of CN. Let p be the subset of P defined by:

$$p = \{p_j \mid I(p_j, t_k) > 0 \text{ for some } t_k \in T\}$$

Suppose V' is a set such that $V' \cap p = \phi$ and there exists a bijection $h^V : p \to V'$. Let then $C^{VIII} = (X^{VIII}, \leq^{VIII})$ be the finite lattice defined by:

1. $X^{VIII} = V' \cup \{I^{VIII}, O^{VIII}\}$; $I^{VIII} \notin V'$ and $O^{VIII} \notin V'$.

2. For each $v_1' \in V'$ and $v_2' \in V'$, v_1' and v_2' are incomparable under the relation $\leq^{VIII}$.

3. I^{VIII} and O^{VIII} are the greatest and the least element of C^{VIII}, respectively.

Then $C_I = C^{IV} \uparrow C^{VII} \uparrow C \uparrow C^{VIII} \uparrow C^{VI} \uparrow C''' \uparrow C'' \uparrow C^V$ where C, C'', C''', C^{IV} and C^V have the same meaning as in Lemma 4.5.1.

Our construct will preserve the places of the original C-CPN CN. Thus, for each $p_j \in P$ we shall introduce a place p_j' in P'. Our construct will also ensure that the restrictions of the reachable color markings of $CP'_{\Sigma'}$ to the set of these places p_j' are exactly the set of reachable color markings of the original net. The initial and final color marking of $CP'_{\Sigma'}$ are defined accordingly. For each place $p_j \in p$ we shall introduce a distinct place p'_{oj} in P'. The role of the places p'_{oj} will be explained later on.

Next, to each place $p_j' \in P'$ corresponding to some place $p_j \in p$ we shall attach in T' a separate group of transitions which have the task to "detect" in any color marking $CM^{i'}$, what particular combination of colors from X forms $\mathcal{D}(C_j^{i'})$. Since $\mathcal{D}(C_j^{i'}) \in \mathcal{P}(X)$ we shall introduce in T' one transition t'_{jv} for each set $\beta_v \in \mathcal{P}(X)$. In order to simplify notations, suppose $\beta_v = \{c_{v1}, \ldots, c_{vm}\}$. Hence, we shall set:

$$I'(p_j', t'_{jv}) = O'(t'_{jv}, p_j') = |\beta_v|$$

$$\left.\begin{aligned} f_{j\ jv}^{\ r} &: \mathcal{P}(X_I) \to c_{vr} \\ f_{jv\ j}^{\ r} &: \mathcal{P}(X_I) \times \mathcal{P}(X_I) \to c_{vr} \end{aligned}\right\} \quad 1 \leq r \leq m$$

Thus, the bag of enabling colors of t'_{jv} from p_j' is always β_v (remember that we are building a C-CPM(I)). We also have $\alpha_{jv\ j} = \beta_v$ and consequently the firing of t'_{jv} does not alter the bag of colors in p_j'.

Next, we shall introduce a place p'_p in P' on which all transitions t'_{jv} self-loop. Let:

$$f_{p\,jv}^{\,s}: \mathcal{P}(X_I) \to v'_j;\ f_{jv\,p}^{\,1}: \mathcal{P}(X_I) \times \mathcal{P}(X_I) \to v'_{j+1}$$

where $v'_j = h^V(p_j)$ and $v'_{j+1} = h^V(p_{j+1})$ are colors from the set X^{VIII}. The place p'_p plays a role similar to that of p'_e or p'_{oc} in the construct of Lemma 4.5.1. The particular selection of the colors v'_j and v'_{j+1} will be explained later on.

Let us assume now that in the given color marking $\mathcal{D}(C_j^{i'}) = \beta_v$. Consequently, t'_{jv} and all the transitions t'_{ju} introduced for sets $\beta_u \in \mathcal{P}(X)$ such that $\beta_u \subset \beta_v$ are enabled. Nevertheless, we want to have only the transition t'_{jv}, i.e. the transition introduced for the particular set of colors β_v, firable in order to properly "detect" the set $\mathcal{D}(C_j^{i'})$. We shall achieve this by manipulating the priorities of the transitions t'_{jv}. For this purpose we shall have each transition t'_{jv} self-loop on a separate place $p'_{ojv} \in P'$ and we shall set

$$f_{ojv\,jv}^{\,1} : \mathcal{P}(X_I) \to h'''(\beta_v)$$

$$f_{jv\,ojv}^{\,1} : \mathcal{P}(X_I) \times \mathcal{P}(X_I) \to h'''(\beta_v)$$

where $h'''(\beta_v) = y_{|\beta_v|}$ is a color of X^{VI}. In the initial color marking $CM^{o'}$ we shall put in each such place p'_{ojv} a token of color $h'''(\beta_v)$. It is easy to see that the bag of enabling colors of t'_{jv} is $\theta_{jv} = \beta_v \cup \langle v'_j \rangle \cup \langle y_{|\beta_v|} \rangle$ and thus $g.\ell.b.(\mathcal{D}(\theta_{jv})) = y_{|\beta_v|}$.

But, if $\beta_u \subset \beta_v$ then $|\beta_u| < |\beta_v|$ and therefore $y_{|\beta_u|} <_I y_{|\beta_v|}$. Consequently, assuming that the place p'_p contains a single token, of color v'_j, only t'_{jv} is firable.

We shall make each transition t'_{jv} deposit upon firing a token of color $h^{IV}(\beta_v)$ in the place p'_{oj}. Thus, p'_{oj} "remembers" that in the current color marking the place p'_j has the set β_v as domain of its bag of colors.

Informally, we shall refer to the set of transitions t'_{jv} and the place p'_{oj} as to the "detection subnet" corresponding to the place p_j. Figure 4.5.8 exhibits a sample detection subnet.

$$\beta_v = \{c_{v1}, \ldots, c_{vm}\}$$

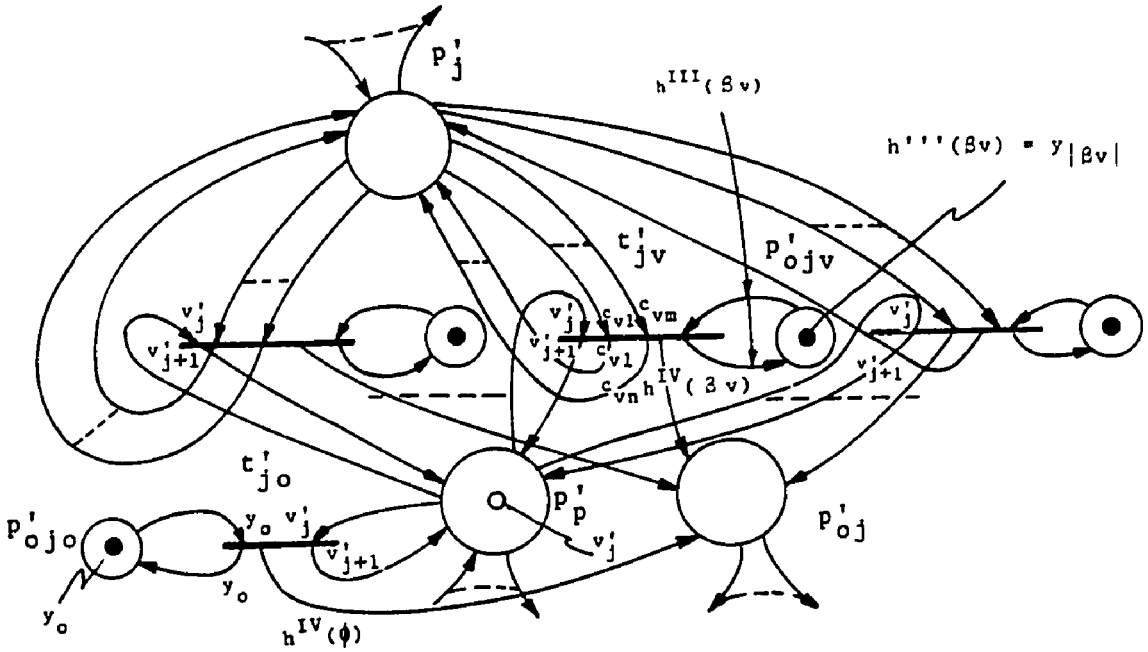


Figure 4.5.8

Sample Detection Subnet

Note that it is possible to have in the given color marking $CM^{i'}$, $\mathcal{D}(c_j^{i'}) = \phi$. Let t'_{jo} be the transition introduced for this situation. We notice that $y_o <_I y_s$, for all $y_s \in X^{VI}$ and thus t'_{jo} is firable if and only if all the other transitions t'_{jv} of the respective detection subnet are disabled.

We shall introduce in CN' a separate detection subnet for each distinct place $p_j \in P$.

The remainder of the construct of CP'_Σ, follows rather closely the mainline of the construct of Lemma 4.5.1. In what follows, we shall sketch the construct of CP'_Σ, without going into details, which are tedious though straightforward.

Suppose c_r is a color of X and let $X_r = \{c \mid c \in X \text{ and } c \geq c_r\}$. Consider an input arc $a^r_{jk} \in A^i$ and let f^r_{jk} be the color threshold function associated with a^r_{jk}. For each color v_u in the set V, i.e. for each distinct set $\beta_u \in \mathcal{P}(X)$, we shall introduce in CN' a separate enabling color selection subnet corresponding to a^r_{jk}. Each such enabling color selection subnet is designed for the particular set X_r corresponding to $c_r = f^r_{jk}(\beta_u)$, where $h^{IV}(\beta_u) = v_u$. Which enabling color selection subnet actually "selects" the enabling color for a^r_{jk} in some given color marking will depend on the color of the token in the respective place p'_{oj}. This can easily be achieved by having all the transitions from each of these enabling color selection subnets self-loop on p'_{oj} and assigning appropriate color threshold functions to the input arcs from p'_{oj}. We notice that due to the particular design of the set of colors C_I, the color of the token in the place p'_{oj} cannot affect the priority of any such transition.

This construct is repeated for all input arcs of the original C-CPN CN. We have to emphasize that throughout our construct we assume all color threshold and color output functions to be total.

Let K_k denote the number of distinct replacement transitions t'_{kv} of $t_k \in T$, as determined by the method of Lemma 4.5.1 but taking into consideration that in any reachable color marking CM^i

$$\theta^i_{jk} \in \bigtimes_{r=1}^{I(p_j,t_k)} \left[\bigcup_{c_s \in R^r_{jk}} X_s \right]$$

where $R^r_{jk} \subseteq X$ denotes the range of the color threshold function f^r_{jk}. We remember that each such transition t'_{kv} is introduced in T' for some distinct tuple of colors β_v such that $\mathcal{D}(\theta_{kv}) = \mathcal{D}(\beta_v)$.

Consider now an output arc a^r_{ks} of t_k and suppose f^r_{ks} is the associated color output function. As we have stated, the color of the token produced by t_k on the arc a^r_{ks} in case t_k fires in some color marking CM^i is $f^r_{ks}(\mathcal{D}(\theta^i_k))$. Hence, in our construct all we have to do is to modify the color output functions of the replacement transitions t'_{kv} accordingly. Thus, we shall set for all output arcs of t'_{kv} to p'_s, for all places $p'_s \in \mathcal{D}(O_{kv})$ corresponding to output

places p_s of t_k,

$$f^{r}_{kv\,s} : \mathcal{P}(X_I) \times \mathcal{P}(X_I) \to f^{r}_{ks}(\mathcal{D}(\beta_v))$$

for each r, $1 \leq r \leq O(t_k, p_s) = O'(t'_{kv}, p'_s)$ (β_v is the tuple of colors for which the replacement transition t'_{kv} has been introduced in T').

Let us now briefly examine the mechanism for the simulation of the execution of CP by the C-CPM(I) CP'. As in Lemma 4.5.1 the simulation of the firing of a transition of CP will be performed in three phases.

Phase I starts with a process of "detection" of the domains of the bags of colors for the places $p_j \in p$. Suppose $p = \{p_1, p_2, \ldots, p_m\}$, that is we have imposed a total ordering on the places of p such that p_1 will be "detected" first while p_m will be last. Consequently, we shall put in the initial color marking $CM^{o'}$ a token of color $v'_1 \in X^{VIII}$, where $v'_1 = h^V(p_1)$, in p'_p. Thus, the detection subnet corresponding to p'_1 will be enabled while all other detection subnets will be disabled. After the detection subnet corresponding to p'_1 has "fired" it will "shift" the control to the next detection subnet by placing in p'_p a token of color v'_2, where $v'_2 = h^V(p_2)$ and p_2 is the place of p following p_1 in our ordering. The process continues in this manner until the detection subnet corresponding to p'_m becomes enabled. This detection subnet will act slightly different, in the sense that it will deposit a token in p'_e rather than p'_p. In this way the detection process is terminated (all detection subnets are now disabled) and the enabling color selection subnet corresponding to the first input arc a^r_{jk} to select its enabling color is activated.

At this stage, each place p'_{oj} contains a token whose color corresponds to the domain of the bag of colors of p'_j in the respective color marking.

Phase I continues now as in Lemma 4.5.1 the only exception consisting in the fact that an input arc will have several enabling color selection subnets corresponding to it. Nevertheless, as we have seen, only one such enabling color selection subnet can be active at a time.

Phase II is executed completely analogous to Lemma 4.5.1. Phase III is slightly modified in the sense that the places p'_{oj} have to be cleaned-up as well.

Let $\Sigma' = \Sigma \cup \{d\}$ where $d \notin \Sigma$. If we label each replacement transition t'_{kv} corresponding to some transition $t_k \in T$ by $L(t_k)$ ($L(t_k) \in \Sigma$) and all other transitions by d we see that the number of symbols d generated between two consecutive symbols from Σ in any firing sequence of CP'_{Σ}, is still bounded by a constant which is predetermined by the structure of CN. In fact, one can verify that at the most $2 \cdot |A^i| + 2 \cdot |p| + 2$ symbols d can be generated in between two symbols from Σ ($|p|$ transitions have to be fired in the "detection" process of Phase I and $|p|$ transitions must be fired during the clean-up process of the places p'_{oj}).

Let then $W' \subseteq (\Sigma \cup \{d\})^*$ be the language generated by CP'_{Σ}, $W \subseteq \Sigma^*$ the language generated by CP_{Σ} and $h'' : (\Sigma \cup \{d\})^* \to \Sigma$ the homomorphism defined by $h''(d) = \lambda$ and $h''(a) = a$, for each $a \in \Sigma$. Again, $h''(W') = W$ and h'' is a k-limited erasing on W' with $k = 2 \cdot |A^i| + 2 \cdot |p| + 2$. By the same argument as in Lemma 4.5.1 and by Theorem 4.5.1 there exists a λ-transition free Labelled C-CPM (without modified color threshold and color output functions) which generates the language W. Thus, the modification regarding the color threshold functions and the color output functions under consideration does not affect the language families Λ(C-CPM) and Λ_o(C-CPM).

In fact, one can show using a similar construct that the language families Λ(C-CPM) and Λ_o(C-CPM) are not affected even if we allow the following more general type of mappings as color output functions:

- if $a^r_{ks} \in A$ is an output arc and $F(a^r_{ks}) = f^r_{ks}$ then $f^r_{ks}: \mathcal{P}(X) \times \mathcal{P}(X) \to X$ and in any color marking CM^i the color of the token produced on the arc a^r_{ks} in case t_k fires in CM^i is $f^r_{ks}(\mathcal{D}(\theta^i_k), \mathcal{D}(C^{i-}_s))$, where

$$C^{i-}_s = \begin{cases} C^i_s & \text{if } p_s \notin \mathcal{D}(I_k) \\ C^i_s - \theta^i_{sk} & \text{if } p_s \in \mathcal{D}(I_k) \end{cases}$$

In this case we have to introduce in CN' a separate detection subnet for each place p_j of the original net.

The construct of CN' is further complicated by the fact that there may exist places p_s on which some original transition t_k self-loops. In this situation we cannot use a detection subnet in order to "detect" the domain of the bag of colors in the corresponding place p'_s before the firing of any replacement transition t'_{kv} of t_k because the firing of t'_{kv} itself modifies the bag of colors in p'_s and possibly the domain of the respective bag of colors. This inconvenience can be however overcome by the following procedure:

a. The simulation of the firing of $t_k \in T$ is performed as before, up to the point where the replacement transition t'_{kv} selected to fire has collected its input tokens. The simulation of the firing of t_k stops and the simulation net CN' "memorizes" which transition t'_{kv} has fired.

b. A clean-up process is started for the places p'_{os} of each detection subnet or at least for the detection subnets corresponding to places $p_s \in \mathcal{D}(I_k) \cap \mathcal{D}(O_k)$.

c. The domains of the bags of colors in this "intermediate" color marking are "detected".

d. The simulation of the firing of t_k resumes. The output tokens produced by t'_{kv} are now deposited in the respective places. The color of each output token is, however, decided based on the results of the last detection process, i.e. corresponding to the intermediate color marking.

The number of consecutive firings of transitions labelled d will still be bounded by a constant predetermined by the structure of the original net.

This completes the discussion regarding these extended versions of the C-CPM.

Two other formal models of concurrent systems, namely the Coordination Nets ([PATI-70]) and the Petri Nets with switches, disjunctive logic and token absorbers, were shown in [AGAR-75] to be "complete". In the terminology of [AGAR-75], a given formal model is "complete with respect to the representation of flow of control in concurrent systems if and only if it is equivalent to the Turing machine central control model". Translated to the framework of our study, the completeness of

a formal model implies that the respective model can generate under unrestricted erasure the set of recursively enumerable languages. A tighter relationship between the Coordination Nets, the Petri Nets with switches, disjunctive logic and token absorbers and the C-CPM can, however, be exhibited. Thus, in the following two sections we shall relate the language families Λ(C-CPM) and Λ_o(C-CPM) to the Λ and Λ_o language families generated by the Coordination Nets and the Petri Nets with switches, disjunctive logic and token absorbers, respectively.

Section 4.6 COORDINATION NETS REVISITED

As we have already mentioned, for any GPN which models a certain process synchronization system, the particular arrangement of arcs, transitions and places of the respective net implements the constraints existing in the modelled system, constraints which force the system to execute only selected synchronization sequences over the set of underlying processes rather than all possible such sequences. In the case of the Coordination Nets, the same constraints are represented implicitly through the structure of the net and explicitly via a constraint set. Thus:

Definition 4.6.1

A Coordination Petri Net (COPN) is a system CON = (N, Ct) such that:

1. N = (T, P, I, O) is a GPN.

2. $Ct \subseteq \mathcal{P}(P)$ is a constraint set. If $\omega \in Ct$ then the subset ω of P is called a constraint.

A marking of a Coordination Petri Net is represented by a total, single-valued mapping from P into Z^o. Given a marking M^i of CON, a transition $t_k \in T$ is enabled in M^i if and only if for each $p_j \in P$ $M^i(p_j) \geq I(p_j, t_k)$. If a transition t_k enabled in M^i, fires in M^i and $M^i[t_k> M^{i+1}$ then for each $p_j \in P$, $M^{i+1}(p_j) = M^i(p_j) - I(p_j, t_k) + O(t_k, p_j)$. Let us denote by NZ^i the set of places $p_j \in P$ such that $M^i(p_j) > 0$. Then:

Definition 4.6.2

A transition t_k enabled in M^i may be selected to fire in the marking M^i if and only if:

$$Ct \cap \mathcal{P}(NZ^{i+1}) = \phi.$$

If M^{i+1} is a marking such that $Ct \cap \mathcal{P}(NZ^{i+1}) = \phi$ then we shall call M^{i+1} an admissible marking. Thus, $t_k \in T$ may be selected to fire in some marking M^i if and only if t_k is enabled in M^i and the firing of t_k leads to an admissible marking.

Definition 4.6.3

A Coordination Petri Model (COPM) is a system $\mathcal{CN} = (CON, M^o)$ where:

1. CON = (N, Ct) is a Coordination Petri Net.
2. M^o is the initial marking of CON. M^o must be an admissible marking.

We can see that the execution of a Coordination Petri Model proceeds from the initial marking M^o such that in any reachable marking M^r (including M^o), for each constraint $\omega \in Ct$, there exists at least one place $p_j \in \omega$ such that $M^r(p_j) = 0$ (i.e, ω is not "violated" in M^r).

A Labelled Coordination Petri Model is defined the same way as a Labelled C-CPM, for example. Similarly, the language families Λ(COPM) and Λ_o(COPM) are defined analogous to Λ(C-CPM) and Λ_o(C-CPM), respectively.

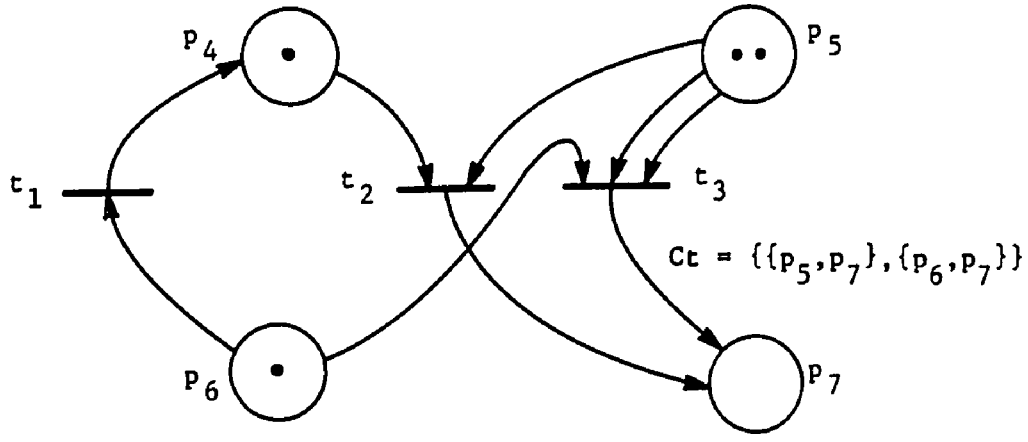


Figure 4.6.1

Sample COPM

The marking given in Figure 4.6.1 is evidently admissible. Transitions t_1 and t_3 are firable while t_2 is not.

Some of the definitions pertinent to the COPM, as given here, are slightly different from those employed in [PATI-70]. Later on in this section we shall elaborate on the differences between our version of the COPM and the original version.

Let us introduce the following notations. For each $p_j \in P$:

$$\Omega(p_j) = \{\omega \mid \omega \in Ct \text{ and } p_j \in \omega\}.$$

Consider now an arbitrary transition $t_k \in T$. Then:

$$IC(t_k) = \bigcup_{p_j \in \mathcal{D}(I_k)} \Omega(p_j)$$

$$OC(t_k) = \bigcup_{p_j \in \mathcal{D}(O_k)} \Omega(p_j)$$

We shall proceed now to investigate the ranges of the language families $\Lambda(COPM)$ and $\Lambda_o(COPM)$.

Lemma 4.6.1

$\Lambda(COPM) \subseteq \Lambda(EPM(I))$ and $\Lambda_o(COPM) \subseteq \Lambda_o(EPM(I))$.

Proof: Let $CN_\Sigma = (CN, \Sigma, L)$ be a Labelled COPM where $CN = (CON, M^o)$, $CON = (N, Ct)$ and $N = (T, P, I, O)$. Consider an arbitrary transition $t_k \in T$ and let ω be a constraint in Ct. It is easy to see that if $\omega \notin IC(t_k) \cup OC(t_k)$ then the firing of t_k in any admissible marking M^i cannot violate the constraint ω. Hence, the firability of t_k in M^i depends only on the constraints in $IC(t_k) \cup OC(t_k)$.

Suppose now that $\omega \in IC(t_k) \cup OC(t_k)$. There are only three possible cases to be considered.

A. $\omega \in IC(t_k) - OC(t_k)$. Suppose M^i is an admissible marking of CON in which t_k is enabled. Let us denote by M^{i+1} the marking which would be obtained upon the firing of t_k in M^i.

Since M^i is assumed to be admissible, there exists a place $p_r \in \omega$ such that $M^i(p_r) = 0$. But, $p_r \notin \mathcal{D}(I_k)$ because t_k is assumed to be

enabled in M^i. Also, $p_r \notin \mathcal{D}(O_k)$ because $\omega \cap \mathcal{D}(O_k) = \phi$. Hence:

$$M^{i+1}(p_r) = M^i(p_r) - I(p_r, t_k) + O(t_k, p_r) = M^i(p_r) = 0$$

Consequently, the constraint ω is not violated in the marking M^{i+1}. We can conclude that the firability of t_k is independent of the constraints in this category.

B. $\underline{\omega \in OC(t_k) - IC(t_k)}$. Let M^i and M^{i+1} be as in case A. Since $\omega \cap \mathcal{D}(I_k) = \phi$, we must have for each $p_r \in \omega$:

$$M^{i+1}(p_r) = M^i(p_r) + O(t_k, p_r).$$

If $p_r \in \mathcal{D}(O_k)$ then $M^{i+1}(p_r) > 0$ independent of $M^i(p_r)$. Therefore:

$$(M^{i+1}(p_r) = 0) \text{ if and only if } (M^i(p_r) = 0) \bigwedge (p_r \notin \mathcal{D}(O_k))$$

Consequently, the constraint ω is not violated in the marking M^{i+1} if and only if

$$\bigvee_{p_r \in \omega - \mathcal{D}(O_k)} M^i(p_r) = 0 \qquad \text{(inclusive or)}$$

C. $\underline{\omega \in IC(t_k) \cap OC(t_k)}$. There are four possibilities to be examined. Suppose M^i and M^{i+1} are as in case A and let p_j be a place in ω.

C1. $\underline{p_j \in \mathcal{D}(I_k) \text{ and } p_j \in \mathcal{D}(O_k)}$. Thus, t_k self-loops on p_j. Consequently, $M^{i+1}(P_j) > 0$ always.

C2. $\underline{p_j \notin \mathcal{D}(I_k) \text{ but } p_j \in \mathcal{D}(O_k)}$. Then:

$$M^{i+1}(p_j) = M^i(p_j) + O(t_k, p_j)$$

and thus $M^{i+1}(p_j) > 0$ independent of $M^i(p_j)$.

C3. $\underline{p_j \in \mathcal{D}(I_k) \text{ but } p_j \notin \mathcal{D}(O_k)}$. Thus:

$$M^{i+1}(p_j) = M^i(p_j) - I(p_j, t_k).$$

Consequently, $M^{i+1}(p_j) = 0$ if and only if $M^i(p_j) = I(p_j, t_k)$.

C4. $\underline{p_j \notin \mathcal{D}(I_k) \text{ and } p_j \notin \mathcal{D}(O_k)}$. As shown in case A we shall have $M^{i+1}(p_j) = M^i(p_j)$ and thus $M^{i+1}(p_j) = 0$ if and only if $M^i(p_j) = 0$.

We can conclude that the constraint ω is not violated in the marking M^{i+1} if and only if:

$$\left[\bigvee_{p_j \in \omega \cap (\mathcal{D}(I_k) - \mathcal{D}(O_k))} M^i(p_j) = I(p_j, t_k)\right] \vee \left[\bigvee_{p_j \in \omega - (\mathcal{D}(I_k) \cup \mathcal{D}(O_k))} M^i(p_j) = 0\right]$$

We can now summarize the results obtained so far as follows:

(M^{i+1} is admissible) if and only if

$$\bigwedge_{\omega \in OC(t_k) - IC(t_k)} \left[\bigvee_{p_r \in \omega - \mathcal{D}(O_k)} M^i(p_r) = 0\right] \wedge$$

$$\bigwedge_{\omega \in IC(t_k) \cap OC(t_k)} \left\{\left[\bigvee_{p_j \in \omega \cap (\mathcal{D}(I_k) - \mathcal{D}(O_k))} M^i(p_j) = I(p_j, t_k)\right] \vee \left[\bigvee_{p_j \in \omega - (\mathcal{D}(I_k) \cup \mathcal{D}(O_k))} M^i(p_j) = 0\right]\right\}$$

This condition can now be put in disjunctive form. Each term of the disjunctive form will represent a condition on M^i under which M^{i+1} is an admissible marking. Let us consider such a term of the disjunctive form. In general, it will involve several places, let us denote them by $p_{j1}, \ldots, p_{jr}$, for which it is required that $M^i(p_{js}, t_k) = I(p_{js}, t_k)$, $1 \leq s \leq r$, and several places, let us denote them by $p_{jr+1}, \ldots, p_{jm}$ for which it is required that $M^i(p_{js}) = 0$, $r + 1 \leq s \leq m$. We emphasize that no contradition can occur because we can have only the following two cases:

i. For each s, $1 \leq s \leq r$, $p_{js} \in \omega \cap (\mathcal{D}(I_k) - \mathcal{D}(O_k))$ for some constraint $\omega \in IC(t_k) \cap OC(t_k)$. Hence, $p_{js} \in \mathcal{D}(I_k)$ but $p_{js} \notin \mathcal{D}(O_k)$.

ii. For each s, $r + 1 \leq s \leq m$, $p_{js} \in \omega - \mathcal{D}(O_k)$ fór some constraint $\omega \in OC(t_k) - IC(t_k)$ or $p_{js} \in \omega - (\mathcal{D}(I_k) \cup \mathcal{D}(O_k))$ for some constraint $\omega \in IC(t_k) \cap OC(t_k)$. In either case $p_{js} \notin \mathcal{D}(I_k)$ and $p_{js} \notin \mathcal{D}(O_k)$.

It is easy now to construct a λ-transition free Labelled EPM(I) $EN_\Sigma = (EN, \Sigma, L')$ which generates the same language as CN_Σ. Let $EN = (EN, M^o)$ where $EN = (T', P', I', O')$. Our construct will preserve the places of P, i.e., for each $p_j \in P$ we shall introduce a corresponding place p'_j in P'. Consider now a transition t_k of the original COPN and suppose we have determined the condition under which

t_k is firable. For each term in the disjunctive form we shall introduce in T' a separate replacement transition. Let t'_{ks} be such a replacement transition. The input and output connections of t'_{ks} are determined as follows:

1. If p is a place for which we require in the respective term of the disjunctive form $M^i(p) = I(p, t_k) = m$ then:

$I'(p', t'_{ks}) = (\sigma, m)$; i.e. p' and t'_{ks} are connected by a positive testing arc.

2. If p is a place for which we require in the respective term of the disjunctive form $M^i(p) = 0$ then:

$I'(p', t'_{ks}) = \zeta$; i.e. p' and t'_{ks} are connected by an inhibitor arc.

3. If p is an input place of t_k other than the places considered in case i) above then:

$I'(p', t'_{ks}) = I(p, t_k)$, i.e. p' is a normal input place of the replacement transition t'_{ks}.

4. For each place $p' \in P'$:

$O'(t'_{ks}, p') = O(t_k, p)$.

We note that if $OC(t_k) = \phi$ for some transition $t_k \in T$ then the firability of t_k does not depend on any constraint $\omega \in Ct$. Hence, t_k is firable in some admissible marking M^i if and only if t_k is enabled in M^i. Consequently we shall introduce in T' only one replacement transition of t_k, let it be t'_k, such that for each place $p' \in P'$ we shall have

$I'(p', t'_k) = I(p, t_k)$ and $O'(t'_k, p') = O(t_k, p)$.

On the other hand, if $OC(t_k) \neq \phi$ but there exists a constraint ω in $OC(t_k)$ such that $\omega - \mathscr{D}(O_k) = \phi$ then t_k can never become firable. Consequently, we can remove t_k from T without modifying the language generated by CN_Σ. We shall certainly introduce no replacement transitions for t_k in T'.

The initial and the final marking of EN are the same as the initial and the final marking of the original COPN, respectively.

The labelling function L' is defined such that all replacement transitions t'_{ks} introduced for a transition t_k of the original net are

assigned the label $L(t_k)$.

It is easy to verify that the Labelled EPM(I) $\mathcal{EN}_\Sigma$ generates the same language as $\mathcal{CN}_\Sigma$. The construct applies to both language families Λ(COPM) and Λ_o(COPM).

□

Figure 4.6.3 depicts the EPM(I) obtained by applying the construct of Lemma 4.6.1 to the COPM of Figure 4.6.2. Note that transition t_3 in Figure 4.6.2 can never become firable and therefore can be removed.

We shall proceed now to prove the converse of Lemma 4.6.1. For the sake of simplicity we have chosen to show how EPM's can be converted in a λ-transition free, language preserving manner into COPM's. The construct is nevertheless still rather complex and therefore we shall first give an example of it. Consider the sample EPM of Figure 4.6.4. Both transitions self-loop on a control place Π. Evidently, such a control place can be introduced in any EPM without altering the firing sequences which can be performed by the respective net (assuming that Π contains one token in the initial marking).

If no final marking is given, the nonempty label sequences generated by this EPM are of the form $a^y b^{y+1}$, where $0 \leq y \leq x$ and x is the initial marking of the place p_s. It can easily be verified that the COPM of Figure 4.6.5 generates exactly the same label sequences.

Transition t_k can fire in some marking M^i if and only if $M^i(p_r) = 0$ and $M^i(p_s) \geq 1$. The transitions t_k', t_k'' and t_k''' can fire only under these conditions also or else one of the constraints would be violated. Transition t_k' simulates the first firing of t_k while t_k'' and t_k''' simulate subsequent consecutive firings of t_k. For example a firing sequence of the original EPM starting with the prefix $t_k t_k t_k t_k \ldots$ would be simulated in the COPM of Figure 4.6.5 by a firing sequence with the prefix $t_k' t_k'' t_k''' t_k'' \ldots$ Evidently, the maximum number of passes through the "circuit" formed by t_k'' and t_k''' (or, equivalently, the maximum number of consecutive firings of t_k in the original EPM) is bounded by the markings of the normal input places of t_k, in this case by the marking of p_s. We shall call t_k' as the "first" replacement transition of t_k.

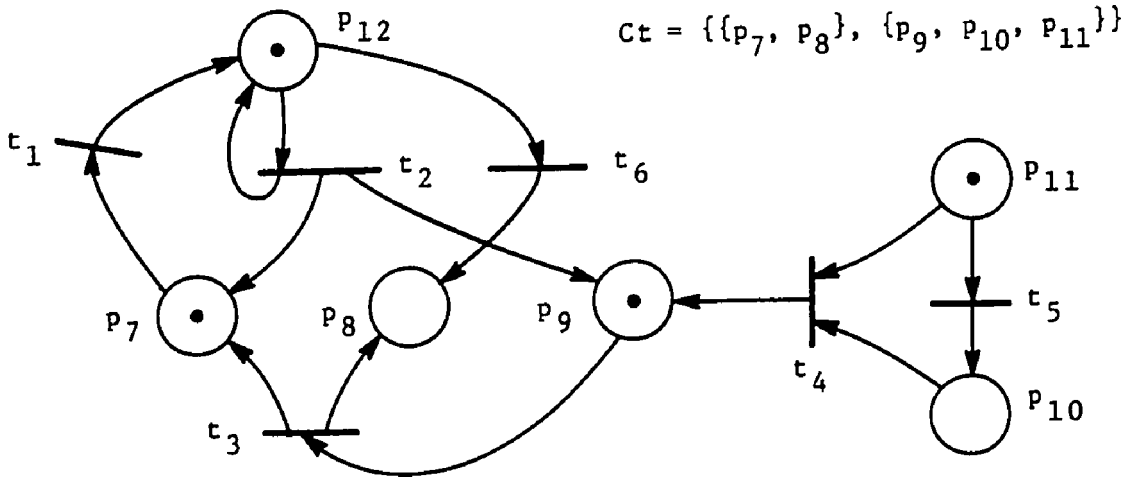


Figure 4.6.2

Sample Coordination Petri Model

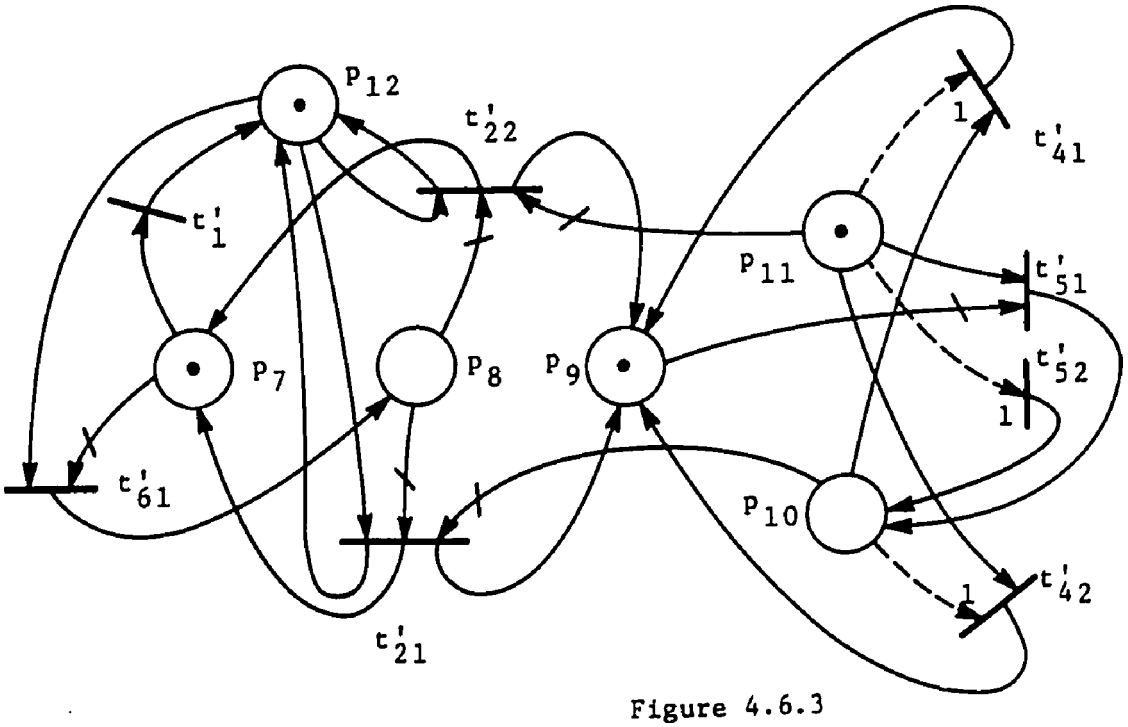


Figure 4.6.3

EPM(I) Obtained from the COPM of Figure 4.6.2

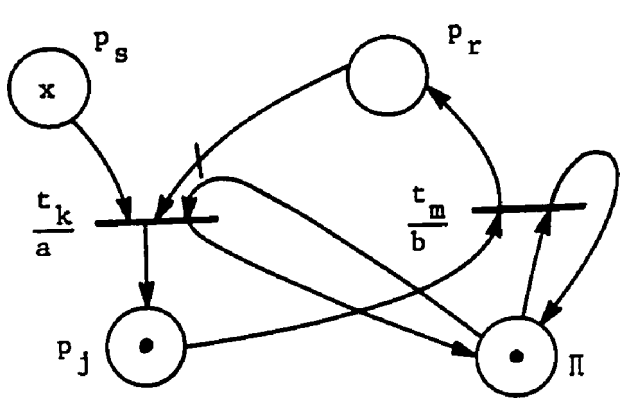


Figure 4.6.4

Sample Labelled EPM

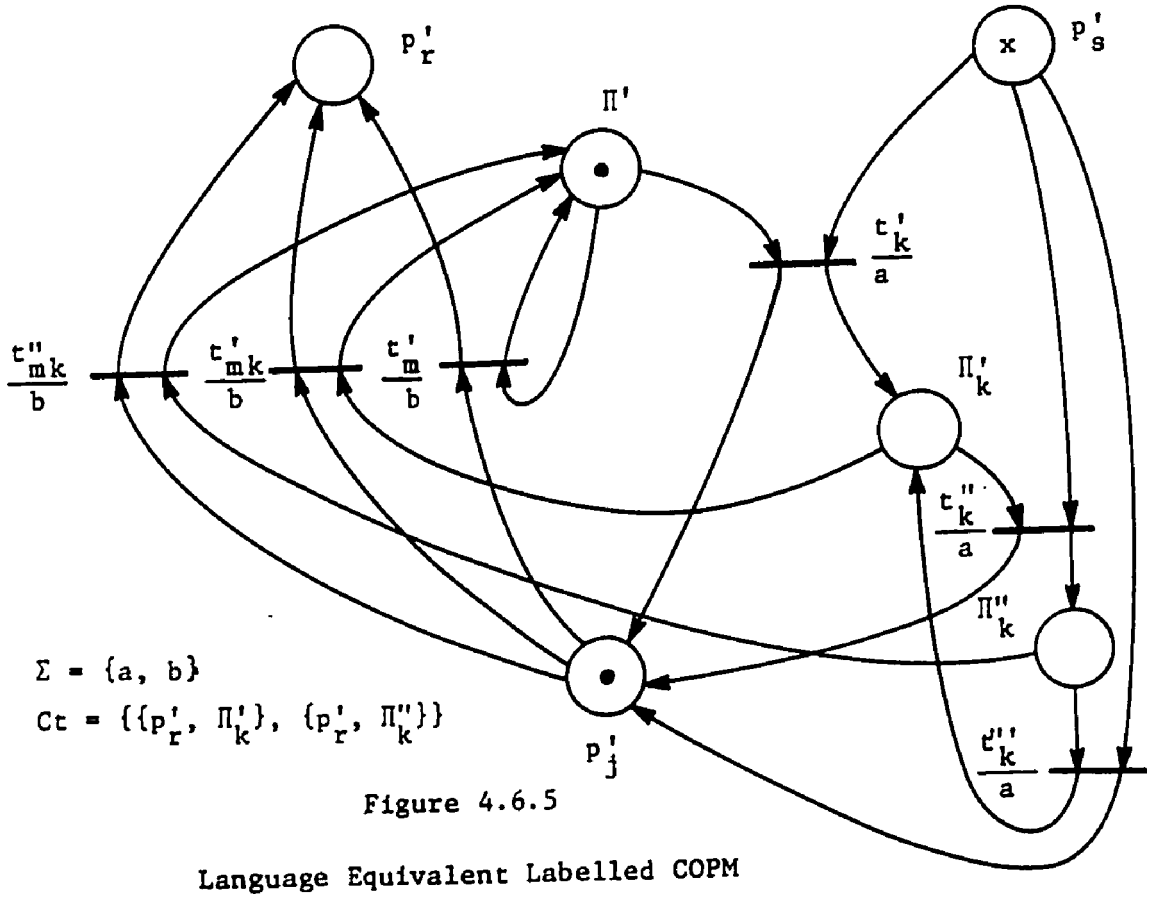


Figure 4.6.5

Language Equivalent Labelled COPM

On the other hand, transition t_m of Figure 4.6.4 can fire if and only if in the current marking M^i, $M^i(p_j) \geq 1$. Transition t'_m of Figure 4.6.5 can fire under exactly the same condition. The additional transitions t'_{mk} and t''_{mk} are used to simulate the firing of t_m immediately following an odd or even number of consecutive firings of t_k, respectively. From this point of view t'_m simulates the firing of t_m in case t_k has not fired and/or any subsequent firings of t_m. For example, the firing sequences of the original EPM starting with the prefixes t_m, $t_k t_m t_m$ or $t_k t_k t_k t_k t_m t_m t_m$ ... would be simulated by firing sequences starting with the prefixes t'_m, $t'_k t'_{mk} t'_m$ and $t'_k t''_k t'''_k t''_k t''_{mk} t'_m t'_m$ ... , respectively. Otherwise viewed, t'_{mk} and t''_{mk} are used to "break" the "circuit" formed by t''_k and t'''_k, thus avoiding the violation of any constraint and restoring the token in the control place Π'.

This example, even though it is rather simple, displays the basic problems encountered when converting EPM's to COPM's without the use of λ-transitions. In what follows, we shall generalize the construct presented above in order to suit arbitrary EPM's.

Lemma 4.6.2

$\Lambda(\text{EPM}) \subseteq \Lambda(\text{COPM})$ and $\Lambda_o(\text{EPM}) \subseteq \Lambda_o(\text{COPM})$

Proof: In order to simplify our construct we shall make use of the following result regarding the structure of the EPM's reported in [HACK-75]:

Any EPM can be converted in a λ-free, language preserving manner into a Restricted EPM.

A Restricted EPM (REPM) is an EPM $EN = (EN, M^o)$ where $EN = (T, P, I, O)$ is such that:

1. $I(p_j, t_k) \leq 1$ or $I(p_j, t_k) = \zeta$ for each pair $(p_j, t_k) \in P \times T$.
2. $O(t_k, p_j) \leq 1$ for each pair $(t_k, p_j) \in T \times P$.
3. $\mathcal{D}(I_k) \cap \mathcal{D}(O_k) = \phi$ for each $t_k \in T$.

Thus, in an REPM multiple arcs and self-loop places are not allowed.

Without loss in generality we shall restrict our construct to REPM's. Consider a Labelled EPM $EN_\Sigma = (EN, \Sigma, L)$ where $EN = (EN, M^o)$ is an REPM. Let $EN = (T, P, I, O)$ and suppose we introduce in P a new place Π on which all transitions of T self-loop. Let

also $M^0(\Pi) = 1$. Certainly, this modification does not affect the firing sequences which could be executed by EN. With this change, EN remains an REPM with the exception of the control place Π. In fact, we are not interested in converting strict REPM's, we rather want to avoid the cases where part of a self-loop is an inhibitor arc (condition which is still satisfied by EN).

Let us now proceed to build a COPM $CN_\Sigma = (CN, \Sigma, L')$ which generates the same language as EN_Σ. Suppose $CN = (CON, M^{0'})$ where CON = (N, Ct) and N = (T', P', I', O') is the underlying GPN.

Our construct will preserve the places in $P \cup \{\Pi\}$, i.e., for each $p \in P \cup \{\Pi\}$ we shall introduce a corresponding place p' in P'. In particular, let Π' be the place of P' corresponding to the control place Π of EN.

Let us denote by T^z the following subset of T:

$$T^z = \{t_k \mid t_k \in T \text{ and } P_k^z \neq \phi\}$$

(P_k^z denotes the set of inhibitor input places of t_k). In order to simplify notations let us assume that $T^z = \{t_1, \ldots, t_s\}$. Consider now a transition $t_k \in T^z$. We shall replace t_k by three transitions in CON, denoted by t_k', t_k'' and t_k'''. We shall also introduce two new places in P' denoted by Π_k' and Π_k''. The transitions t_k', t_k'' and t_k''' will have the following input and output connections:

1. For each place $p_j' \in P'$ corresponding to some place $p_j \in P_k^n$ (P_k^n denotes the set of normal input places of the original transition t_k):

$$I'(p_j', t_k') = I'(p_j', t_k'') = I'(p_j', t_k''') = I(p_j, t_k)$$

2. For each place $p_j' \in P'$ corresponding to some place $p_j \in P_k^z$.

$$I'(p_j', t_k') = I'(p_j', t_k'') = I'(p_j', t_k''') = 0$$

3. For each place $p_j' \in P'$ corresponding to some place $p_j \in P$

$$O'(t_k', p_j') = O'(t_k'', p_j') = O(t_k''', p_j') = O(t_k, p_j)$$

To Π', Π_k' and Π_k'' the transitions t_k', t_k'' and t_k''' are connected exactly as shown in Figure 4.6.5.

Next, we shall introduce in Ct the constraints $\omega_{kj}' = \{p_{kj}', \Pi_k'\}$ and $\omega_{kj}'' = \{p_{kj}', \Pi_k''\}$, for $1 \leq j \leq r$, where we have assumed $P_k^z = \{p_{k1}, \ldots, p_{kr}\}$.

This construct is performed for each transition $t_k \in T^z$. The

transitions t'_k, t''_k and t'''_k play the same role as in the example we have examined earlier.

Finally, for each transition $t_k \in T^z$ we shall introduce $2 \cdot (s - 1)$ more replacement transitions in T' denoted by t'_{kq} and t''_{kq}, $1 \leq q \leq s$ and $q \neq k$. All these replacement transitions have the same input and output connections as t'_k except for the place Π'. The replacement transitions t'_{kq} and t''_{kq} have Π'_q and Π''_q rather than Π' as input places, respectively. The complete set of replacement transitions introduced for some transition $t_k \in T^z$ is illustrated in Figure 4.6.6. The role of the transitions t'_{kq} and t''_{kq} is to simulate the firing of the original transition t_k immediately following an arbitrary number of consecutive firings of some other transition $t_q \in T^z$.

Consider now a transition $t_m \in T$ which does not have any inhibitor input places. We shall introduce in T' for t_m $2 \cdot s + 1$ replacement transitions denoted by t'_m, t'_{mq} and t''_{mq}, for $1 \leq q \leq s$. We shall set:

1. For each place $p'_j \in P'$ corresponding to some place $p_j \in P$

$$I'(p'_j, t'_m) = I'(p'_j, t'_{mq}) = I'(p'_j, t''_{mq}) = I(p_j, t_m)$$

$$O'(t'_m, p'_j) = O'(t'_{mq}, p'_j) = O'(t''_{mq}, p'_j) = O(t_m, p_j)$$

In addition, the transitions t'_m, t'_{mq} and t''_{mq} are connected to Π', Π'_q and Π''_q, for each q, $1 \leq q \leq s$, as shown in Figure 4.6.7.

The initial marking of CON will be as follows:

1. For each place $p'_j \in P'$ corresponding to some place $p_j \in P$, $M^{o'}(p'_j) = M^o(p_j)$.

2. $M^{o'}(\Pi') = M^o(\Pi) = 1$, $M^{o'}(\Pi'_q) = M^{o'}(\Pi''_q) = 0$ for each q, $1 \leq q \leq s$.

It is important to see that in any reachable marking of CON one and only one of the places Π', Π'_q and Π''_q, $1 \leq q \leq s$, contains a token. Consequently, at any time at most one of the replacement transitions introduced for some transition of the original EPM can be enabled.

Suppose for example that in the current marking $M^{i'}$, $M^{i'}(\Pi'_q) = 1$ for some q, $1 \leq q \leq s$, and consider the replacement transitions introduced for the transition t_m, where $t_m \in T - T^z$. Certainly, only t'_{mq} can be enabled. If $M^{i+1'}$ is the marking obtained upon the firing of t'_{mq} in $M^{i'}$, then $M^{i+1'}(\Pi') = 1$ and $M^{i+1'}(\Pi'_r) = M^{i+1'}(\Pi''_r) = 0$ for each

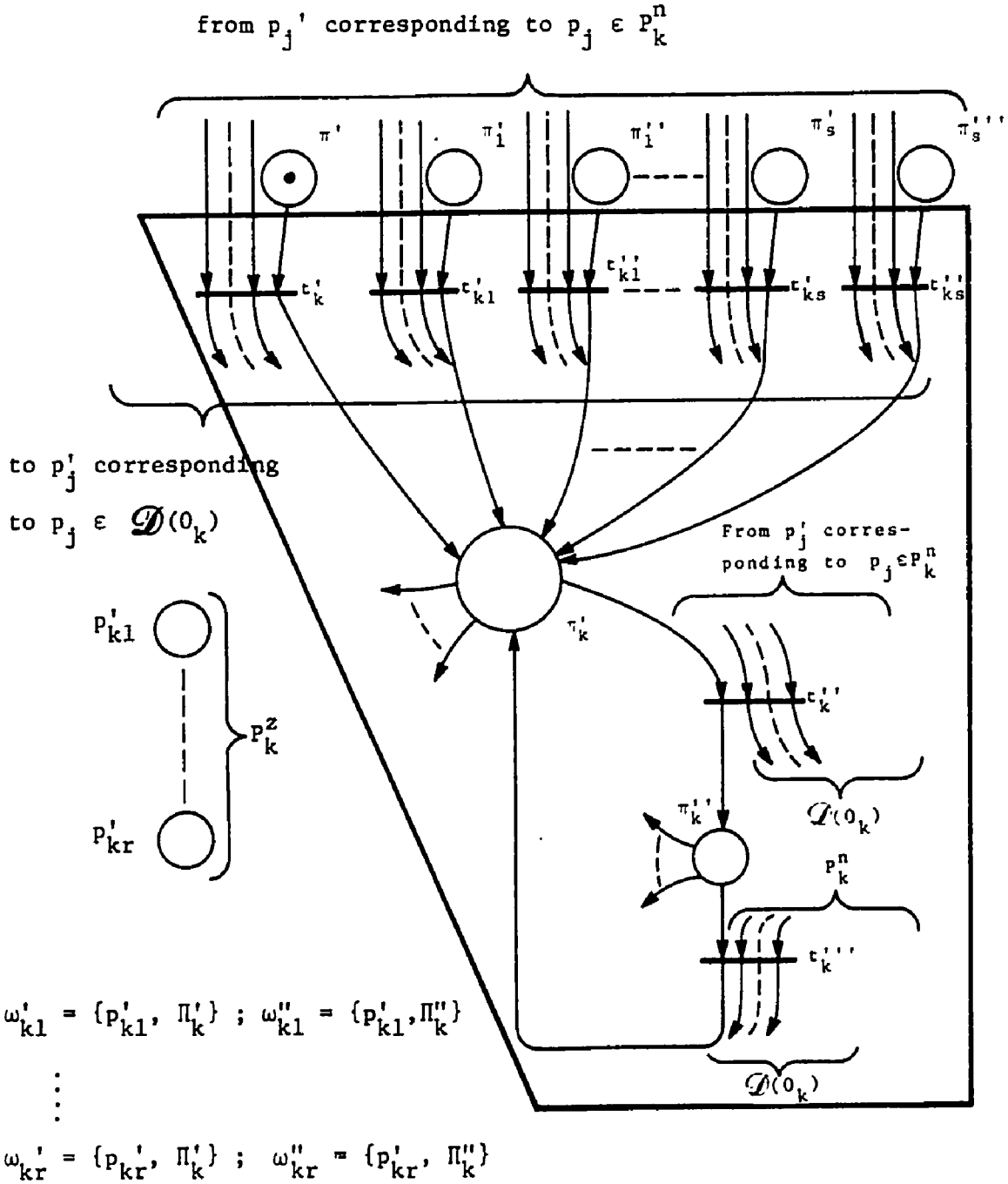


Figure 4.6.6

Replacement Transitions of $t_k \in T^z$

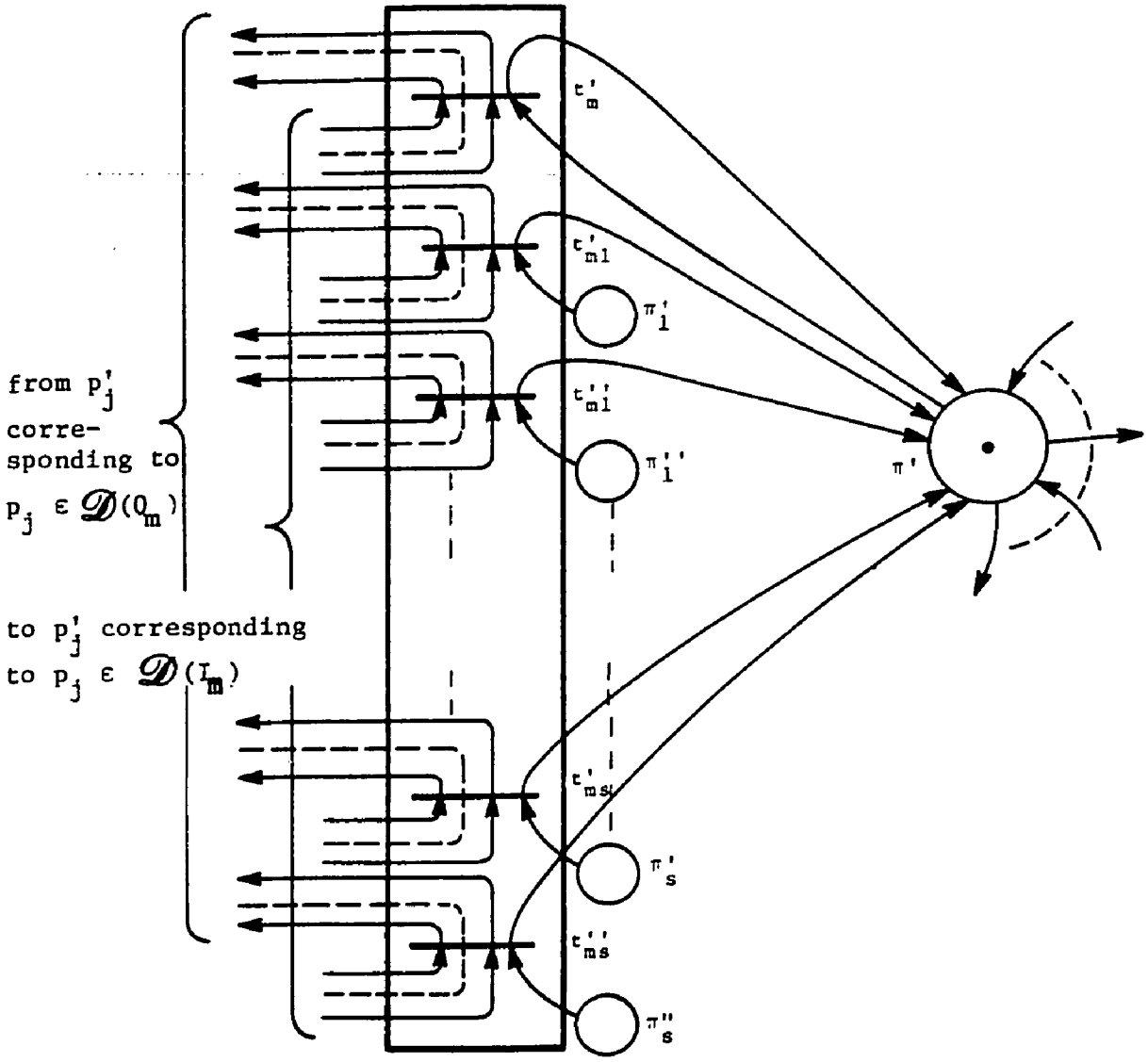


Figure 4.6.7

Replacement Transitions of $t_m \in T\text{-}T^z$

r, $1 \leq r \leq s$. Thus, the firing of t'_{mq} cannot violate any constraint and for that matter the firability of t'_{mq} is independent of any constraint.

Consider now the replacement transitions introduced for some transition $t_k \in T$, $k \neq q$. In $M^{i'}$, only t'_{kq} can be enabled. Let again $M^{i+1'}$ be the marking which the firing of t'_{kq} in $M^{i'}$ will lead to. Then $M^{i+1'}(\Pi') = M^{i+1'}(\Pi''_k) = M^{i+1'}(\Pi'_r) = M^{i+1'}(\Pi''_r) = 0$ for each r, $1 \leq r \leq s$ and $r \neq k$. Also $M^{i+1'}(\Pi'_k) = 1$. It follows that the firing of t'_{kq} is independent of any constraint of the form $\{p'_{qj}, \Pi'_q\}$ or $\{p'_{qj}, \Pi''_q\}$ for $1 \leq q \leq s$, $q \neq k$ and for all j, $1 \leq j \leq |P^z_q|$. The only constraints which influence the firability of t'_{kq} are the constraints introduced for t_k, in particular the constraints of the form $\{p'_{kj}, \Pi'_k\}$ where p'_{kj} is a place of P' corresponding to a place $p \in P^z_k$. Finally, the firing of t''_k and t'''_k follows the pattern explained in our earlier example.

The construct presented so far shows that $\Lambda(EPM) \subseteq \Lambda(COPM)$. For languages in the family $\Lambda_o(EPM)$ the construct does not guarantee unique final markings due to the places Π'_q and Π''_q, $1 \leq q \leq s$. In what follows we shall describe a method to solve this problem.

Consider the original EPM $EN = (EN, M^o)$ and suppose we are given a final marking M^f of EN. Let t_s be a transition of T such that there exists a marking M^{f-1} of EN and $M^{f-1}[t_s > M^f$. The transition t_s will be called a "last" transition. Certainly, given M^f the set of all "last" transitions can be determined.

Let us introduce in EN a distinct "stop" transition t_{sstop} for each "last" transition t_s. We shall set:

1. For each place $p_j \in P$

$$I(p_j, t_{sstop}) = I(p_j, t_s)$$

and

$$O(t_{sstop}, p_j) = O(t_s, p_j)$$

2. $I(\Pi, t_{sstop}) = 1$; $O(t_{sstop}, \Pi) = 0$

Thus, t_{sstop} can fire exactly when t_s can fire but does not restore upon firing the token in the control place Π. In particular, if $M^{f-1}[t_{sstop} > M^{f'}$ then $M^{f'}(p_j) = M^f(p_j)$ for each $p_j \in P$ and $M^{f'}(\Pi) = 0$ (rather than $M^f(\Pi) = 1$). We shall have $M^{f'}$ be the final marking of EN rather than M^f. We note that in this new final marking all transitions of EN are disabled. Nevertheless, if each "stop" transition t_{sstop} is assigned the same label as the corresponding "last" transition t_s then the language generated by EN_Σ is not altered.

Suppose now that t_s is a "last" transition with inhibitor input places. Then, for each place $p_j \in P_s^z$ we must have $M^{f-1}(p_j) = 0$ in order for t_s to be enabled. But t_s self-loops only on Π and therefore if $p_j \in P_s^z$ then $p_j \notin \mathcal{D}(O_s)$. Hence $M^f(p_j) = M^{f'}(p_j) = 0$.

Suppose now that t_r is "last" transition of EN, $t_r \neq t_s$. Then for each $p_j \in P_s^z$, $p_j \notin \mathcal{D}(O_r)$ as well or else t_r cannot lead upon firing to M^f (it is however possible to have p_j belong to $\mathcal{D}(I_r)$). If we denote by T_{last} the set of all "last" transitions and by T_{last}^z the subset of T_{last} containing all "last" transitions with inhibitor input places then we can conclude that:

1. $M^f(p_j) = 0$ for each place $p_j \in P_{last}^z$, where $P_{last}^z = \bigcup_{t_s \in T_{last}^z} P_s^z$.

(*)

and

2. $\left[\bigcup_{t_r \in T_{last}} \mathcal{D}(O_r)\right] \cap \left[\bigcup_{t_s \in T_{last}^z} P_s^z\right] = \phi$

The same remarks can be made for the set of "stop" transitions corresponding to the transitions in T_{last}.

In the COPN CON, we shall introduce in P' a new place Π'' and we shall introduce in T' 2 . s + 1 replacement transitions for each "stop" transition of EN. In Figure 4.6.8 t'_{mf}, t'_{mfq} and t''_{mfq} denote the replacement transitions introduced for some "stop" transition t_{mstop} without inhibitor input places while t'_{kf}, t'_{kfq} and t''_{kfq} denote the replacement transitions introduced for some "stop" transition t_{kstop} with inhibitor input places, $1 \leq q \leq s$. We shall also introduce in Ct a constraint $\{p', \Pi''\}$ for each place p' in P' corresponding to some

place p of P^z_{last}. Finally, we shall define the unique final marking $M^{f''}$ of *CN* by:

1. $M^{f''}(p') = M^{f'}(p) = M^{f}(p)$ for each place $p' \in P'$ corresponding to some place $p \in P$.

2. $M^{f''}(\Pi') = M^{f''}(\Pi'_q) = M^{f''}(\Pi''_q) = 0$ for each q, $1 \leq q \leq s$.

3. $M^{f''}(\Pi'') = 1$.

It is easy to see that any replacement transition introduced for some stop transition can fire only if its firing leads to a marking in which each place p' corresponding to a place $p \in P^z_{last}$ is empty or, else, one of the constraints introduced aboye would be violated. By condition (*1.) and by construction, however, $M^{f''}$ must satisfy this condition. On the other hand, by condition (*2.) and by our construct, no replacement transition introduced for some "stop" transition can deposit upon firing tokens in places p' corresponding to places $p \in P^z_{last}$. Thus, no contradiction can arise and the final marking $M^{f''}$ can indeed be attained as an admissible marking.

We note that the important factor in our construct relative to the final marking is the fact that we can avoid having inhibitor arcs as part of self-loops in EN. In fact, this assumption stands at the basis of the simulation of any transition with inhibitor input places of EN.

□

Based on Lemmas 4.6.1 and 4.6.2 and on the results reported earlier in this chapter, we can conclude:

Theorem 4.6.1

$\Lambda(COPM) = \Lambda(C\text{-}CPM)$ and $\Lambda_o(COPM) = \Lambda_o(C\text{-}CPM)$.

At the beginning of this section we have mentioned that there exist differences between our definition of the Coordination Petri Model and the original definition of it, as it appears in [PATI-70]. In the remainder of this section we shall briefly examine these differences.

Thus, in [PATI-70] the process of firing an enabled transition of a COPM encompasses two phases:

1. The transition initiates by claiming tokens from each of its input places.

2. The transition terminates by removing tokens from its input places and depositing tokens in its output places.

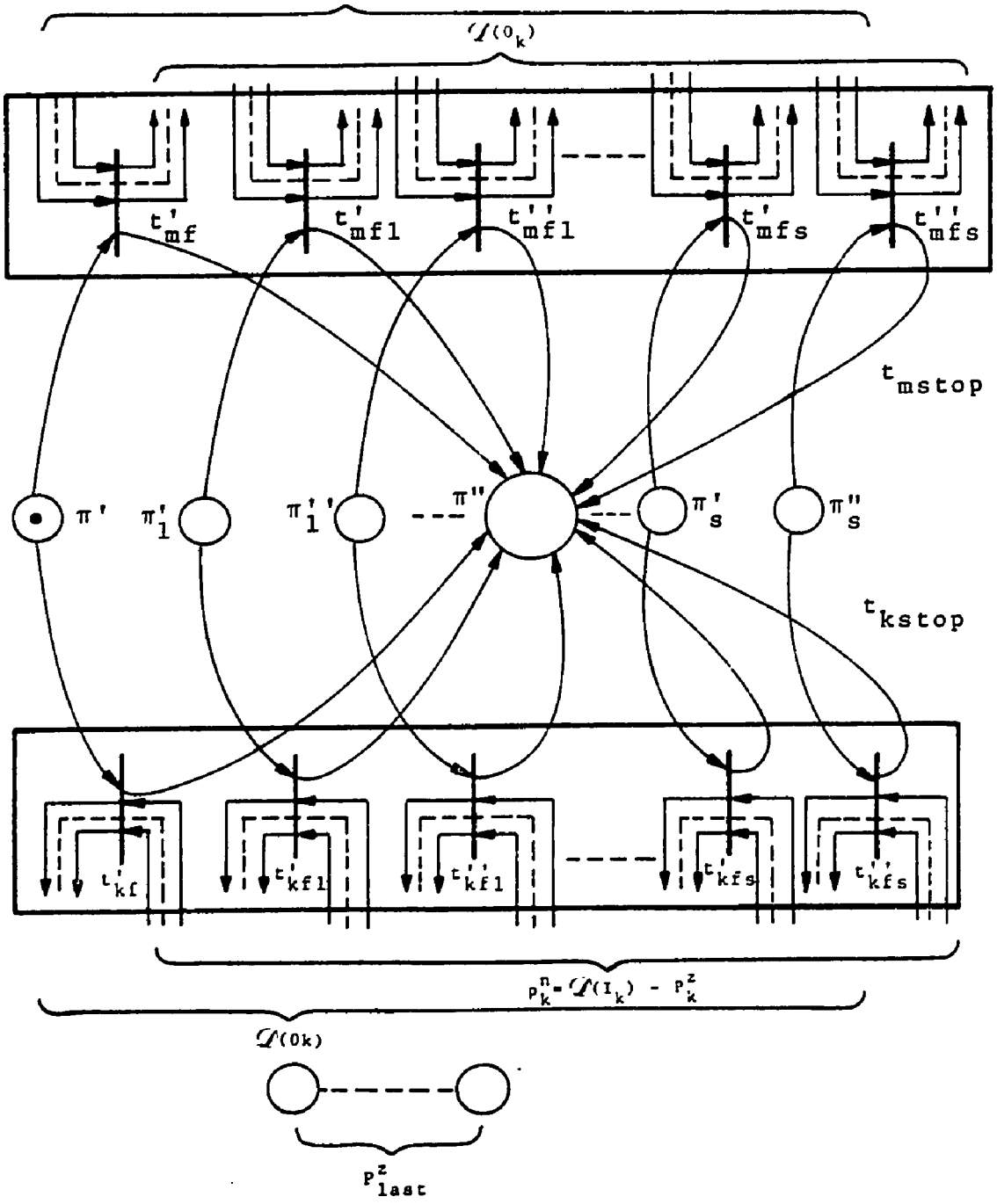


Figure 4.6.8

Replacement Transitions of "Stop" Transitions

Our definition of the firing of a transition follows the same pattern except that phase 1 does not occur explicitly.

Another distinction consists in the fact that the set T of transitions was originally defined to contain three categories of transitions:

1. Input transitions. An input transition does not initiate until the external world has reached a certain condition associated with that input transition.

2. Output transitions. 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. A transition of the COPM may be both an input and an output transition.

3. Internal transitions. These transitions do not take part in the interaction with the external world and can initiate and terminate freely, independent of external events.

Input and output transitions were introduced in order to provide a COPM with communication facilities to the external world, i.e., to the collection of processes to be coordinated by the respective net.

Certainly, the EPM, EPM(I) and for that matter the C-CPM can be provided with similar input and output transitions. In this case Lemmas 4.6.1 and 4.6.2 as well as the corresponding constructs remain applicable to these extended versions of the EPM and EPM(I). In our study, however, we are interested in comparing the intrinsic representation capabilities of the COPM with those of the other models considered in this chapter and in particular with the representation power of the C-CPM. From this point of view, it is the internal transitions which gain relevance. This is why we have treated all transitions of a COPM as internal transitions.

In [PATI-70], multiple input and/or output arcs were not considered. For the sake of modelling convenience, we have removed this restriction. COPM's without multiple arcs are certainly a particular case of the model which we have considered. Consequently, Lemma 4.6.1 remains true in this special situation. On the other hand, we note that the COPM's resulting from the construct of Lemma 4.6.2 do not employ multiple arcs. Hence, we can conclude that the introduction of multiple arcs in the COPM does not affect the language families

Λ(COPM) and Λ_o(COPM).

Even though not explicitly stated it appears that in [PATI-70] only "safe" Coordination Petri Models were considered (a COPM is "safe" if any of its reachable markings M^r is safe, that is $M^r(p_j) \leq 1$ for each place p_j of the respective net). This assertion is based on the statement "... a place [of a COPM] is either empty or it has a stone [i.e., a token] ...". encountered in [PATI-70]. Moreover the initial marking M^o is required in the original definition of the COPM to be safe. We note, however, that a safe initial marking is only a necessary but not a sufficient condition for a COPM to be safe. We also note that for safe COPM's, the language families Λ_o and Λ are equal to $\mathscr{R}$ and PREF($\mathscr{R}$), respectively where $\mathscr{R}$ denotes the set of regular languages. Indeed the set of distinct reachable markings of a safe COPM is finite and, therefore, the firing sequences of the respective net can be simulated by a finite state automaton. In this case, Λ(COPM) and Λ_o(COPM) are proper subsets of Λ(C-CPM) and Λ_o(C-CPM), respectively. In fact, as shown in [HACK-75] the sets $\mathscr{R}$ and PREF($\mathscr{R}$) can be generated as Λ_o and Λ language families, respectively by safe GPN's. Hence, in this case, the presence of the constraint set does not increase the representation capabilities of the (safe) GPN's. In our definition of the COPM, the safeness restriction has been removed which in turn lead to the result announced in Theorem 4.6.1.

The last distinction between our definition of the COPM and that given in [PATI-70] consists in the fact that in [PATI-70] input and output places of a transition were not allowed to be part of the same constraint. In our definition of the COPM we have removed this topological restriction regarding allowable constraints. It is easy to see that if t_k is a transition of a COPM whose constraint set obeys the restriction mentioned above then $IC(t_k) \cap OC(t_k) = \phi$. Lemma 4.6.1 and the corresponding construct apply directly to this case. Consequently, the topological restriction under consideration does not extend the ranges of the language families Λ(COPM) and Λ_o(COPM).

Let us now examine how the construct of Lemma 4.6.2 violates this topological restriction of [PATI-70]. Let $\{p', \Pi'_q\}$ be one of the constraints which we have introduced. Let us examine first the replacement transitions introduced for a transition t_m of the original

EPM which does not have inhibitor input places (Figure 4.6.7). Evidently, Π'_q is an input place only of the replacement transition t'_{mq} and is not an output place for any replacement transition of t_m. On the other hand, p' cannot be both an input and an output place of any replacement transition of t_m. Hence, the topological restriction on $\{p', \Pi'_q\}$ can be violated only if p' is an output place of t_{mq} or, equivalently, if p, the place of EN corresponding to p', is an output place of the original transition t_m.

Consider now the replacement transitions introduced for some transition $t_k \in T^z$ (Figure 4.6.6). Again, Π'_q is an input place only of t'_{kq} and is not an output place of any replacement transition introduced for t_k. Since no replacement transition of t_k can self-loop on p' it follows that the topological restriction on the constraint $\{p', \Pi'_q\}$ can be violated only if p' is an output place of t'_{kq}. By a similar argument we see that the replacement transitions of t_k cannot violate the topological restriction on any constraint involving Π'_k or Π''_k and in particular, the replacement transitions t'_k, t''_k and t'''_k cannot violate the topological restriction on any of the constraints which we have introduced in Ct.

The same remarks apply to constraints of the form $\{p', \Pi''_q\}$, $1 \leq q \leq s$, as well.

We can conclude that a violation of the topological restriction on constraints occurs only if a place of the original EPM is both an inhibitor input place and an output place for different transitions. As an example consider the place p_r of Figure 4.6.4. As a consequence of that particular structure, the transitions t'_{mk} and t''_{mk} of Figure 4.6.5 violate the topological restriction on the constraints $\{p'_r, \Pi'_k\}$ and $\{p'_r, \Pi''_k\}$, respectively. Finally, replacement transitions introduced for a "stop" transition of the original EPM may violate the topological restriction on constraints of the form $\{p', \Pi'_q\}$ or $\{p', \Pi''_q\}$ the same way as the other categories of replacement transitions discussed above. In addition, however, replacement transitions of a "stop" transition may also violate the restriction on constraints of the form $\{p', \Pi''\}$ if p' is an input place of the respective replacement transition (p' denotes a place of P' corresponding to some place $p \in P^z_{last}$).

Nevertheless, it appears that the construct of Lemma 4.6.2 can be modified in order to avoid these violations of the topological restriction concerning allowable constraints. The resulting construct is fairly complex, especially in the case of languages from the Λ_o family, and for the sake of brevity will not be given here.

Section 4.7 A RESULT CONCERNING THE LANGUAGE FAMILIES Λ(C-CPM) AND Λ_o(C-CPM)

In this section we shall introduce yet another variant of the EPM, which we shall call the EPM(II). The EPM(II) is a fairly interesting model in itself. Moreover, it will enable us to easily establish a result about the relationship between the language families Λ(C-CPM) and Λ_o(C-CPM). We shall also use the EPM(II) in our study of the relationship between the C-CPM and the Petri Nets with switches, disjunctive logic and token absorbers.

The EPM(II) is obtained by appending token absorbers to the EPM. Thus:

Definition 4.7.1

An Extended Petri Net (II) (EPN(II)) is a modified EPN $EN = (T, P, I, O)$ such that:

1. T is a finite set of transitions.
2. P is a finite set of places.
3. $I : P \times T \rightarrow Z^o \cup \{\zeta\}$ is the input incidence function.
4. $O : T \times P \rightarrow Z^o \cup \{\xi\}$ is the output incidence function.

The sets T and P and the mapping I are defined exactly as in the case of the EPM. Suppose now that (t_k, p_j) is a pair in $T \times P$. If $O(t_k, p_j) = s$, for some $s \in Z^+$, then p_j is called a normal output place of t_k. If, however, $O(t_k, p_j) = \xi$ then the transition t_k is connected to the place p_j by a special type of arc called a token absorber. Graphically, a token absorber is represented as a double arc with a small circle instead of an arrowhead (Figure 4.7.1). We note that the mapping O is single-valued and therefore a place p_j cannot be connected to the same transition t_k both by output arcs and by a token absorber. We shall define for each transition t_k the set:

$$P_k^a = \{p_j \mid p_j \in P \text{ and } O(t_k, p_j) = \xi\}.$$

A marking of EN is defined as a total, single-valued mapping from the set of places into Z^o. Given an arbitrary marking M^i of EN, a transition t_k is enabled in M^i under the same conditions as in the case of the EPN (Definition 4.1.10). Any transition enabled in M^i may be selected to fire in M^i.

Definition 4.7.2

If a transition t_k, enabled in the marking M^i, fires in M^i and $M^i\ [t_k> M^{i+1}$, then:

$$M^{i+1}(p_j) = \begin{cases} M^i(p_j) - I(p_j, t_k) + O(t_k, p_j) & \text{if } p_j \in P - (P_k^z \cup P_k^a) \\ O(t_k, p_j) & \text{if } p_j \in P_k^z - P_k^a \\ 0 & \text{if } p_j \in P_k^a \end{cases}$$

We note that the token absorbers are indeed a special category of arcs. Thus, if $O(t_k, p_j) = \xi$, upon the firing of t_k in M^i the place p_j will become empty, irrespective of $M^i(p_j)$. Equivalently, t_k "absorbs" all the tokens (if any) present in p_j in the marking M^i.

Figure 4.7.1 exhibits a sample EPN(II). The firing of transition t_3 in any marking will empty place p_6. We note that the marking of p_6 may grow unboundedly.

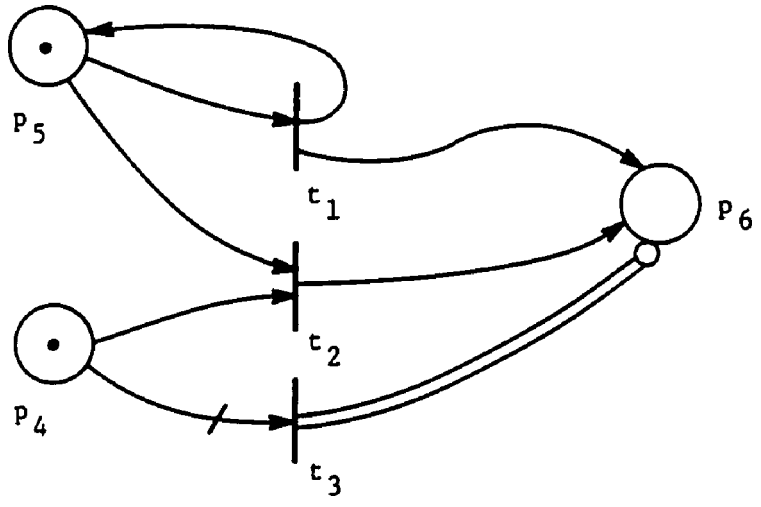


Figure 4.7.1

Sample EPN(II)

Definition 4.7.3

An Extended Petri Model (II) (EPM(II)) is a system $\mathit{EN} = (EN, M^o)$ where:

1. $EN = (T, P, I, O)$ is an EPN(I).
2. M^o is the initial marking of EN.

A Labelled EPM(II) is defined completely analogous to Definition 4.1.9. Similarly, the language families $\Lambda(EPM(II))$ and $\Lambda_o(EPM(II))$ are defined the same way as $\Lambda(EPM)$ and $\Lambda_o(EPM)$, respectively.

The following theorem states a rather interesting result regarding the EPM(II), namely that the introduction of token absorbers does not affect the language family Λ_o generated by the EPM. Thus:

Theorem 4.7.1

$\Lambda_o(EPM(II)) = \Lambda_o(EPM)$.

Proof: Obviously, any EPM is an EPM(II) as well. Consequently, $\Lambda_o(EPM) \subseteq \Lambda_o(EPM(II))$. Therefore, we only have to show that this inclusion relation is also true in opposite direction. The proof relates the EPM(II) and the EPM to the family of multi-counter E-acceptors operating in quasi-realtime. In order to preserve the continuity of the presentation, the detailed proof appears in Appendix C.

□

Theorem 4.7.2

$\{W - \{\lambda\} \mid W \in \Lambda(C\text{-}CPM)\} \subset \Lambda_o(C\text{-}CPM)$.

Proof: Consider an arbitrary Labelled EPM $\mathit{EN}_\Sigma = (\mathit{EN}, \Sigma, L)$ where $\mathit{EN} = (EN, M^o)$ and $EN = (T, P, I, O)$. Suppose $W = \Lambda(\mathit{EN}_\Sigma)$. By definition, $\lambda \in W$.

We shall construct a Labelled EPM(II) $\mathit{EN}'_\Sigma = (\mathit{EN}', \Sigma, L')$ and define a final marking M^f of EN'_Σ such that $\Lambda_o(\mathit{EN}'_\Sigma, M^f) = W - \{\lambda\}$. Let $\mathit{EN}' = (EN', M^{o'})$ where $EN' = (T', P', I', O')$. Our construct will preserve the places of P, that is for each place $p_j \in P$ we shall introduce a corresponding place p'_j in P'. Moreover, we shall introduce in P' a new place, let us denote it by Π. For each transition $t_k \in T$, we shall introduce in T' two distinct replacement transitions, t'_k and t''_k, and we shall set:

1. $I'(p'_j, t'_k) = I'(p'_j, t''_k) = I(p_j, t_k)$ for each $p'_j \in P'$, $p'_j \neq \Pi$.

$I'(\Pi, t'_k) = I'(\Pi, t''_k) = 1$

2. $O'(t'_k, p'_j) = O(t_k, p_j)$; $O'(t''_k, p'_j) = \xi$ for each $p'_j \in P'$, $p'_j \neq \Pi$
   $O'(t'_k, \Pi) = 1$; $O'(t''_k, \Pi) = 0$
3. $L'(t'_k) = L'(t''_k) = L(t_k)$

Thus, the transitions t'_k self-loop on Π while the transitions t''_k have Π only as an input place. Finally, let us define the markings $M^{o'}$ and M^f by:

1. $M^{o'}(p'_j) = M^o(p_j)$ for each place $p'_j \in P' - \{\Pi\}$ and $M^{o'}(\Pi) = 1$.
2. $M^f(p'_j) = M^f(\Pi) = 0$ for each place $p'_j \in P'$, $p'_j \neq \Pi$.

According to our construct, the replacement transitions t'_k and t''_k can fire only under the same conditions as the original transition t_k. Moreover, the firing of t'_k in some marking produces the same effect as the firing of t_k in the respective marking (except for the place Π). On the other hand, the firing of t''_k in any marking leads invariably to the final marking M^f. We note that M^f can be reached only through the firing of some transition t''_k and that in M^f all transitions of T' are disabled (no firing sequence can be continued from M^f). We also note that if p' is an inhibitor input place of t''_k then we can in fact set $O'(t''_k, p') = 0$ (rather than $O'(t''_k, p') = \xi$) without altering the effect produced by the firing of t''_k.

In order to simplify notations, let us define the injections $S' : T \to T'$ and $S'': T \to T'$ such that for each $t_k \in T$, $S'(t_k) = t'_k$ and $S''(t_k) = t''_k$. One can easily verify that if $\gamma = \delta t_k$ is a firing sequence of EN from M^o, i.e. $\gamma \in S(M^o)$ then there exists a firing sequence $\gamma' = S'(\delta)S''(t_k)$ of EN' [∓] and $\gamma' \in T(M^{o'}, M^f)$. Hence, each nonempty string $w \in W$ is also in $\Lambda_o(EN'_\Sigma, M^f)$. Let now $\gamma' = \delta' t''_k$ be a terminal firing sequence of EN'. According to our previous remarks δ' cannot contain any of the replacement transitions

---

[∓] The mappings S' and S'' can be naturally extended to firing sequences, i.e. if $\delta = t_{k1} \ldots t_{km}$ is a firing sequence in T^+ then $S'(\delta) = S'(t_{k1}) \ldots S'(t_{km})$ and $S''(\delta) = S''(t_{k1}) \ldots S''(t_{km})$.

t''_r. Clearly, there exists a firing sequence $\gamma = S'^{-1}(\delta')\ S''^{-1}(t''_k)$ of EN. Thus, $W - \{\lambda\} = \Lambda_o(EN'_\Sigma, M^f)$.

We can conclude that $\Lambda(EPM) \subseteq \Lambda_o(EPM(II))$, up to λ. Then, from Theorems 4.7.1, 4.5.1 and 4.1.1(a), we have:

$\{W - \{\lambda\} \mid W \in \Lambda(C\text{-}CPM)\} \subseteq \Lambda_o(C\text{-}CPM)$.

Some thought will show that the context-free language $W' = \{a^n b^n \mid n \geq 1\}$ is contained in $\Lambda_o(C\text{-}CPM)$ but $W' \cup \{\lambda\}$ is not contained in $\Lambda(C\text{-}CPM)$. Hence, the inclusion is proper.

□

Corollary 1

$\{W - \{\lambda\} \mid W \in \Lambda(EPM(II)\} \subset \Lambda_o(EPM)$

Proof: Let $EN_\Sigma = (EN, \Sigma, L)$ be a Labelled EPM(II) and let $W = \Lambda(EN_\Sigma)$. Using a construct analogous to that of Theorem 4.7.2 one can build a Labelled EPM(II) EN'_Σ such that $W - \{\lambda\} = \Lambda_o(EN'_\Sigma, M^f)$ where M^f is the zero marking. By Theorem 4.7.1, we have $\{W - \{\lambda\} \mid W \in \Lambda(EPM(II))\} \subseteq \Lambda_o(EPM)$. By the same argument as in Theorem 4.7.2 the inclusion is shown to be proper.

□

Regarding the language families $\Lambda(EPM(II))$ and $\Lambda(EPM)$, evidently $\Lambda(EPM) \subseteq \Lambda(EPM(II)$. Unfortunately, we do not know whether $\Lambda(EPM(II)) \subseteq \Lambda(EPM)$, as well. Appendix C contains further comments pertinent to $\Lambda(EPM)$ and $\Lambda(EPM(II))$.

Let us now briefly examine the Petri Nets with switches, disjunctive logic and token absorbers. The formal definition of this model is given in Appendix D.

It has been shown (Section 3.7 of [AGAR-75]) that the "zero testing" function of the inhibitor arcs can be simulated (in a λ-transition free manner) using disjunctive input logic. On the other hand, it is easy to see how switches and transitions with AND and EOR input and/or output logic can be simulated using inhibitor arcs and no λ-transitions. Since token absorbers were directly carried over to the EPM(II), Labelled Petri Net Models with switches, disjunctive logic and token absorbers can be transformed into language equivalent Labelled EPM(II)'s, in a λ-transition free manner. Thus, the EPM(II) and the Petri Net

Model with switches, disjunctive logic and token absorbers generate the same Λ and Λ_o language families. In view of Theorem 4.7.1 and Corollary 1 of Theorem 4.7.2 the Petri Net Model with switches, disjunctive logic and token absorbers and the C-CPM generate the same language family Λ_o while the Λ language family generated by the Petri Net Model with switches, disjunctive logic and token absorbers is a proper subset of Λ_o(C-CPM).

Section 4.8 THE RELATIONSHIP BETWEEN THE C-CPM AND THE EC-CPM

In this section, we shall examine the relationship between the various language families generated by the C-CPM and the EC-CPM.

Following characterization was given in [HACK-75].

The language families Λ(EPM) and Λ_o(EPM) are the closure under finite substitution, restriction and λ-free renaming of the language $\{-\}$ and Q, respectively Q_o.

Given two languages L_1 and L_2 over alphabets Σ_1 and Σ_2, respectively, the restriction operation Ⓡ on L_1 and L_2 is defined by:

$$L_1 \text{ Ⓡ } L_2 = (L_1 \Delta (\Sigma_2 - \Sigma_1)^*) \cap (L_2 \Delta (\Sigma_1 - \Sigma_2)^*).$$

The language Q_o mentioned above is defined as $Q_o = (P_o 0)^* P_o$ where 0 is a terminal symbol and P_o is the complete parenthesis language over the alphabet {+, -}, as generated by the context-free grammar:

$$S \rightarrow SS$$
$$S \rightarrow +S-$$
$$S \rightarrow \lambda$$

(S denotes the start symbol of the grammar.) Obviously, Q_o is a context-free language. Q denotes the language containing all the prefixes of Q_o, i.e., $Q = \text{PREF}(Q_o)$.

We note that in the characterization given above P_o and therefore Q_o are not Λ_o languages in the strict sense of the definition since they both contain λ. Thus, Λ_o(EPM) actually contains only the λ-free languages from this family.

If we extend the definition of the language family Λ_o(EC-CPM) in the same sense (by removing the restriction that the final color marking UM^f be different from the initial color marking UM^o) then we immediately have Λ(EPM) $\subseteq$ Λ(EC-CPM) and Λ_o(EPM) $\subseteq$ Λ_o(EC-CPM). This is so because

in this case $Q_o \in \Lambda_o$(EC-CPM) and $Q \in \Lambda$(EC-CPM). Moreover, the language families generated by the EC-CPM contain regular languages of the form $(\Sigma_1 - \Sigma_2)^*$ and are closed under concurrency, intersection, finite substitution and λ-free renaming. We stress that in the above characterization, the operation of finite substitution is restricted to one-symbol strings only (see [HACK-75]) and thus the closure under this operation extends to the family Λ(EC-CPM) as well.

In fact, one can use an alternate, direct method in order to reach to the same conclusion. Thus, let $\mathcal{EN}_\Sigma = (\mathcal{EN}, \Sigma, L)$ be an arbitrary Labelled EPM where $\mathcal{EN} = (EN, M^o)$ and $EN = (T, P, I, O)$ is the underlying EPN. We can easily construct a Labelled EC-CPM $\mathcal{ECP}_\Sigma = (\mathcal{ECP}, \Sigma, L')$ which generates the same language as $\mathcal{EN}_\Sigma$. Let $\mathcal{ECP} = (CN, UM^o)$ and $CN = (N, U(C), H)$. The GPN $N = (T', P', I', O')$ is defined as follows. For each place $p_j \in P$ we shall introduce in P' a corresponding place p_j' and for each transition $t_k \in T$ we shall introduce in T' a corresponding transition t_k' carrying the same label as t_k.

Let $C = (\{c_e, c_f, c_m, c_M\}, \leq)$ be a finite set of colors where the partial ordering $\leq$ satisfies the conditions of Section 2.5 (the Hasse diagram of C appears in Figure 4.8.2). The extension U(C) is the set of colors associated with CN. We shall use the colors c_e and c_f to encode the "empty" and "full" status of any place p_j of the original EPN. Thus:

1. If p_j is "empty" then the color bag of p_j' will be $< (c_e,1) >$.

2. If p_j is "full", i.e., p_j contains x tokens then the color bag of p_j' will be $< (c_f,2)^x, (c_e,1) >$. The initial and the final color marking of CN are formed from M^o and the final marking M^f of EN, respectively according to this rule.

Figure 4.8.1 is our example. The sample transition given there exhibits all possible interconnections of places to a transition in an EPN. The corresponding replacement transition is given in Figure 4.8.2. Using the encoding rule given above as a correspondence between the markings of EN and the color markings of CN, one can easily verify that t_k' can fire exactly under the same conditions as t_k and the effect of the firing of t_k' is the same as that produced by the firing of t_k.

The construct works for both language families Λ(EPM) and Λ_o(EPM). We can conclude that Λ(EPM) $\subseteq \Lambda$(EC-CPM) and Λ_o(EPM) $\subseteq \Lambda_o$(EC-CPM).

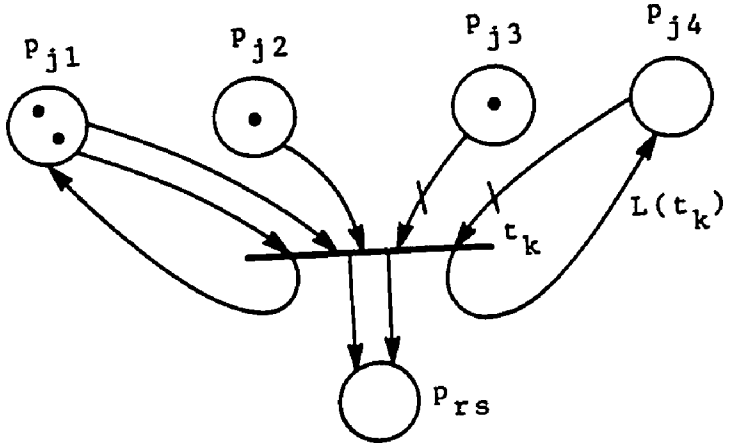


Figure 4.8.1

Sample Transition of an EPM

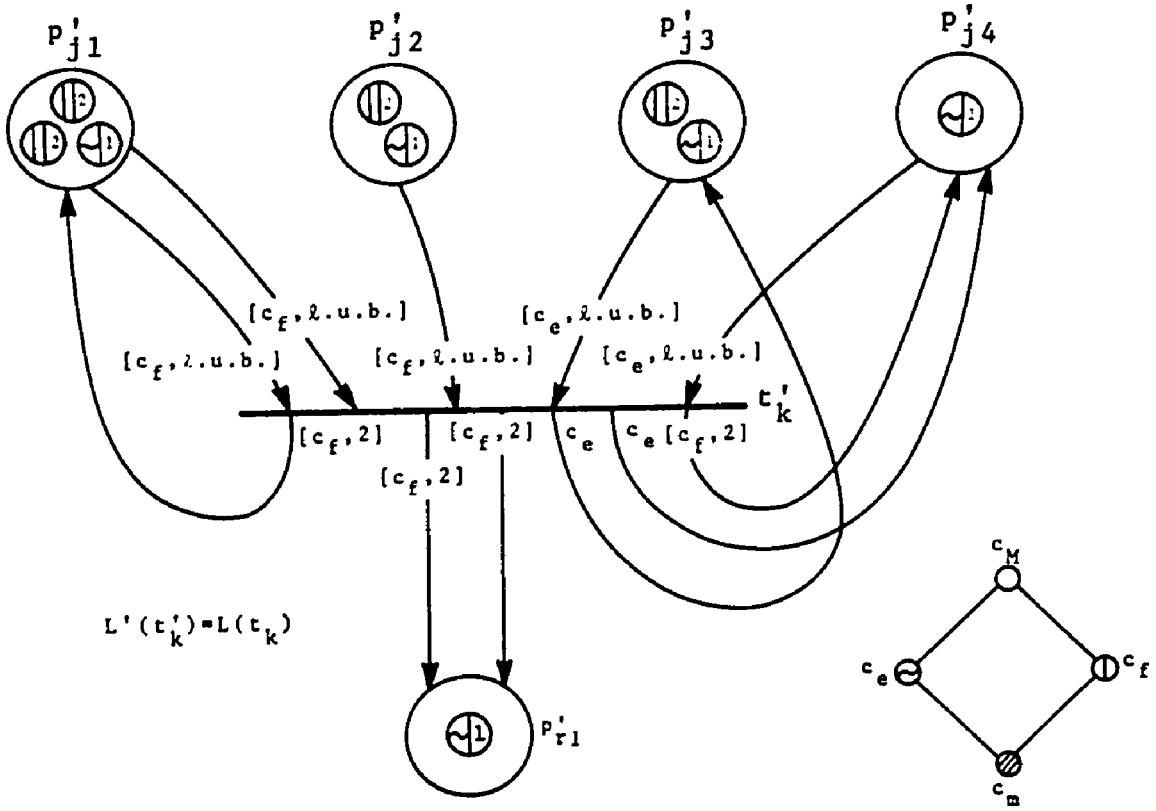


Figure 4.8.2

Replacement Transition for the Transition of Figure 4.8.1

Obviously, the same inclusion relations hold between Λ(C-CPM), Λ_o(C-CPM) and Λ(EC-CPM), Λ_o(EC-CPM), respectively.

Consider now the context-free language $W = \{ww^R \mid w \in \{a, b\}^+\}$. As noted in [HACK-75] the number of distinct intermediate markings needed to generate such palindromes of length n grows exponentially with n while the number of distinct markings reachable by a λ-transition free EPM during the generation of a string of length n grows as a polynomial function of n. Hence $W \notin \Lambda_o$(EPM) but since Λ_o(EC-CPM) contains all λ-free context-free languages, $W \in \Lambda_o$(EC-CPM). Therefore, Λ_o(EPM) and for that matter Λ_o(C-CPM) are properly contained in Λ_o(EC-CPM).

Note, however, that the above argument depends only on the length of the firing sequence and not on whether the respective firing sequence leads to a final marking or not. Therefore, the same argument can be applied to the Λ language families as well. Consider the language PREF(W') where $W' = \{wcw^R \mid w \in \{a, b\}^+\}$. If wcw^R is a string in W', with $|w| = n$, then all its prefixes, including the string wcw^R itself, are in PREF(W'). Suppose now that PREF(W') $\in \Lambda$(EPM). Then, there exists a λ-transition free EPM which can generate by means of firing sequences of length n a sufficiently large number N_n of intermediate markings in order to properly "remember" all the strings $w \in \{a, b\}^+$ of length n (N_n is exponentially related to n). By the same token this is, however, not possible and thus PREF(W') $\notin \Lambda$(EPM). Since PREF(W') $\in \Lambda$(EC-CPM), we have Λ(EPM) $\subset \Lambda$(EC-CPM).

As a bonus of the discussion presented above we obtain two more results regarding the language families generated by the EPM and the EC-CPM.

We have seen that Λ_o(EPM) $\subset \Lambda_o$(EC-CPM). By Theorem 4.7.2 Λ(EPM) $\subset \Lambda_o$(EPM), up to λ. Hence, Λ(EPM) $\subset \Lambda_o$(EC-CPM), up to λ.

Consider now the language families Λ_o(EPM) and Λ(EC-CPM). We already know that the context-free language $\{a^n b^n \mid n \geq 1\}$ for example, is in Λ_o(EPM) but, according to Chapter III, is not in Λ(EC-CPM). On the other hand, using the same argument as in the case of the family Λ(EPM), one can show that the language PREF(W') is not in Λ_o(EPM). Since PREF(W') $\in \Lambda$(EC-CPM), the language families Λ_o(EPM) and Λ(EC-CPM) are incomparable.

We can summarize these results as follows:

Theorem 4.8.1

a) Λ_o(C-CPM) $\subset$ Λ_o(EC-CPM)

b) Λ(C-CPM) $\subset$ Λ(EC-CPM)

c) Λ_o(C-CPM) and Λ(EC-CPM) are incomparable.

d) Λ(C-CPM) $\subset$ Λ_o(EC-CPM) (up to λ).

□

We have already mentioned that the Λ_o language family generated by Generalized Petri Nets properly contains the family of regular languages, up to λ ([PETE-73], [HACK-75]). Hence, all the λ-free regular languages are in Λ_o(C-CPM) as well. On the other hand, the languages in Λ_o(C-CPM) are deterministic context-sensitive. The relationship between Λ_o(C-CPM) and the families of regular ($\mathscr{R}$), context-free ($\mathscr{CF}$) and deterministic context-sensitive ($\mathscr{DCS}$) languages is exhibited in Figure 4.8.3 (where we have ignored for a moment the absence of the empty string λ from the languages in Λ_o(C-CPM)). Regarding the language family Λ(C-CPM), the following can be stated:

i. There are regular languages (e.g. a*) which are contained in Λ(C-CPM) but not all regular languages are in Λ(C-CPM) (e.g. a*c).

ii. The intersection between the family of strict context-free languages and Λ(C-CPM) is not empty (the context-free language $\{a^n b^m \mid 0 \leq m \leq n\}$ for example is in Λ(C-CPM)).

iii. There are languages in Λ(C-CPM) which are context-sensitive but not context-free (e.g. $\{a^n b^m c^r \mid 0 \leq r \leq m \leq n\}$). Note, however, that the context-sensitive language $\{a^n b^n c^n \mid n \geq 1\}$ is in Λ_o(C-CPM) but not in Λ(C-CPM).

The relationship between Λ(C-CPM) and the families $\mathscr{R}$, $\mathscr{CF}$ and $\mathscr{DCS}$ is also displayed in Figure 4.8.3.

In the previous sections of this chapter, the Λ_o language family generated by the C-CPM was shown to be equal to the language family Λ_o of the EPM, PPM, COPM and of the Petri Net Model with switches, disjunctive logic and token absorbers. Another common characteristic of these formal models is the fact that they can generate under unrestricted erasure the set of recursively enumerable languages. Hence, each language in Λ_o(EC-CPM) and/or Λ(EC-CPM) can be expressed as the

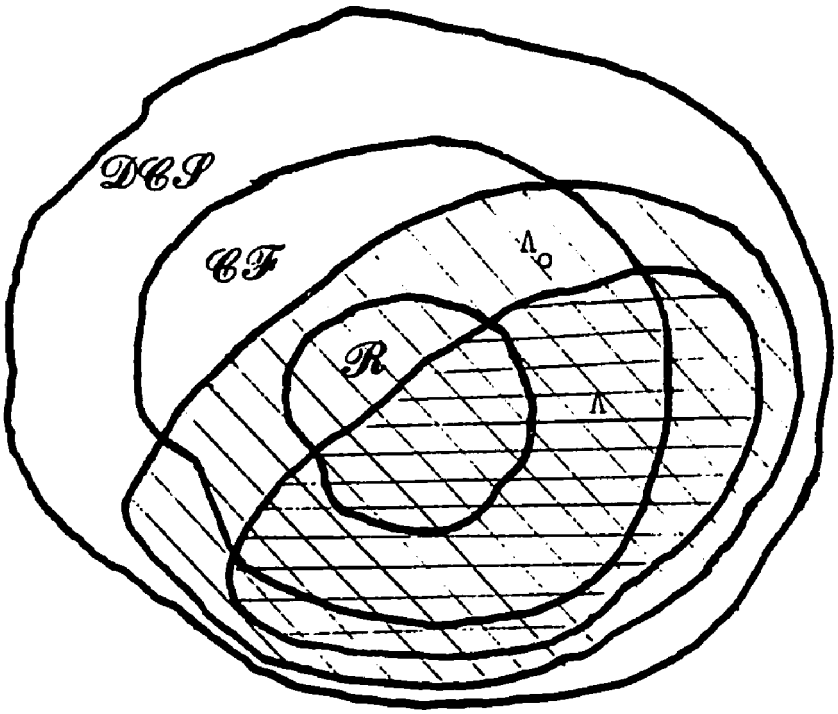


Figure 4.8.3

Relationship of Λ_o(C-CPM) and Λ(C-CPM) to Other Language Families

image under some erasing homomorphism of a language in Λ_o(C-CPM). Since the languages in Λ(EC-CPM) and Λ_o(EC-CPM) are still context-sensitive the question which naturally arises is whether the erasing needed to generate them is bounded and if so what are the necessary and sufficient bounds. This question has a certain practical significance. We recall that our interest in the study of the Λ and Λ_o language families was motivated by the fact that they provided a theoretical framework within which the "representation power" of various formal models of concurrent systems can be "measured" and "compared". In this context the C-CPM, EPM, PPM, COPM and the Petri Net Model with switches, disjunctive logic and token absorbers proved to be fully equivalent models with respect to the Λ_o language family generated but strictly weaker than the EC-CPM. Thus, it appears that there are synchronization systems whose coordination sequences can

be correctly modelled without the aid of λ-transitions by the EC-CPM but not by any of the other formal models mentioned above (we shall reexamine this statement in Chapter V). On the other hand, any such synchronization system can be modelled by the C-CPM and/or the other models equivalent to the C-CPM with the aid of λ-transitions. From this point of view, λ-transitions can be considered to correspond merely to internal operations of the model; they do not represent processes of the modelled system. By answering the question which we have raised earlier regarding the bound on the erasing homomorphisms required to generate Λ(EC-CPM) and Λ_o(EC-CPM) we shall also determine the bound on the number of internal (λ-transition) firings necessary to simulate arbitrary EC-CPM's by C-CPM's. This will provide insight to the degree of difficulty encountered by C-CPM's in simulating arbitrary EC-CPM's or, equivalently, to the degree of difficulty encountered by C-CPM's in modelling synchronization systems which can be modelled by EC-CPM's without the aid of λ-transitions.

Let us first examine how EPM's augmented with λ-transitions can generate the family of context-free languages. We shall assume that each context-free language W is generated by some context-free grammar G in a special type of Greibach normal form. Thus, G may contain only productions of the form:

$$A \rightarrow a$$

$$A \rightarrow aB$$

$$A \rightarrow aBC$$

where A, B , C are nonterminal symbols and a is a terminal symbol. Such a grammar can be found for any context-free language (see Exercise 4.11 of [HOPC-69] for example).

Let $G = (V_N, V_T, P, S)$ be a context-free grammar as described above. Our construct will define for any such grammar a Labelled EPM(I) $\mathcal{EN}_\Sigma = (\mathcal{EN}, \Sigma, L)$ which simulates leftmost derivations in the respective grammar. Suppose $\mathcal{EN} = (EN, M^o)$, where $EN = (T, P, I, O)$ is the underlying EPN(I). $\Sigma = V_T \cup \{d\}$ where d is a new symbol, not in V_T. Each such EPM(I) will be composed of three interconnected parts, which we shall informally call:

i. the control subnet

ii. the concatenation subnet

iii. the deletion subnet.

The control subnet will contain a distinct transition for each production of G and six places, denoted by p_o, p_s, p_t, p_m, p_d and p_r.

All transitions of the control subnet will have p_o as an input place.

Suppose $A_{q+1} \rightarrow aA_{q+2}$ is a production of G. Then, the control subnet will contain a transisition for it of the form presented in Figure 4.8.4. Similarly, if $A_{q+1} \rightarrow aA_{q+3}A_{q+2}$ is a production of G, the corresponding transition of the control subnet is displayed in Figure 4.8.5. Figure 4.8.6 exhibits a transition corresponding to a production of the form $A_{q+1} \rightarrow a$.

The places p_s and p_t are used to "remember" the nonterminal part of the sentential form obtained along any leftmost derivation in G. Suppose G has r - 1 variables. Then, each variable A_j can be uniquely represented by an integer $|A_j| = j$, where $1 \leq j \leq r - 1$. Let $A_{q+1}A_q \ldots A_1A_o$ be the nonterminal part of a sentential form. It also can be uniquely represented by an integer:

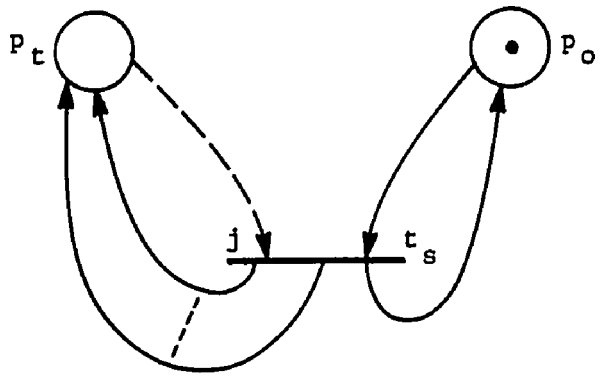
$$|A_o| \cdot r^{q+1} + |A_1| \cdot r^q + \ldots + |A_q| \cdot r^1 + |A_{q+1}| \cdot r^o = \sum_{j=0}^{q+1} |A_j| \cdot r^{q+1-j}$$

This encoding is rather common for automata with counters. In our case we shall have the marking of p_t be $|A_{q+1}|$ tokens and the marking of p_s be

$$\sum_{j=0}^{q} |A_j| \cdot r^{q-j}$$

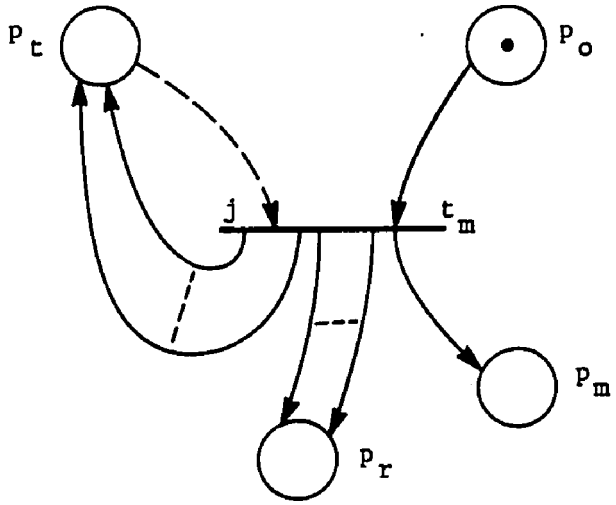
tokens, i.e., p_t always "remembers" the leftmost variable A_{q+1}. Consequently, at this stage only the transitions of the control subnet corresponding to productions having A_{q+1} on the left side can be enabled.

Suppose the next production to be applied is of the form $A_{q+1} \rightarrow aA_{q+2}$ After applying this production the nonterminal part of the sentential form becomes $A_{q+2}A_q \ldots A_1A_o$, i.e. only the leftmost variable is possibly changed. Also the terminal a is generated. The transition of Figure 4.8.4 performs exactly these operations, by replacing the $|A_{q+1}|$ tokens in p_t by $|A_{q+2}|$ tokens and generating a.



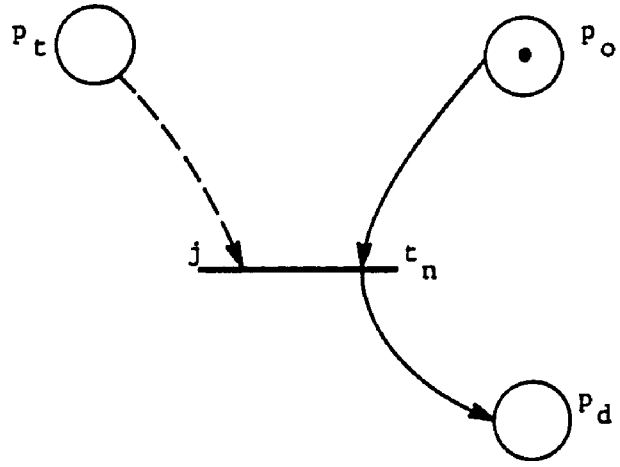
$L(t_s) = a$
$j = |A_{q+1}|$
$O(t_s, p_t) = |A_{q+2}|$

Figure 4.8.4
Sample Transition



$L(t_m) = a$
$j = |A_{q+1}|$
$O(t_m, p_t) = |A_{q+3}|$
$O(t_m, p_r) = |A_{q+2}|$

Figure 4.8.5
Sample Transition



$L(t_n) = a$
$j = |A_{q+1}|$

Figure 4.8.6
Sample Transition

Suppose now that we apply the production $A_{q+1} \rightarrow aA_{q+3}A_{q+2}$. The nonterminal part of the sentential form becomes $A_{q+3}A_{q+2}A_q \ldots A_1A_0$. Accordingly, we must change the marking of p_t to $|A_{q+3}|$ tokens and the marking of p_s to

$$|A_o| \cdot r^{q+1} + |A_1| \cdot r^q + \ldots + |A_q| \cdot r^1 + |A_{q+2}| \cdot r^o =$$

$$= \left(\sum_{j=0}^{q} |A_j| \cdot r^{q-j} \right) \cdot r + |A_{q+2}| \quad \text{tokens.}$$

The transition of Figure 4.8.5 initiates these operations. Thus, it replaces the $|A_{q+1}|$ tokens in p_t by $|A_{q+3}|$ tokens, it consumes the token in p_o (and thus disables all the transitions of the control subnet), it deposits $|A_{q+2}|$ tokens in p_r and it "starts" the concatenation subnet by placing a token in p_m. The concatenation subnet will duplicate the tokens in p_s r times and will add to them the $|A_{q+2}|$ tokens stored in p_r, thus simulating the concatenation of A_{q+2} to the nonterminal part of the sentential form.

Let us now assume that the production $A_{q+1} \rightarrow a$ is applied. In this case the nonterminal part of the sentential form becomes $A_qA_{q-1} \ldots A_jA_0$, i.e. the leftmost variable is deleted. Consequently, the marking of p_t must become $|A_q|$ tokens and the marking of p_s must become

$$\sum_{j=0}^{q-1} |A_j| \cdot r^{q-1-j} \quad \text{tokens.}$$

The transition of Figure 4.8.6 partly performes these operations by consuming the $|A_{q+1}|$ tokens in p_t and placing a token in p_d. In this way the deletion subnet is activated and it will divide the number of tokens in p_s by r. The remainder will be deposited in p_t and the quotient will be placed back in p_s.

Let us now present the concatenation and deletion subnets. Our constructs can be viewed as extensions of the multiplication and division nets used in [AGAR-75].

A sample concatenation subnet is displayed in Figure 4.8.7. Let us assume that the concatenation subnet has been activated by the

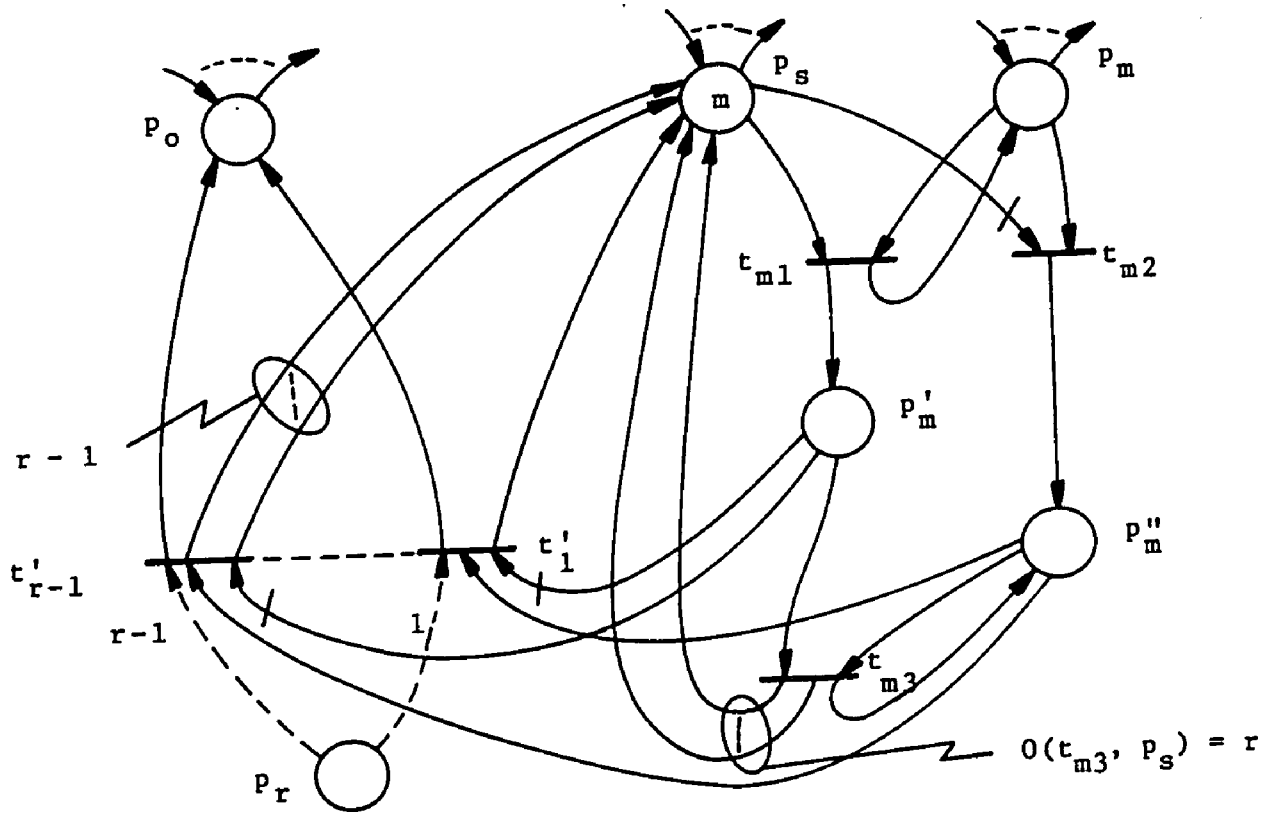


Figure 4.8.7

Concatenation Subnet

transition of the control subnet corresponding to some production $A_{q+1} \rightarrow aA_{q+3}A_{q+2}$ of G. Let us also assume that at that stage, p_s contained m tokens. Transition t_{m1} transfers the m tokens to the place p'_m while t_{m3} performs the actual multiplication, i.e., t_{m3} deposits $m \cdot r$ tokens into p_s. Transitions $t'_1, \ldots, t'_{r-1}$ are used to "stop" the concatenation subnet after the multiplication has been performed. This group of transitions contains a distinct transition for each variable of G. Thus, t'_j corresponds to the variable A_j, $1 \leq j \leq r - 1$ and $I(p_r, t'_j) = (\sigma, |A_j|) = (\sigma, j)$. Due to the positive testing arcs at the most one of these transitions can be enabled at a time, in this case the transition corresponding to A_{q+2}. By firing this transition, the $|A_{q+2}|$ tokens will be removed from p_r and added

to the place p_s, thus completing the concatenation operation. At the same time, the concatenation subnet becomes disabled and the control subnet is reactivated (each of these "stop" transitions restores the token in p_o). We note that the concatenation subnet operates rather deterministically. The inhibitor arcs and the positive testing arcs ensure that the execution of the concatenation subnet ends only after the concatenation operation has been fully executed. One can easily see that exactly $2 \cdot m + 2$ transition firings are required by each execution of the concatenation subnet.

We shall assign the label d to all transitions of the concatenation subnet.

Figure 4.8.8 displays a sample deletion subnet. Transition t_{d1} transfers the m tokens from p_s to p'_d while t_{d3} performs the actual division. Note that in our encoding each division operation must have a remainder, and the remainder must be greater than or equal to 1. The transitions $t''_1, \ldots, t''_{r-1}$ are used to "stop" the deletion subnet. For each variable of G there exists a distinct "stop" transition in this group. Thus, t''_j is the transition corresponding to the nonterminal A_j and $I(p'_d, t''_j) = (\sigma, |A_j|) = (\sigma, j)$ for each j, $1 \leq j \leq r - 1$. As in the case of the concatenation subnet, at the most one of these transitions can become enabled at a time. Moreover, these transitions can become enabled only when the marking of p'_d is positive but strictly less than r (in which case t_{d3} is disabled). The purpose of these "stop" transitions is to transfer the remainder of the division operation from p'_d to p_t. At the same time they disable the deletion subnet and reactivate the control subnet. All transitions of the deletion subnet carry the label d.

One can easily verify that each execution of the deletion subnet requires exactly m + quotient (m/r) + 2 transition firings. In fact, if we complicate the structure of the concatenation subnet, the number of transition firings necessary to simulate the concatenation operation can be lowered to this bound as well. It is more important, however, to notice that the number of transition firings required by an execution of either the concatenation or the deletion subnet will always have an upper bound of the form $c \cdot m$, for some constant $c > o$.

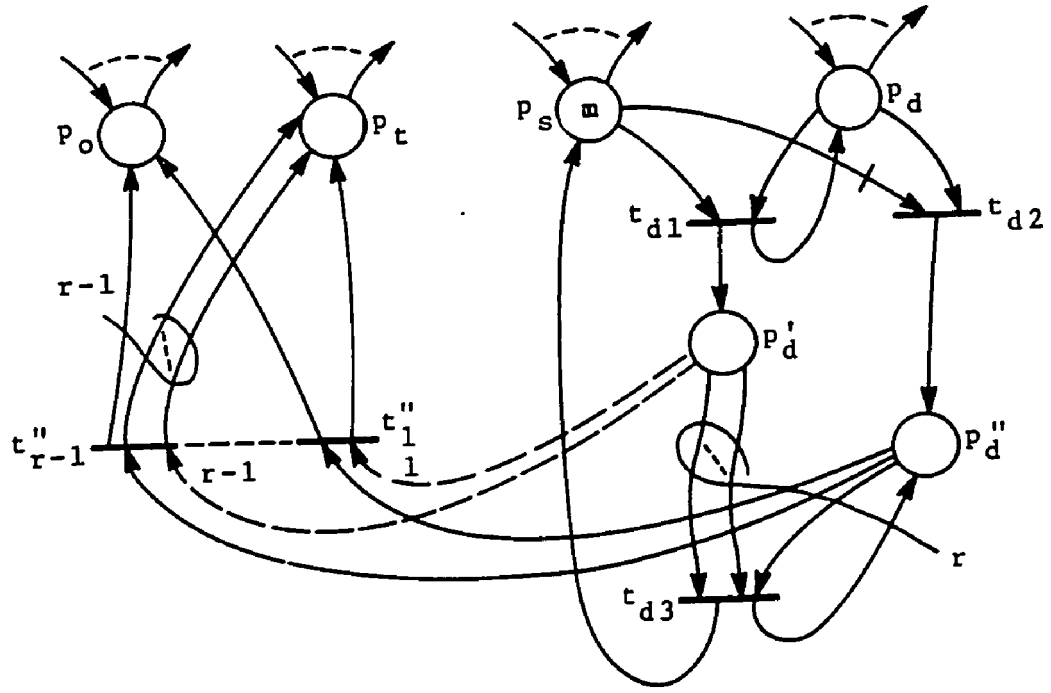


Figure 4.8.8

Deletion Subnet

Let us now put together the three subnets examined above. Figure 4.8.9 displays schematically the typical structure of a Labelled EPM(I) simulating the leftmost derivations of some context-free grammar G. In the initial marking all places are empty, except p_o which contains a token and p_t which contains $|S|$ tokens, where S is the start symbol of G.

We note that a derivation in the grammar G ends when the nonterminal part of the respective sentential form becomes empty. Moreover, the production used at the last step in the derivation must be of the

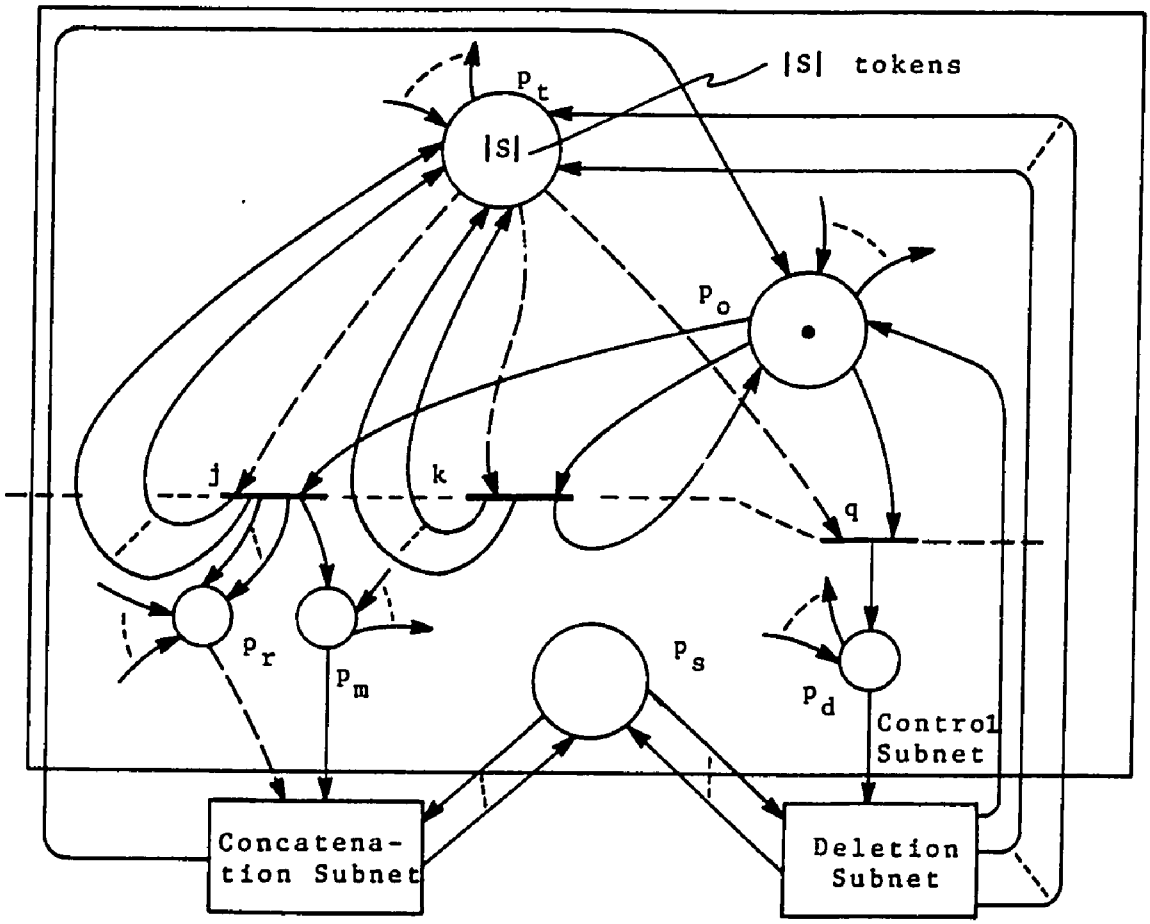


Figure 4.8.9

Generation of a Context-Free Language

form $A_{q+1} \rightarrow a$. Hence, we shall have in the final marking all places empty, except p_d which will contain a token (recall that we are generating an Λ_o language).

Let now $h : \Sigma^* \rightarrow V_T^*$ be the erasing homomorphism defined by $h(a) = a$, for all $a \in V_T$, and $h(d) = \lambda$. If W is the context-free language generated by G and W' is the language generated by EN_Σ then by our construct $h(W') = W$.

For languages in Λ(EC-CPM) we must be able to generate the empty string λ as well. For this purpose we shall assume that the start symbol S of G does not appear on the right side of any production of G. If $\lambda \in W$ then G must contain the production $S \rightarrow \lambda$. For this production we shall introduce in the control subnet a transition of the form presented in Figure 4.8.6, labelled d.

Let us now determine the bound of the erasing homomorphism h. For this purpose we shall make use of the following definition of [BOOK-70a]:

> if $h : \Gamma^* \rightarrow \Delta^*$ is a homomorphism, $L \subseteq \Gamma^*$, and f is a function such that for some $k > 0$ and all $w \in L$, $|w| \leq k \cdot f(|h(w)|)$, then h is f-bounded on L (equivalently, we shall say that h is bounded on L by f). Here, f is assumed to be a real-valued function of a single real variable.

In view of the previous discussion, we can assume for simplicity that the number of transition firings required by each execution of a concatenation or deletion subnet is bounded by $2 \cdot m$. If the nonterminal part of a sentential form is $A_{q+1}A_q \ldots A_1A_o$, then:

$$m = \sum_{j=0}^{q} |A_j| \cdot r^{q-j} < r \cdot \sum_{j=0}^{q} r^{q-j} < r \cdot r^{q+1} \quad \text{(note that } r > 1\text{)}$$

Thus, the number of consecutive d symbols generated for each concatenation or deletion operation is upper-bounded by $2 \cdot r^{q+2}$.

On the other hand, due to the special form of the grammars G under consideration, if $w \in W$ then each derivation of w in G must have exactly $|w|$ steps. Also, the nonterminal part of the sentential form along any such derivation cannot contain more than $\left\lceil \frac{|w|}{2} \right\rceil$ variables, where $\left\lceil \frac{|w|}{2} \right\rceil$ denotes the smallest integer larger than or equal to $\frac{|w|}{2}$.

Hence, $q \leq \left\lceil \frac{|w|}{2} \right\rceil$. We can conclude that:

Lemma 4.8.1

If W is a context-free language then there exists a language $W' \in \Lambda_o(EPM)$, a homomorphism h and a constant $c > 1$ such that $W = h(W')$ and h is bounded on W' by $f(x) = c^x$.

□

Let us now examine how EPM's augmented with λ-transitions can generate the set of quasi-realtime languages. As we have already mentioned, according to [BOOK-70b], any quasi-realtime language can be represented as the length-preserving homomorphic image of the intersection of three context-free languages. Suppose now that $G_1 = (V_N^1, V_T, P_1, S_1)$ and $G_2 = (V_N^2, V_T, P_2, S_2)$ are two context-free grammars over the same terminal alphabet V_T. Let us also assume that G_1 and G_2 are specified in the special type of Greibach normal form mentioned earlier in this section. Let then W_1 and W_2 denote the languages generated by G_1 and G_2, respectively and suppose $w = a_1 a_2 \cdots a_{m-1} a_m$ is a string in $W_1 \cap W_2$. Hence, we must have the leftmost derivations:

$$S_1 \overset{}{\Rightarrow}_{G_1} a_1 A_1^1 \gamma_1^1 \overset{}{\Rightarrow}_{G_1} a_1 a_2 A_2^1 \gamma_2^1 \overset{}{\Rightarrow}_{G_1} \cdots \overset{}{\Rightarrow}_{G_1} a_1 a_2 \cdots a_{m-1} A_{m-1}^1$$

$$\overset{}{\Rightarrow}_{G_1} a_1 a_2 \cdots a_{m-1} a_m$$

$$S_2 \overset{}{\Rightarrow}_{G_2} a_1 A_1^2 \gamma_1^2 \overset{}{\Rightarrow}_{G_2} a_1 a_2 A_2^2 \gamma_2^2 \overset{}{\Rightarrow}_{G_2} \cdots \overset{}{\Rightarrow}_{G_2} a_1 a_2 \cdots a_{m-1} A_{m-1}^2$$

$$\overset{}{\Rightarrow}_{G_2} a_1 a_2 \cdots a_{m-1} a_m .$$

Each derivation of w in either G_1 or G_2 must have exactly $|w| = m$ steps. Moreover, at each step k of such a derivation, $1 \leq k \leq m$, the production employed at that step must have the terminal symbol a_k on the right side, where a_k is the k-th symbol from the left of w.

These observations can be extended to the intersection of three or more context-free languages. Our construct for the generation of quasi-realtime languages is based on the construct presented earlier in connection with the generation of context-free languages and will employ the remarks on the intersection of context-free languages given above.

Suppose we are given the context-free grammars G_1, G_2 and G_3 which generate the languages W_1, W_2 and W_3, respectively. Suppose also that EN_{Σ_1}, EN_{Σ_2} and EN_{Σ_3} are the Labelled EPM(I)'s obtained by our previous construct which generate W_1, W_2 and W_3, respectively under bounded erasure. Let EN_{Σ} denote the Labelled EPM(I) which we are constructing.

We shall preserve the places of EN_{Σ_1}, EN_{Σ_2} and EN_{Σ_3} as well as the respective concatenation and deletion subnets. In addition we shall combine the transitions of the particular control subnets in such a way that the resulting Labelled EPM(I) generate only the strings common to W_1, W_2 and W_3. From this point of view, our construct is similar to the construct used in Section 3.2 in order to prove closure under the operation of intersection.

Let a be a terminal symbol common to G_1, G_2 and G_3 and suppose t_a^1, t_a^2 and t_a^3 are transitions of the control subnets of EN_{Σ_1}, EN_{Σ_2} and EN_{Σ_3}, respectively each carrying the label a. We shall replace this combination of three transitions by a unique transition of the control subnet of EN_{Σ}, let us denote it by t_a. The firing of the transition t_a will have the same effect as the combined firings $t_a^1 t_a^2 t_a^3$ and is assigned the common label a. A sample transition t_a is exhibited in Figure 4.8.10. This reduction will be performed for each possible combination of three equally-labelled transitions, each belonging to a different control subnet. Certainly, transitions whose labels are not terminals common to all three grammars G_1, G_2 and G_3 will be discarded since they cannot possibly lead to label sequences in $W_1 \cap W_2 \cap W_3$.

The initial and the final markings of EN_{Σ} are defined as $M_1^o \nabla M_2^o \nabla M_3^o$ and $M_1^f \nabla M_2^f \nabla M_3^f$, respectively where M_i^o and M_i^f denote the initial and final marking of EN_{Σ_i}, respectively, i = 1, 2, 3 (the direct sum operation can easily be particularized to the "one-color" markings of the EPM(I)).

We note that a transition of the control subnet of EN_{Σ}, for example the transition t_a of Figure 4.8.10, may simultaneously "start" up to three of the six distinct concatenation and deletion subnets incorporated in EN_{Σ}. Nevertheless, the control subnet of EN_{Σ} is reactivated only after all the "active" concatenation and/or deletion subnets have terminated their independent executions. Let us also

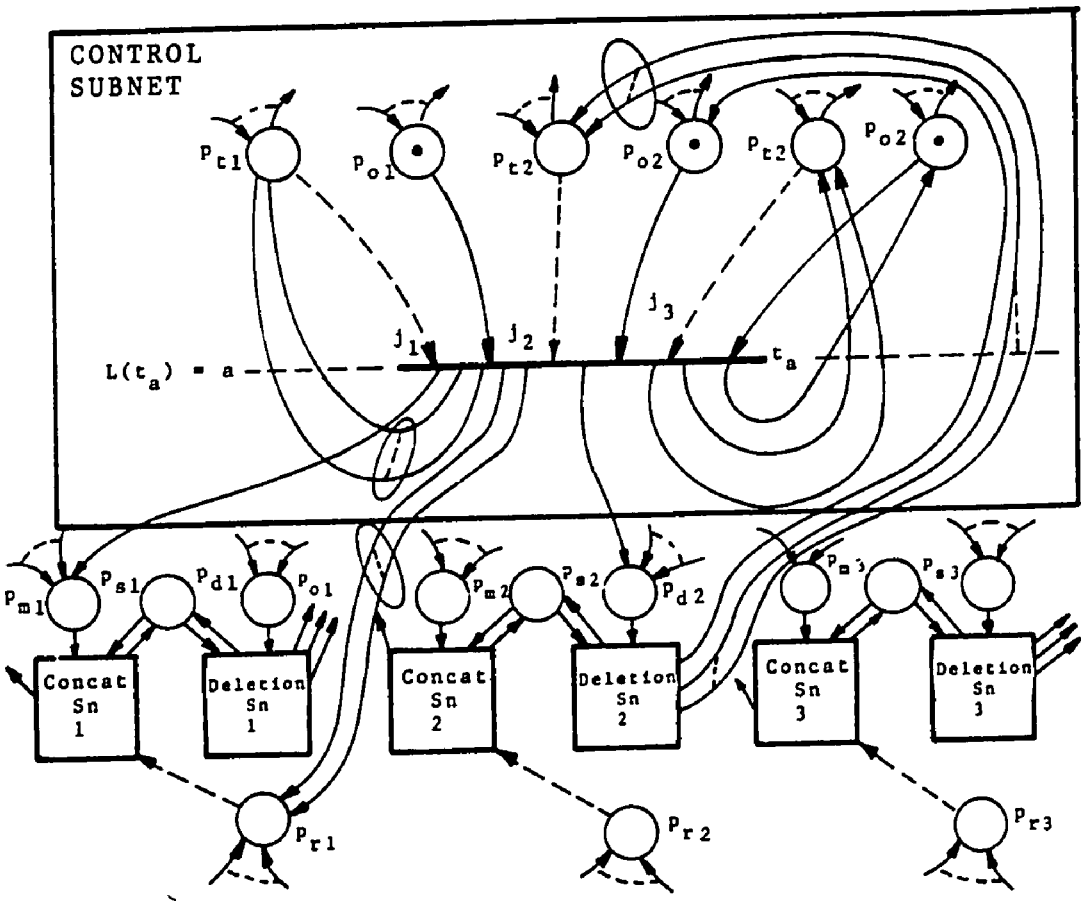


Figure 4.8.10

Sample Transition

recall that each transition t_a^i of the control subnet of EN_{Σ_i} considered in the definition of t_a, i = 1, 2, 3, has been introduced for some production of the grammar G_i. In this regard, EN_Σ may be viewed as a synchronous simulation of three distinct leftmost derivations (in G_1, G_2 and G_3, respectively) "running" in parallel. This is possible only because of the special characteristics of the derivations in either grammar of the strings in $W_1 \cap W_2 \cap W_3$, which we have mentioned earlier.

If W' is the language generated by EN_Σ and $h : \Sigma^* \to (V_T^1 \cap V_T^2 \cap V_T^3)^*$ is a homomorphism defined by h(d) = λ and h(a) = a, for all $a \in V_T^1 \cap V_T^2 \cap V_T^3$ (V_T^i denotes the set of terminal symbols of G_i, i = 1,2,3 and d denotes the label common to all transitions of the concatenation and deletion subnets incorporated in EN_Σ) then $h(W') = W_1 \cap W_2 \cap W_3$. This is so because if $w \in W_1 \cap W_2 \cap W_3$ then there exists at least one leftmost derivation of w in each of the grammars G_1, G_2 and G_3. Therefore there exists a terminal firing sequence of EN_Σ which yields w under the erasing homomorphism h. It can also be verified that conversely, if there exists a terminal firing sequence of EN_Σ and w' is the associated label sequence then there exists a (leftmost) derivation in G_1, G_2 and G_3 for w = h(w').

We note that if $w \in W_1 \cap W_2 \cap W_3$ then the length of the nonterminal part of the sentential form obtained after k steps of some derivation of w in G_1, $k < |w|$, does not have to be equal to the length of the nonterminal part of the sentential form obtained after the same number of steps along a derivation of w for example in G_2. It is important to observe, however, that the nonterminal parts of the respective sentential forms must simultaneously become empty.

Let us now consider the empty string λ. If $\lambda \in W_1 \cap W_2 \cap W_3$ then the production $S_i \to \lambda$ must be contained in G_i, i = 1, 2, 3, where S_i is the start symbol of G_i. Our construct is applied to the transitions introduced for these productions in the corresponding control subnets of EN_{Σ_1}, EN_{Σ_2} and EN_{Σ_3} and the resulting transition of EN_Σ is is assigned the label d.

This completes the description of the construct employed for the generation of the intersection of three arbitrary context-free languages W_1, W_2 and W_3. In order to generate quasi-realtime languages we must be capable of generating the image of $W_1 \cap W_2 \cap W_3$ under arbitrary

length-preserving homomorphisms as well. This can, however, be easily achieved by simply relabelling the transitions of the control subnet of EN_Σ.

Let us now examine the bound of the erasing homomorphism h. We shall again assume that the number of transition firings required by an execution of a concatenation or deletion subnet is bounded by $2 \cdot m_i$ where m_i is the number of tokens in the corresponding place p_{si} at the beginning of the execution of the subnet ($i = 1, 2, 3$). Consequently, the number of d symbols generated by the EPM(I) of Figure 4.8.10 between two consecutive firings of transitions from the respective control subnet has the upper bound:

$$2 \cdot m_1 + 2 \cdot m_2 + 2 \cdot m_3 < 2 \cdot (r_1^{q_1+2} + r_2^{q_2+2} + r_3^{q_3+2})$$

Since each derivation of some string $w \in W_1 \cap W_2 \cap W_3$ in either grammar G_1, G_2 or G_3 has exactly $|w|$ steps it follows that $q_i \leq \left\lceil \frac{|w|}{2} \right\rceil$, $i = 1, 2, 3$. Let $R = \max\{r_1, r_2, r_3\}$. Then:

$$2 \cdot m_1 + 2 \cdot m_2 + 2 \cdot m_3 < 6 \cdot R^{\left\lceil \frac{|w|}{2} \right\rceil + 2}$$

We can therefore conclude that:

Lemma 4.8.2

If W is a quasi-realtime language then there exists a language $W' \in \Lambda_o(\text{EPM})$, a homomorphism h and a constant $c > 1$ such that $W = h(W')$ and h is bounded on W' by $f(x) = c^x$.

□

Theorem 4.8.2

If $W \in \Lambda(\text{EC-CPM})$ or $W \in \Lambda_o(\text{EC-CPM})$ then there exists a language $W'' \in \Lambda_o(\text{EPM})$, a homomorphism h'' and a constant $d_2 > 1$ such that $W = h''(W'')$ and h'' is bounded on W'' by $f''(x) = d^{x^2}$.

Proof: By Corollary 1 of Theorem 3.4.2 $\Lambda(\text{EC-CPM}) \subseteq \text{TIME}(x^2)$ and $\Lambda_o(\text{EC-CPM}) \subseteq \text{TIME}(x^2)$. Consequently, by Theorem 1.7 of [BOOK-70a], if $W \in \Lambda(\text{EC-CPM})$ or $W \in \Lambda_o(\text{EC-CPM})$ then there exists a quasi-realtime language W' and a homomorphism h such that $W = h(W')$ and h is bounded on W' by $f(x) = x^2$.

But if W' is a quasi-realtime language then by Lemma 4.8.2 there exists a language $W'' \in \Lambda_o(EPM)$, a homomorphism h' and a constant $c > 1$ such that $W' = h'(W'')$ and h' is bounded on W'' by $f'(x) = c^x$.

Hence, $W = h(W') = h(h'(W'')) = h''(W'')$. We must also have the positive integer constants k and k' such that for all $w \in W''$:

$$|w| \leq k' \cdot f'(|h'(w)|) = k' \cdot c^{|h'(w)|}$$

and

$$|h'(w)| \leq k \cdot f(|h(h'(w))|) \leq k \cdot |h''(w)|^2$$

Thus,

$$|w| \leq k' \cdot c^{k \cdot |h''(w)|^2} = k' \cdot d^{|h''(w)|^2}, \text{ where } d = c^k > 1.$$

We can conclude that h'' is bounded on W'' by the function $f''(x) = d^{x^2}$, for some constant $d > 1$.

□

Note: Consider the languages $W' = \{wcw^R \mid w \in \{a, b\}^+\}$ and PREF(W') in Λ_o(EC-CPM) and Λ(EC-CPM), respectively. Using an argument similar to that of Theorem 1.3 of [FISH-68] in combination with the definition of bounded erasing homomorphism one can show that W' and PREF(W') can be generated as images under bounded erasing homomorphisms of languages in Λ_o(EPM) only if the amount of erasing is bounded by functions of the form $f(x) = c^x$, $c > 1$. Hence, the exponential bound is a necessary bound.

CHAPTER V

MODELLING WITH THE C-CPM AND THE EC-CPM

Section 5.1 MODELLING WITH THE C-CPM

In Chapter IV we have shown the following formal models:

i. Priority Petri Model
ii. Extended Petri Model
iii. C-Colored Petri Model
iv. Coordination Petri Model
v. Petri Net Model with switches, disjunctive logic and token absorbers

as well as various variants of some of these models (introduced by us in order to simplify certain constructs) to be completely equivalent from the point of view of the Λ_o-language family generated. This fact is rather surprising if one considers that these models were introduced both independently and for distinct purposes. Thus, each of the models listed above was defined in order to offer natural representation and modelling ease with respect to a certain class of synchronization systems of interest to the respective authors.

From this point of view, the study presented in Chapter IV provides insight to the capabilities as well as to the limitations common to all these models. On the other hand, it is our belief that the equivalence results given in Chapter IV do not diminuish the usefulness of any of these models, mainly because, as we have already mentioned, their principal merit is their adequacy for the modelling of a certain class of control structures. The latter statement is supported by the considerable complexity of some of the conversion procedures presented in Chapter IV by means of which instances of the models in the above list can be converted into each other in a language preserving manner. Evidently, the numerous additional places and/or transitions which eventually have to be introduced for this purpose do not contribute to the clarity and simplicity of the representation offered by the respective models. Thus, even if any control structure representable by one of the models mentioned above is also representable by any other model in the list, the representation offered by some models may be more

cumbersome than the representation offered by other models, for the same coordination system. This fact may be considered a drawback from the pragmatic point of view.

In the following three sections we shall present a few synchronization situations which can be modelled in a natural and simple way by the C-CPM. These examples (as well as the other modelling examples to be presented in the remainder of this chapter) were selected with the following statement from [PATI-70] in mind:

"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."

We note that in view of the proper inclusion relation between Λ(C-CPM), Λ_o(C-CPM) and Λ(EC-CPM), Λ_o(EC-CPM), respectively, the examples to be presented next can be modelled in a λ-transition free manner by the EC-CPM as well. Nevertheless, the C-CPM offers in these cases a natural, clear and correct representation and therefore the additional complexity of the EC-CPM is not needed.

Section 5.2 A PRODUCER-CONSUMER SYNCHRONIZATION PROBLEM WITH DYNAMIC PRIORITY HIERARCHY

Consider the following refinement of the producer-consumer synchronization problem presented in Section 2.4. Suppose the coordination system contains two buffers, B_1 and B_2. Two producer processes, P_{i1} and P_{i2}, and a consumer process, C_i, are connected to each buffer B_i, $i = 1, 2$. A producer process can be activated at any time and it produces and deposits an item in the associated buffer. A consumer process becomes active only if the associated buffer is nonempty. Following rules govern the access of the consumer processes to the respective buffers. Consumer process C_i may consume from B_i items deposited there by P_{i1} only if B_i does not contain any item produced by P_{i2}. Moreover, it is assumed that C_1 and C_2 can consume items from the corresponding buffers only through a channel with capacity 1, i.e., through a channel which can accomodate only one consumer process at a time. Following priority rule determines which consumer process can seize the channel in case both C_1 and C_2 are simultaneously active:

- if C_1 chooses to consume an item produced by P_{12} while C_2 attempts to consume an item produced by P_{21} then either consumer process can gain control over the channel.

- in all other cases C_1 has priority over C_2. This producer-consumer synchronization problem is represented schematically in Figure 5.2.1. The shared channel plays the same role as the decider module of the original producer-consumer synchronization problem but a different priority hierarchy is imposed on the competing processes. In fact, we note that the present system incorporates two interrelated but distinct priority hierarchies:

i. The consumer processes must select items from the respective buffers according to a fixed priority assignment to the parent producer processes.

ii. Control over the channel is obtained in accordance to a varying priority hierarchy among the consumer processes.

These coordination aspects can be modelled in the C-CPM in a natural way mainly due to the following specific features of this model:

- the rule used for the selection of enabling colors.

- the priority of a transition depends on the current color marking and on the particular selection of the enabling colors (is not fixed "a priori").

Figure 5.2.2 exhibits the C-CPM representation of this producer-consumer synchronization system. There, transitions t_1, t_2, t_3 and t_4 model the producer processes P_{11}, P_{12}, P_{21} and P_{22}, respectively while the transitions t_5 and t_6 model the consumer processes C_1 and C_2, respectively. The places p_7 and p_8 represent the buffers B_1 and B_2, respectively while p_9 can be viewed as the implementation of the shared channel. It can easily be verified that the firing sequences of the C-CPM of Figure 5.2.2 model correctly the coordination sequences which can be executed by the producer-consumer system under consideration. Using the C-CPM one can model correctly even more sophisticated producer-consumer synchronization problems of this type, involving several producer and/or consumer processes and more complicated priority hierarchies (as long as the respective priorities can be encoded as finite lattices of colors).

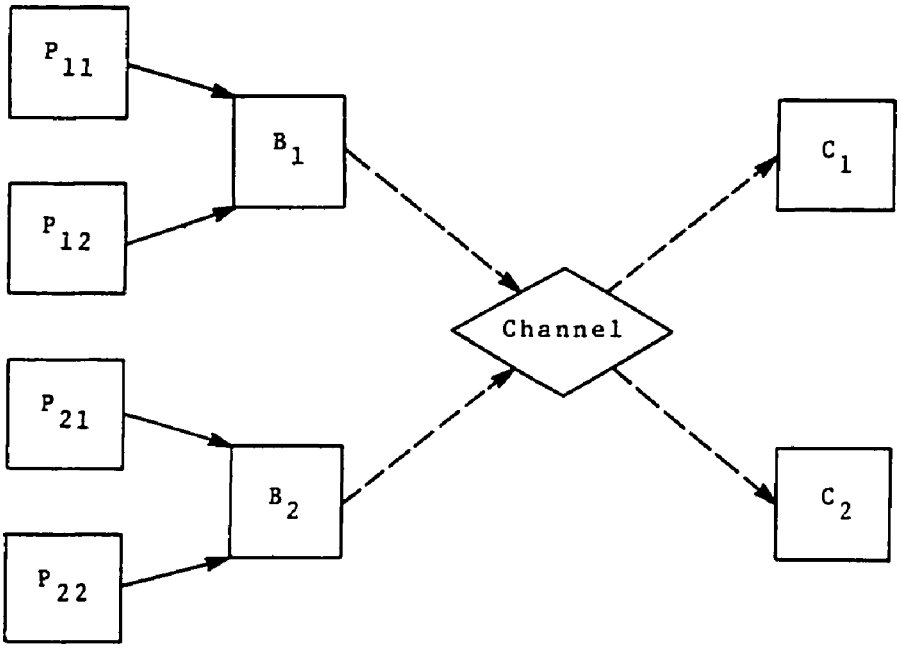


Figure 5.2.1

Producer-Consumer Synchronization System with Dynamic Priority Hierarchy

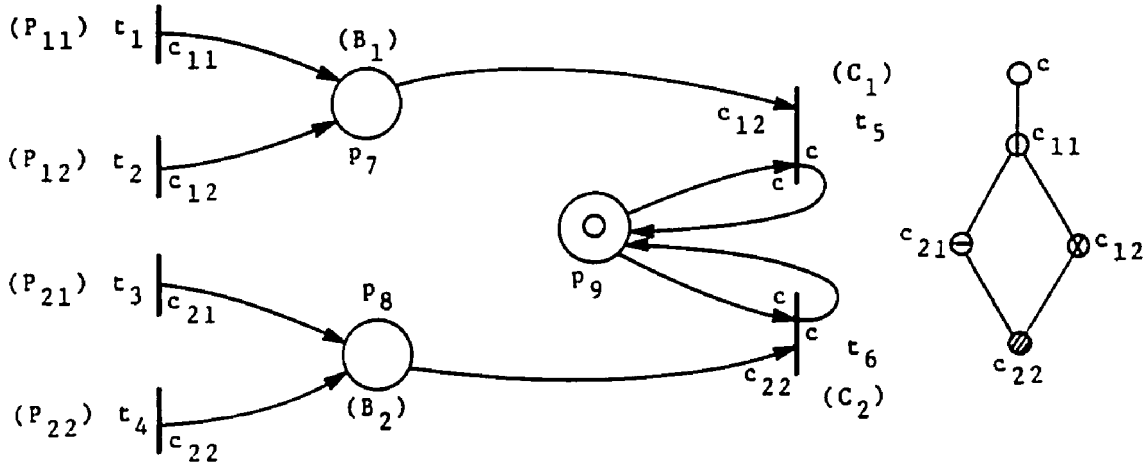


Figure 5.2.2

C-CPM of the Producer-Consumer Synchronization System of Figure 5.2.1

This type of synchronization problems can actually be encountered in computing environments. Consider for example the following multiprocessor computing system configuration. Suppose the system contains k memory units and m processors, $m \geq k$. The set of processors is partitioned into k classes such that the members of each class may access a single memory unit, designated to that particular class. Moreover, a memory unit may not be accessed by processors from different classes. We shall assume that all processors and memory units are connected to a unique bidirectional bus which can accomodate only one data transfer between a processor-memory unit pair at a time.

In order to solve conflicts for access to the associated memory unit, a fixed priority hierarchy is imposed among the processors within each class. Since processors from various classes may simultaneously attempt to gain control over the shared bus, another priority hierarchy among the processors has been designed for solving this type of conflicts. We can assume that a separate arbitration logic determines within each processor class which of the processors requesting to access the respective memory unit may first proceed to do so. Another, "global" arbitration logic polls the processors designated by the particular arbitration logics corresponding to each processor class and decides which processor can actually seize the bus. After the respective processor has executed the data transfer the bus is released and the pending access requests are reconsidered. In this perspective, there does not exist a fixed priority hierarchy among the k processor classes or, for that matter, among the k memory units. Actually, the priority of a class is determined for each "poll" by the priority of the particular processor from that class designated by the corresponding arbitration logic to access the memory unit associated with the class. This dynamic priority hierarchy enhances the flexibility of the system and in fact is quite reasonable if one considers the fact that the processors were separated into classes only with respect to the memory unit which they may access but not with respect to their function, i.e., their "priority", within the system.

Suppose for a moment that we want to model the producer-consumer synchronization problem presented at the beginning of this section using one of the other models examined in Chapter IV. Due to the priority

hierarchies involved, the PPM of [HACK-75] and the PPM(I) and the C-CPM(I), which we have introduced, appear to be the logical first choices. Nevertheless, in the remainder of this section we shall informally argue that these models cannot provide a natural representation of this synchronization problem.

Figure 5.2.4 displays a possible PPM representation of the synchronization system of Figure 5.2.1. Transitions t_1, t_2, t_3 and t_4 model the producer processes P_{11}, P_{12}, P_{21} and P_{22}, respectively. We note that in the problem statement, the producer processes P_{i1} and P_{i2} share the buffer B_i and that the consumer process C_i must recognize which items present in B_i were produced by P_{i1} and which by P_{i2}, $i = 1, 2$. Therefore, items in the shared buffer must uniquely identify their parent producer process. This modelling characteristic can easily be taken care of in the C-CPM by placing distinctly colored tokens in the place which models the shared buffer. In the PPM, however, since we lack colored tokens, it appears that we have to use two separate places in order to model a buffer which contains two categories of items (places p_{22} and p_{23} model the buffer B_1 while p_{24} and p_{25} model B_2). This drawback is amplified in case more than two producer processes deposit items in the same buffer. This inadequacy carries over to the PPM(I) but not to the C-CPM(I).

Next, let us examine the modelling of the consumer processes. Due to the fixed and dynamic priority hierarchies incorporated in our producer-consumer system, on one hand, and to the fact that the transitions of a PPM are assigned fixed priorities, on the other hand, it appears that we cannot model each consumer process by a single transition, as we have done in Figure 5.2.2. Figure 5.2.3 displays all possible contents of the buffers B_1 and B_2. For each situation, the table indicates which consumer process can proceed and what item it has to consume as well as the transition which models the respective operation (note the relationship between the leftmost two columns of the Table and the input/output connections of the transitions t_5, t_6, ... , t_{21} of Figure 5.2.4). The Table also indicates the priority assignment to the transitions which ensures that the PPM of Figure 5.2.4 functions correctly.

Thus, even if the PPM can correctly model the producer-consumer

p_{22}	p_{23}	p_{24}	p_{25}	Consumer	Transition	Priority (Y)
0	0	0	0	-	-	-
0	0	0	1	C_{22}	t_5	q_1
0	0	1	0	C_{21}	t_6	
0	1	0	0	C_{12}	t_7	
1	0	0	0	C_{11}	t_8	
0	0	1	1	C_{22}	t_9	q_2
0	1	0	1	C_{12}	t_{10}	
0	1	1	0	C_{12} or C_{21}	t_{11} or t_{12}	
1	0	0	1	C_{11}	t_{13}	
1	0	1	0	C_{11}	t_{14}	
1	1	0	0	C_{12}	t_{15}	
0	1	1	1	C_{12}	t_{16}	q_3
1	0	1	1	C_{11}	t_{17}	
1	1	0	1	C_{12}	t_{18}	
1	1	1	0	C_{12} or C_{21}	t_{19} or t_{20}	
1	1	1	1	C_{12}	t_{21}	q_4

p_{2r} { empty (zero marking) $\longleftrightarrow$ 0
full (positive marking) $\longleftrightarrow$ 1

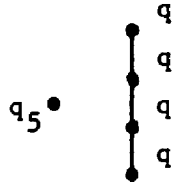
$r = 2,3,4,5$

$C_{ij} \longleftrightarrow$ Consumer C_i consumes an item produced by P_{ij}

$i = 1,2$

$j = 1,2$

$Pr = (\{q_1,q_2,q_3,q_4,q_5\}, \underline{\alpha})$



$Y(t_1) = Y(t_2) = Y(t_3) = Y(t_4) = q_5$

Figure 5.2.3

Buffer contents for the Producer-Consumer Synchronization System of Figure 5.2.1

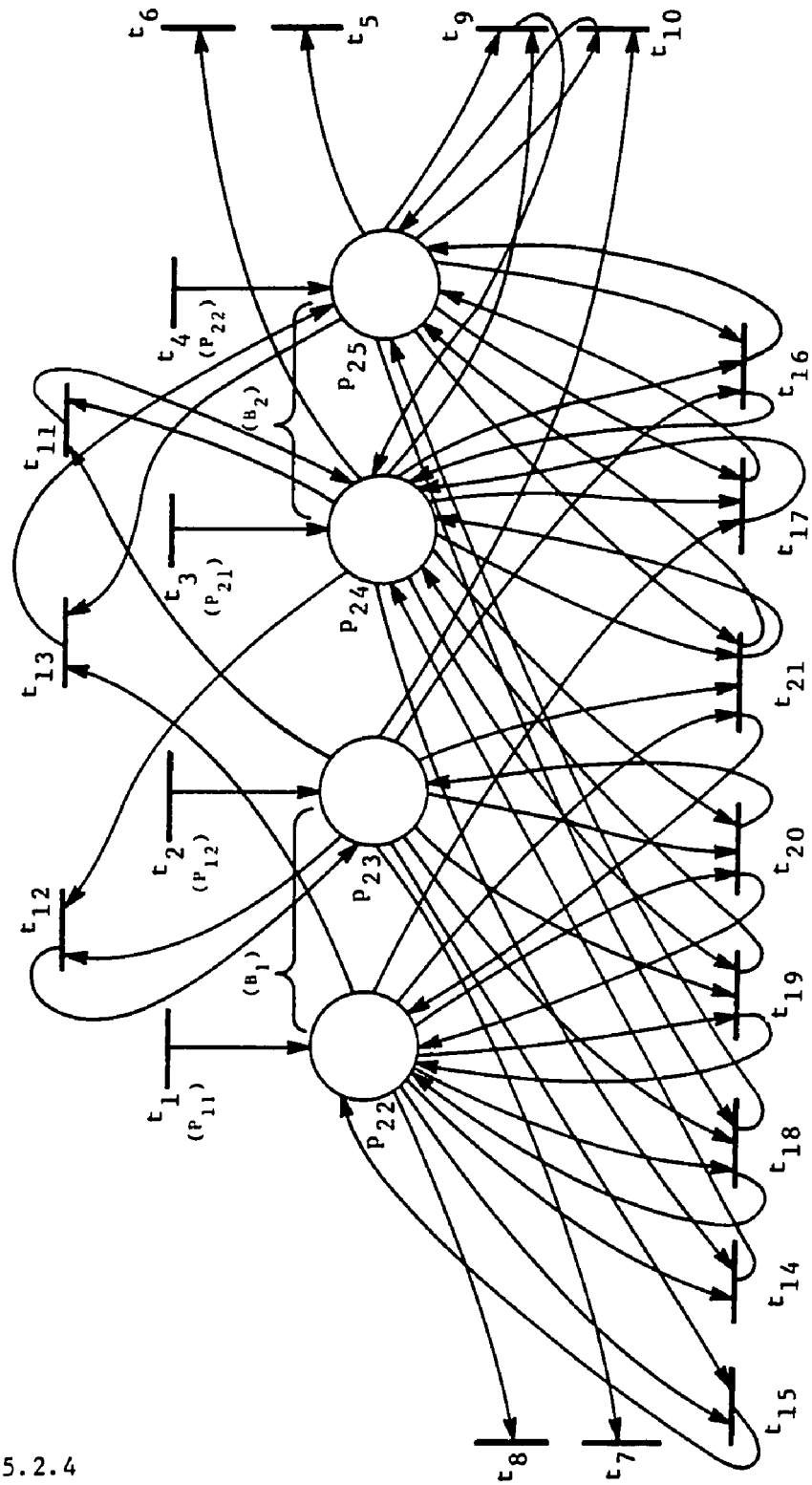


Figure 5.2.4

PPM of the Producer-Consumer Synchronization System of Figure 5.2.1

problem under consideration, it fails to offer a natural and easy to understand representation. Without the aid of the table of Figure 5.2.3 it is rather difficult to establish a correspondence between the transitions and places of the PPM of Figure 5.2.4 and the entities of the modelled system (Figure 5.2.1). From this point of view, the PPM(I) and the C-CPM(I) present the same inadequacies since in both cases transitions have fixed, predetermined priorities.

Section 5.3 A PROCESS COORDINATION PROBLEM WITH CONFLICT

Let us examine the following process coordination problem, depicted schematically in Figure 5.3.1. Suppose the system contains four buffers, denoted by B_1, B_2, B_3 and B. Three processes, denoted by P_{i1}, P_{i2} and Proc(i), are connected to each buffer B_i, i = 1, 2, 3. The buffer B can be accessed only by the three processes Proc(i) and is assumed to contain initially two undistinguishable items. Any of the processes P_{ij}, j = 1, 2, may be activated at any time and it produces and deposits an item in the associated buffer B_i. The process Proc(i) becomes active only if the associated buffer B_i and the shared buffer B are nonempty. If active, Proc(i) attempts to consume an item from both B_i and B with the restriction that it may consume from B_i an item produced by P_{i1} only if B_i does not contain any items produced by P_{i2}. Moreover, the following priority rule is imposed:

i. If the number of items currently contained in the buffer B is (strictly) less than the number of active processes Proc(i) then we shall say that a conflict situation has arised. In this case Proc(1) has highest priority to proceed, Proc(3) has lowest priority and Proc(2) has lower priority than Proc(1) but higher than Proc(3).

ii. Otherwise, either active process Proc(i) may proceed.

We note that in both situations only one process Proc(i) may proceed at a time. After the chosen process, say Proc(k), has consumed the respective items from B_k and B it will temporarily deactivate itself. For example, we can assume that Proc(k) initiates at this stage an abstract operation associated with it which requires an arbitrarily long (possibly null) period of time to execute. Only after the abstract operation is terminated will Proc(k) restore the item consumed from B and regain its active status, as soon as both B_k and B are nonempty.

Meanwhile, however, the pending active processes Proc(i), $i \neq k$,

are polled again in order to determine which process proceeds next. From this point of view the processes Proc(i) operate asynchronously and concurrently.

The characteristic aspect of this process coordination problem is the conflict situation which arises in case the number of available resources in a shared pool of homologous resources is less than the number of processes competing for the respective resources. The priority rule designed for solving the conflict situation can be represented in the C-CPM in a natural way mainly due to the conflict structure (conflict cluster, conflict subclusters, etc.) employed for the selection of a transition to be fired. Figure 5.3.2 exhibits the C-CPM representation of the coordination system of Figure 5.3.1. We have indicated adjacent to the places and transitions of Figure 5.3.2 which entities of the coordination system they model. Let us examine closer the modelling of the processes Proc(i). For example, transition t_7 may be considered to model the initiation of the abstract operation associated with Proc(1) while t_8 may be assumed to model the termination of that operation. The presence of a token in place P_{17} signifies that the abstract operation is in execution. Conversely, the occurrence of a token in place P_{18} signals that the respective operation is terminated. An analogous approach has been taken to the modelling of the processes Proc(2) and Proc(3).

This type of coordination problems can certainly be further extended. In particular, we note that we have used a totally ordered priority hierarchy for solving the conflict situation at the buffer B only in order to have a simple problem statement. Actually, more sophisticated priority hierarchies can be used. For example, we can make the priority of the processes Proc(i) depend on the particular selection of items from the associated buffers B_i. In this case the conflict situation could be solved according to a dynamic priority hierarchy similar to that described in Section 5.2.

This kind of process coordination problem can again be recast to the computing system structure described in Section 5.2. In this case, however, we shall assume that the bus is formed of n bidirectional, parallel lines, $n < k$. Each line may connect any processor-memory unit pair (as long as the connection is legal from the processor

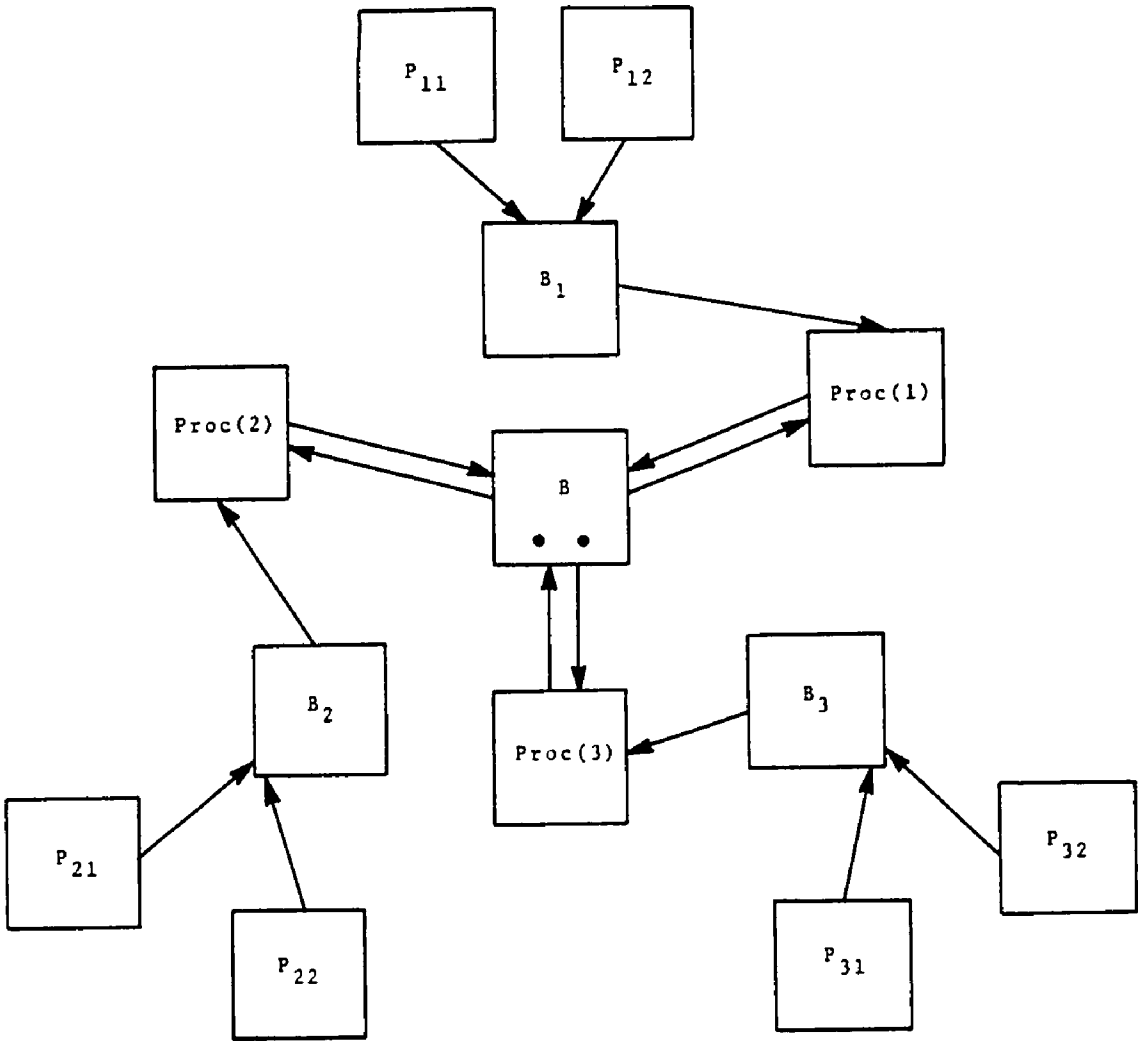


Figure 5.3.1

Process Coordination System with Conflict

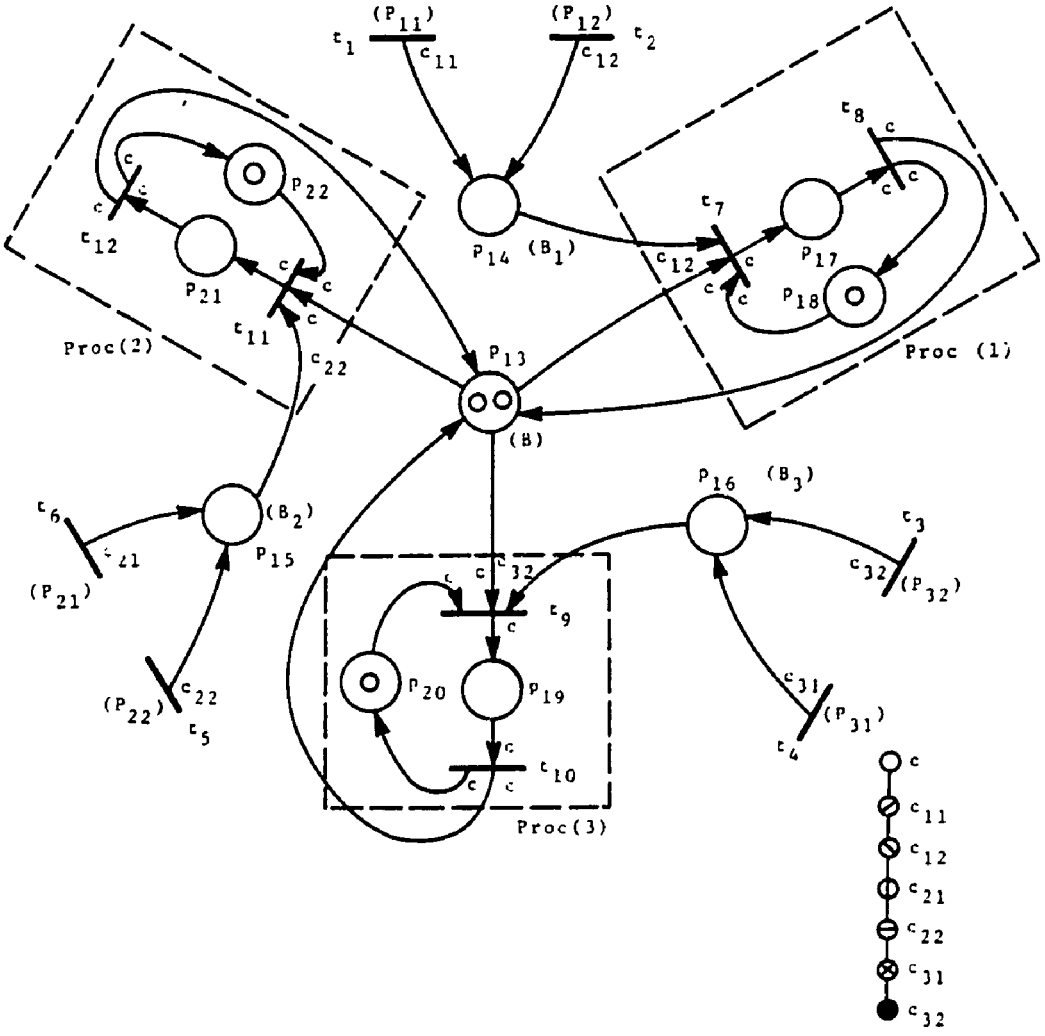


Figure 5.3.2

C-CPM Representation of the Coordination System of Figure 5.3.1

class point of view). After the "local" arbitration logics corresponding to each class have designated a processor from the respective class to be connected to the associated memory unit, the "global" arbitration logic assigns one of the free lines to one of the designated processors. If the number of available lines is greater than or equal to the number of processors requesting to be connected, either processor may seize a communication line. Otherwise, the "global" arbitration logic enforces the priority hierarchy designed for solving the conflict situation.

Once a processor has seized a line, it will not release it until the data transfer to the corresponding memory unit has been performed. After the data transfer is terminated, the line will be returned to the pool of available communication lines. Meanwhile, since the memory unit is busy, no other processor from the respective class may ask for another line. Nevertheless, processors from other classes (for which the corresponding memory unit is not occupied by a data transfer) can seize the remaining available lines (if any). Therefore, after a line has been assigned, the remaining pending processors and the pool of communication lines are again "polled" by the "global" arbitration logic in order to assign another line.

Suppose now that we want to use the PPM for the modelling of the synchronization system of Figure 5.3.1. We note that the buffers B_i, $i = 1, 2, 3$, may contain two distinct categories of items at a time. As in Section 5.2, due to the fact that the PPM does not incorporate colored tokens, the easiest solution to this modelling aspect appears to be the use of two distinct places for the representation of each buffer B_i. Let us simplify for a moment the original synchronization problem and suppose that only one process P_i deposits items in the buffer B_i, i.e. the items present in B_i are undistinguishable. Figure 5.3.4 exhibits a PPM representation of this restricted version of the synchronization system under consideration. In the figure, we have indicated adjacent to the places and transitions of the net which entities of the modelled system they represent. Let us have a closer look at the modelling of the processes Proc(i). We note that in the synchronization system of Figure 5.3.1, even though we have used a simple (totally ordered) and fixed priority hierarchy for solving the conflict situation at the shared buffer B, the processes Proc(i) are

	Active Processes Proc(1)	Proc(2)	Proc(3)	Proc(i)	Transition
	0	0	0	-	-
	0	0	1	Proc(3)	t_4
	0	1	0	Proc(2)	t_5
B contains	0	1	1	Proc(2)	t_5
1 item	1	0	0	Proc(1)	t_6
	1	0	1	Proc(1)	t_6
	1	1	0	Proc(1)	t_6
	1	1	1	Proc(1)	t_6
	0	0	0	-	-
	0	0	1	Proc(3)	t_7
	0	1	0	Proc(2)	t_8
B contains	0	1	1	Proc(2)/Proc(3)	t_7/t_8
2 items	1	0	0	Proc(1)	t_9
	1	0	1	Proc(1)/Proc(3)	t_9/t_7
	1	1	0	Proc(1)/Proc(2)	t_9/t_8
	1	1	1	Proc(1)	t_{10}

t_k	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
$Y(t_k)$	q_8	q_8	q_8	q_1	q_2	q_3	q_4	q_5	q_6	q_7

$$\text{Proc(i)} = \begin{cases} 1 \longleftrightarrow \text{active} \\ 0 \longleftrightarrow \text{disabled} \end{cases}$$

i = 1,2,3.

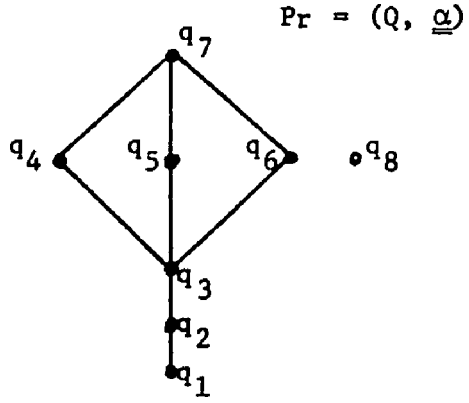


Figure 5.3.3

Coordination Policy for the Process

Coordination System of Figure 5.3.1

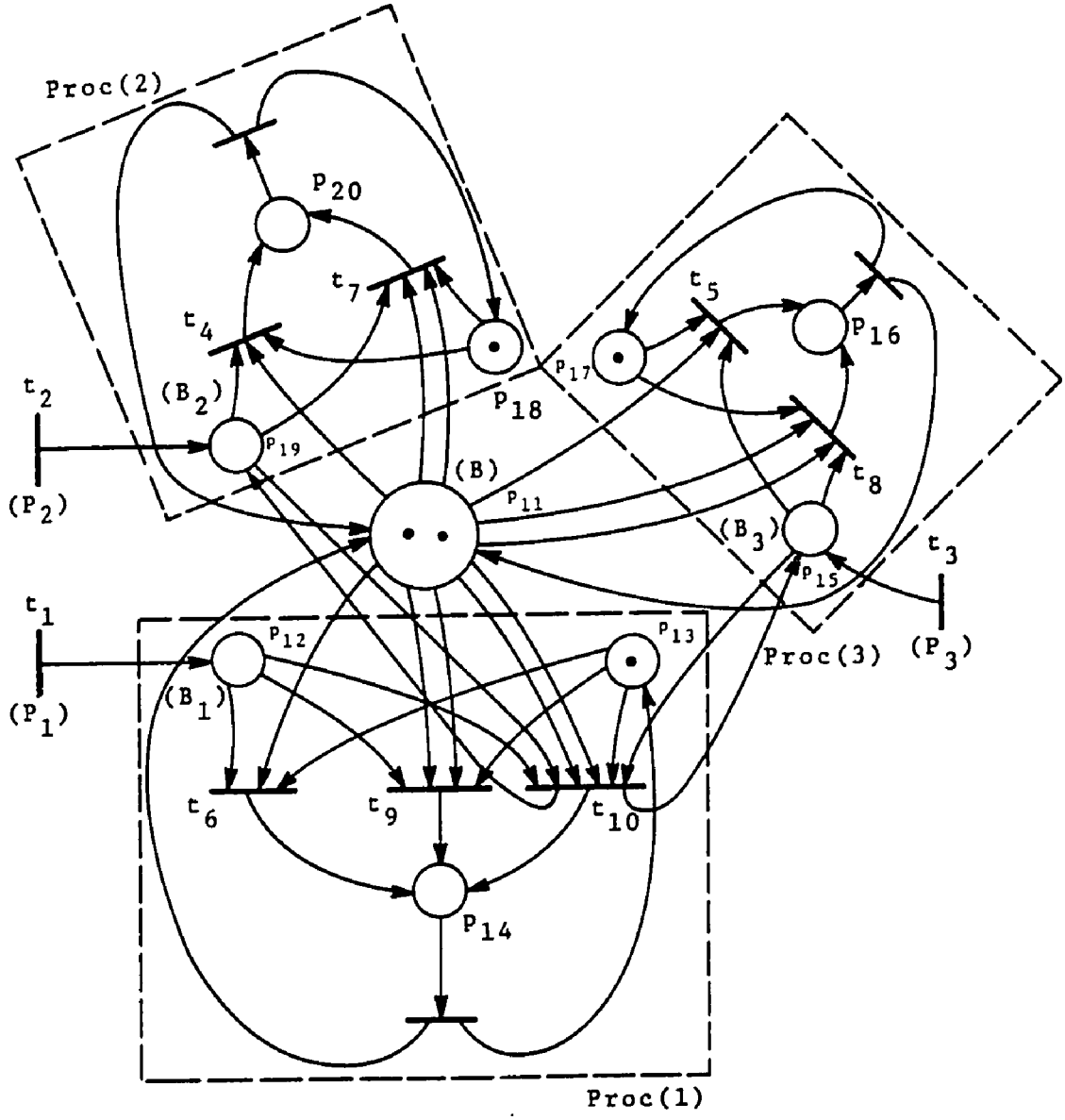


Figure 5.3.4

PPM of the Process Coordination System of Figure 5.3.1 (only one process P_i is connected to each buffer B_i, $i = 1,2,3$)

not assigned a fixed priority. In fact the priority to proceed of a particular process Proc(i) depends both on the current content of the buffer B and on the status (active versus disabled) of the other processes Proc(j), $1 \leq j \leq 3$ and $j \neq i$. On the other hand, the transitions of a PPM carry a fixed priority, in particular, independent of the markings of the respective net. Therefore, it appears that we cannot model the initiation of the abstract operation associated with a process Proc(i) by a single transition, as we have done in the C-CPM of Figure 5.3.2. The Table of Figure 5.3.3 considers all possible situations of the particular processes Proc(i). In each case the Table indicates (second column) which process Proc(i) can proceed and which transition models the initiation of the particular abstract operation associated with that process (for example, transitions t_6, t_9 and t_{10} model each the initiation of the abstract operation associated with Proc(1) in distinct situations). The size of the PPM of Figure 5.3.4 would be approximately doubled if we would have modelled the original system where the buffers B_i may contain two distinct categories of items. The size of the PPM of Figure 5.3.4 would also increase considerably if the modelled synchronization system contained more than three processes Proc(i) and more than two items in the buffer B.

We can conclude that the PPM fails to offer natural representation for conflict situations such as that examined in this section. The PPM(I) and the C-CPM(I) which we have defined in Chapter IV appear to be more adequate for this purpose since they incorporate the concepts of conflict cluster, conflict subcluster, etc. in a manner similar to the C-CPM. Nevertheless, as we have mentioned in Section 5.2, the PPM(I) and the C-CPM(I) would fail to offer a natural representation in case the priority hierarchy designed for solving conflicts would vary dynamically (i.e., would depend on the particular items selected from B_i by the processes Proc(i)).

Section 5.4 MODELLING REENTRANCY

In this section we shall examine the modelling of a special kind of concurrent control mechanism in software systems, namely the reentrant subroutines. As pointed out in [GRIE-71], this programming feature is "most useful in a multi-programming or time-sharing environment, where several programs may call, say, a SIN routine which is reentrant. Only

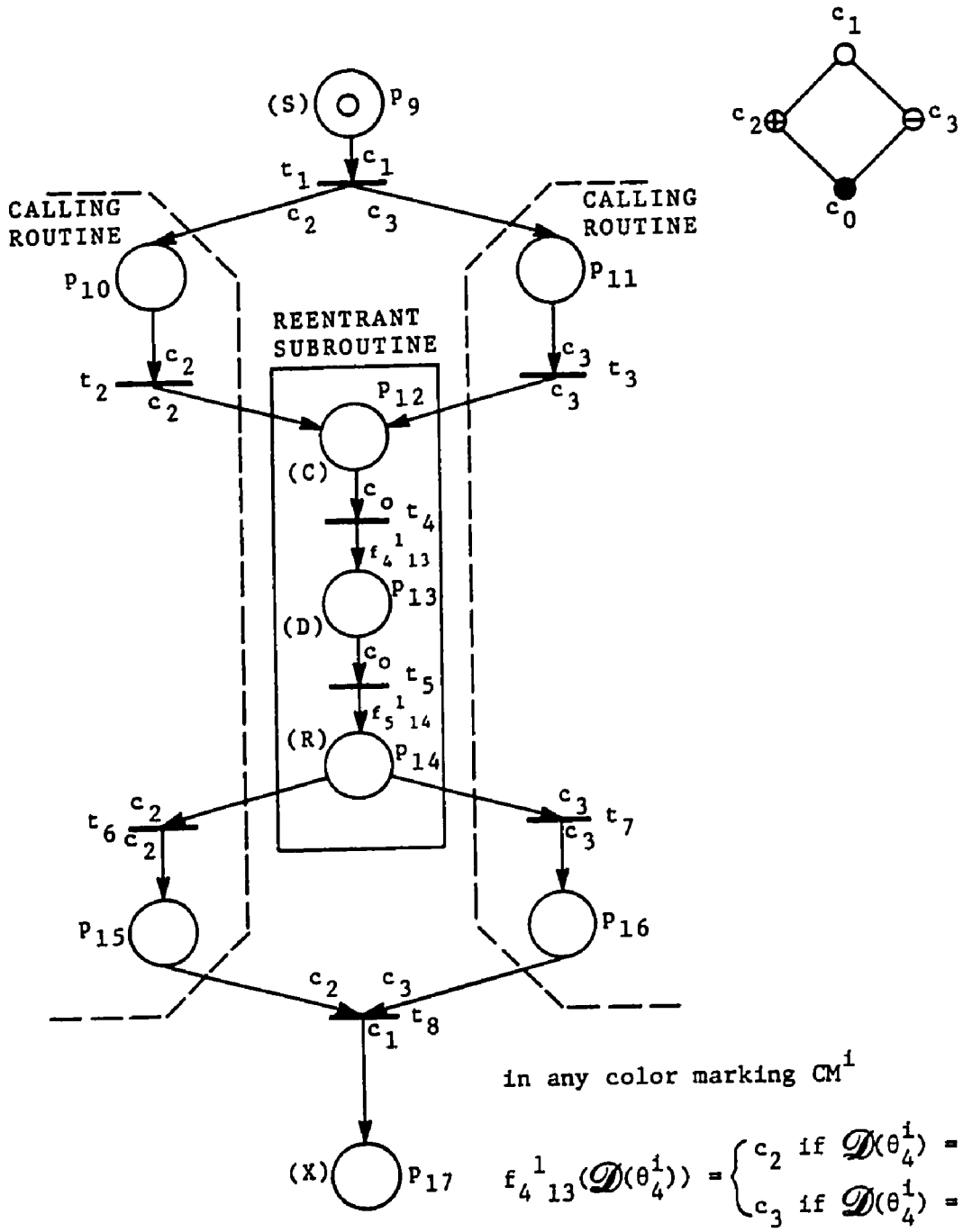
one copy of the routine need be kept in memory, no matter how many people are executing it."

The efficient and clear modelling of reentrancy becomes especially important when one is interested in modelling real-life collections of programs such as compilers or reentrant portions of operating systems.

Let us reexamine the example presented in Section 1.1. Figure 5.4.1 exhibits our approach to the modelling of the flow of control in the reentrant subroutine of Figure 1.1.1. Our only requirement is that distinct control flow streams operating in a reentrant subroutine be represented by distinct colors. We note that upon firing, transition t_5 places in p_{14} a token of color c_2 or c_3, depending on its enabling color. On the other hand, due to the particular structure of the lattice of colors, the transitions t_6 and t_7 can select as enabling colors from p_{14} only the colors c_2 and c_3, respectively. Thus, control is always returned from the reentrant subroutine to the proper calling routine.

We note that the purpose of the color output functions $f_4{}^1{}_{13}$ and $f_5{}^1{}_{14}$ is to "paint" the output tokens produced by the transitions t_4 and t_5, respectively with the same color as that of the tokens absorbed by the respective transitions from their input places. In this way, the continuity of the control flow streams operating concurrently in the reentrant subroutine is ensured. In view of the equivalence results presented in Section 4.5 the usage of this type of mappings as color output functions does not alter the representation power of the C-CPM but, we believe, enhances the naturalness of the representation of reentrancy.

Our method applies equally well to nested reentrant subroutines. Consider for example the synchronization situation depicted schematically in Figure 5.4.2. There, both subroutines A and B may be invoked concurrently by different calling programs. Figure 5.4.3 displays the C-CPM representation of the flow of control in this synchronization system. Note that due to the structure of the lattice of colors, the return of control from subroutine B is correctly modelled, that is the control flow stream originating from PROGRAM 3 can never be routed to subroutine A upon termination of the execution of subroutine B and vice-versa.



in any color marking CM^i

$$f_4{}^1{}_{13}(\mathscr{D}(\theta_4^i)) = \begin{cases} c_2 & \text{if } \mathscr{D}(\theta_4^i) = \{c_2\} \\ c_3 & \text{if } \mathscr{D}(\theta_4^i) = \{c_3\} \end{cases}$$

$$f_5{}^1{}_{14}(\mathscr{D}(\theta_5^i)) = \begin{cases} c_2 & \text{if } \mathscr{D}(\theta_5^i) = \{c_2\} \\ c_3 & \text{if } \mathscr{D}(\theta_5^i) = \{c_3\} \end{cases}$$

Figure 5.4.1

Modelling of a Sample Reentrant Subroutine

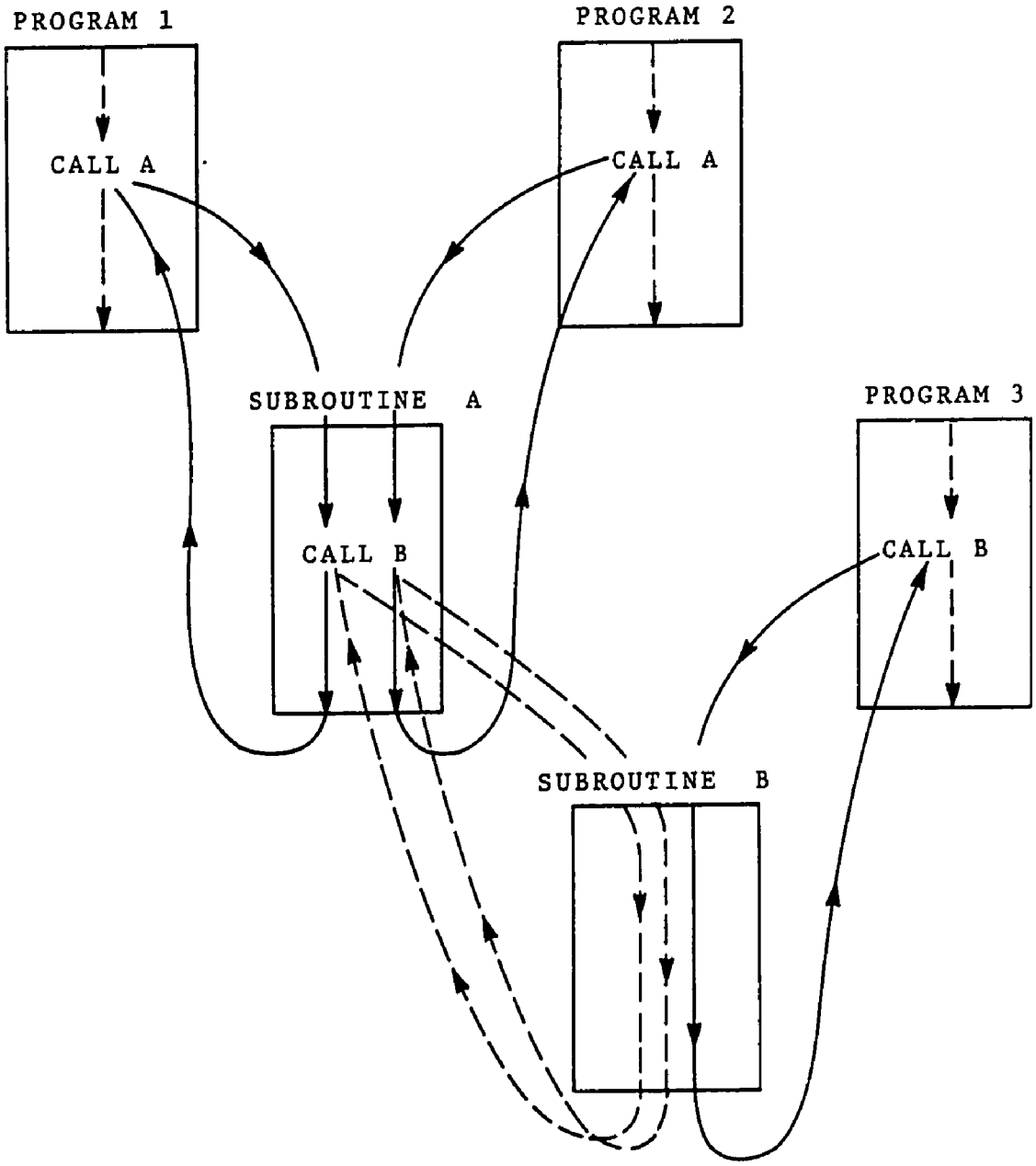


Figure 5.4.2

Nested Reentrant Subroutines

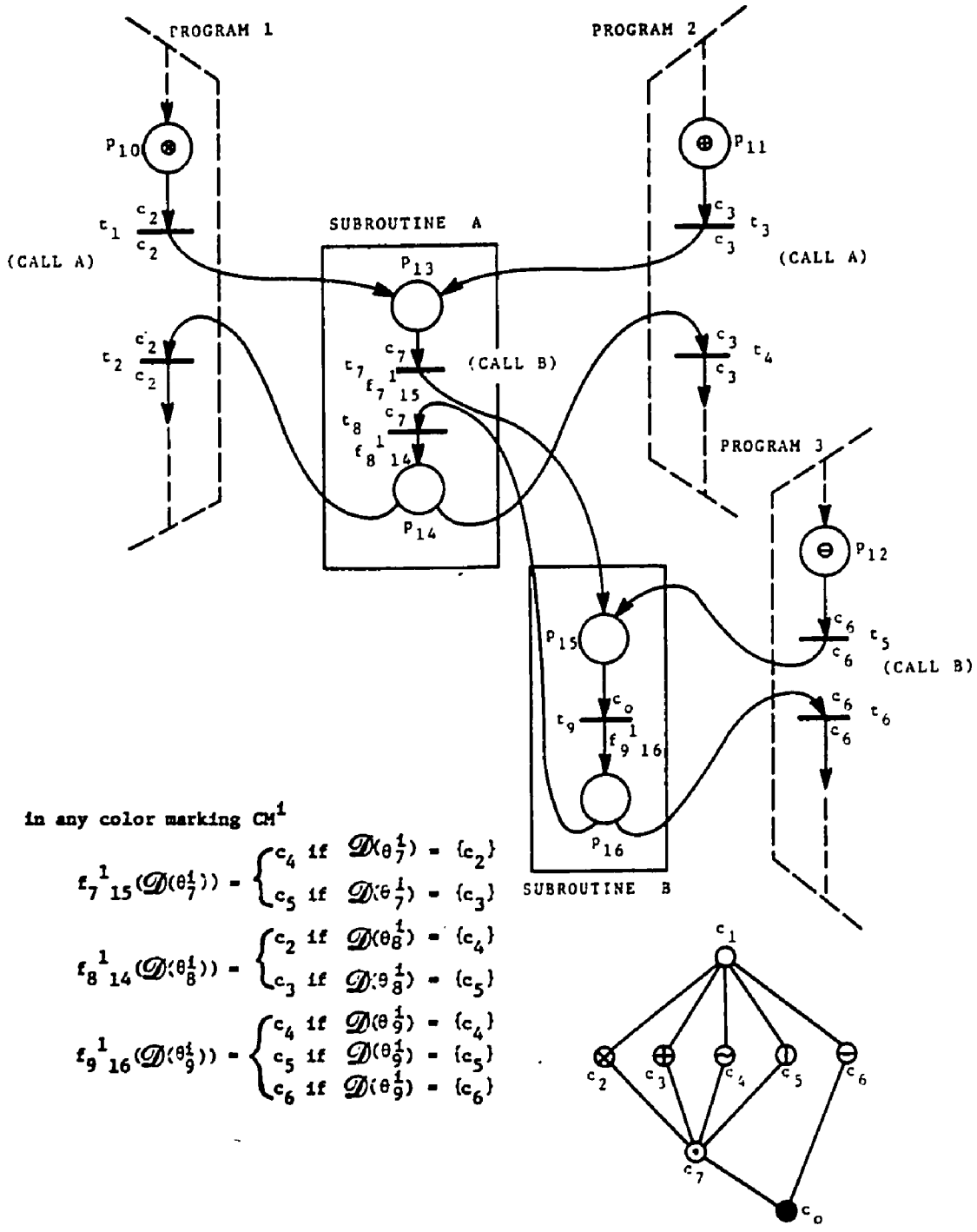


Figure 5.4.3

Modelling of Nested Reentrant Subroutines

In the examples presented above we have attempted to show that the use of colors enhances a natural representation of the distinct control flow streams which operate concurrently in reentrant programs. Without going into details, we argue informally that the EPM, PPM or any of the other models examined in Chapter IV which do not incorporate colored tokens fail to offer a natural representation of reentrancy. Thus, due to the lack of colors, it appears that any of these models will have to approach the modelling of reentrancy in a manner similar to the "copy" method of [GOST-71] and [CERF-72] which we have described in Section 1.1.

Section 5.5 MODELLING WITH THE EC-CPM

We have shown (Theorem 4.8.1) that the language families Λ(C-CPM) and Λ_o(C-CPM) are properly included in Λ(EC-CPM) and Λ_o(EC-CPM), respectively. On the other hand, we have also shown that any recursively enumerable language can be generated as the image under (possibly unrestricted) erasing homomorphism of a language in Λ_o(C-CPM). Thus, according to Theorem 4.8.2, given any Labelled EC-CPM ECP_Σ one can construct a Labelled C-CPM CP_Σ which will simulate the firing sequences γ of ECP_Σ using at most $d^{|\gamma|^2}$ λ-transition firings in order to do so, for some constant $d > 1$. We have also mentioned that there are languages in Λ_o(EC-CPM) and Λ(EC-CPM) for which an exponential bound on λ-transition firings is necessary. Thus any Labelled EC-CPM can be simulated by a Labelled C-CPM but not without considerable difficulty. As pointed out in [MILL-74]: "A problem with such a construction, however, is that one no longer has a one-one relationship, between "processes" in the two models. Here, additional "dummy" transitions and places are required, and in comparing the behaviors in the two models a distinction has to be made between dummy transitions or places and real transitions or places. This distinction is somewhat unnatural both from the viewpoint that the formal Petri Net model makes no such distinctions, as well as that the naturalness of the process structure is either lost or hidden." Indeed, we recall that each labelled transition was assumed to correspond to the activation of a particular process of the system modelled by the respective net. In this perspective, λ-transitions do not correspond to any process of the modelled system, they merely serve as internal operations of the net.

Thus, even though the introduction of λ-transitions extends the set of languages which can be generated by the C-CPM, it impairs the clarity and naturalness of the representation offered by the C-CPM. It is therefore only natural to conclude that a formal model which can represent in a λ-transition free manner a class of systems as large as possible, is desirable. This viewpoint adds relevance to the results which we have mentioned earlier regarding the relationship between the C-CPM and the EC-CPM. At the same time, the equivalence results of Chapter IV also show that the language families Λ and Λ_o generated by any of the "complete" formal models examined in [HACK-75] and [AGAR-75], i.e., the PPM, EPM, Coordination Petri Model and the Petri Nets with switches, disjunctive logic and token absorbers, are properly included in the respective language families generated by the EC-CPM.

In the remainder of this chapter we illustrate the validity of our viewpoint regarding the natural representation of coordination systems. We shall present synchronization systems which can be modelled in a natural way by the EC-CPM and which do require the additional representation power of the EC-CPM.

Section 5.6 A PRODUCER-CONSUMER SYNCHRONIZATION PROBLEM WITH LIFO STRUCTURED BUFFER, REVISITED

Let us reexamine the producer-consumer synchronization problem described at the beginning of Section 2.5 (represented schematically in Figure 2.5.1). Let us assume for a moment that the following additional restriction is imposed on the system. The producer processes are disabled after either consumer process, C_1 or C_2, has consumed an item from the shared buffer until the buffer becomes empty. With this restriction, the synchronization system is forced to (repeatedly) execute the following basic "cycle". First, the producer processes P_1 and P_2 produce and deposit in the buffer B an arbitrary number of items, until an item is consumed by one of the consumer processes. After that, the producer processes are disabled and must wait until the consumer processes have emptied the buffer. All this time, the buffer B is accessed in a LIFO mode. The producer processes may eventually restart to produce and deposit items as soon as the buffer becomes empty and therefore, a coordination sequence of the system may be viewed as an arbitrarily long sequence (repetition) of such "cycles".

Let us now label the processes P_1 and C_1 with the label a and the processes P_2 and C_2 with the label b. It is easy to see that if we record the labels of the processes occurring along one of the execution "cycles" mentioned above, the resulting string is of the form ww^R, where $w \in \{a, b\}^+$. Consequently, the set of strings generated along all possible coordination sequences of the system is the language $W' = (W_o)^+$, where $W_o = \{ww^R \mid w \in \{a, b\}^+\}$.

Let us now remove the restriction mentioned earlier, i.e., the producer processes may produce and deposit items at any time, independent of the content of the buffer B. In this case, the coordination sequences performed by the system follow the same basic pattern as in the restricted case except that execution "cycles" may be nested. Thus, the language W generated becomes:

$$W = (S(W_o))^+$$

where S is the substitution mapping defined by:

$$S(a) = \{a\}(W_o)^*$$
$$S(b) = \{b\}(W_o)^*$$

The language W has the simple context-free grammar:

$$S \to aAS \qquad S \to bBS$$
$$S \to aA \qquad S \to bB$$

$$A \to aAA \qquad A \to bBA \qquad A \to a$$
$$B \to bBB \qquad B \to aAB \qquad B \to b$$

where S denotes the start symbol.

We already know that the language W_o is not contained in Λ(C-CPM) or Λ_o(C-CPM). The techniques mentioned in Section 4.8 can be applied in order to show that W is not in Λ(C-CPM) or Λ_o(C-CPM), either. Moreover, one can show that if W is expressed as the image under an erasing homomorphism h of some language $W'' \in \Lambda_o$(C-CPM) then h must be exponentially-bounded on W''. The reason is again the fact that the number of distinct intermediate markings necessary to generate strings w from W grows as an exponential function of $|w|$ while the number of distinct markings reachable by a λ-transition free Labelled C-CPM grows as a polynomial function of the length of the respective firing sequences.

On the other hand, the EC-CPM can model this type of synchronization systems in a natural way, which preserves the process structure. This is mainly due to the fact that the EC-CPM has the capacity to "remember" the order of arrival of colored tokens in the places of a net. Such information is essential for the correct modelling of the consumer processes, i.e., for the correct modelling of the policy for the retrieval of items from the buffer.

The type of synchronization problems under consideration in this section has practical significance and actually can be often encountered in computing environments. For example, they can easily be recast to operating system scheduling problems. Thus, imagine a multiprogrammed computing system in which several user-job processes operate concurrently. Each job process has input and/or output files stored on external memory units (disks, tapes, etc.). At any time, a job process can produce an output message which is stored in a private memory area owned by it. As soon as such an output message has been assembled, the respective job process will lodge an output request with the supervisor process SV. Similarly, a job can request at any time for an input message to be transmitted from one of its input files into a memory region owned by that job process. Input requests are lodged with the supervisor process as well. The requests for I/O operations are stored in the order of their arrival in a buffer shared by all user-job processes present in the system.

Concurrently, the supervisor process SV examines the shared buffer and routes the I/O requests to the appropriate I/O channels, according to a LIFO or FIFO retrieval policy. It is assumed that several I/O units are controlled by one I/O channel. Therefore, it is possible for several job processes to have their I/O files under the control of the same I/O channel. Since the affiliation of I/O units to the particular I/O channels is transparent to the user-job processes, the system requires the intervention of the SV process in the distribution of the job process I/O requests to the corresponding I/O channels. Also, due to the fact that the job processes and the I/O channel processes operate asynchronously and concurrently at independent speeds, each I/O channel process may be provided with its own LIFO or FIFO structured buffer in which the supervisor process can store the I/O requests

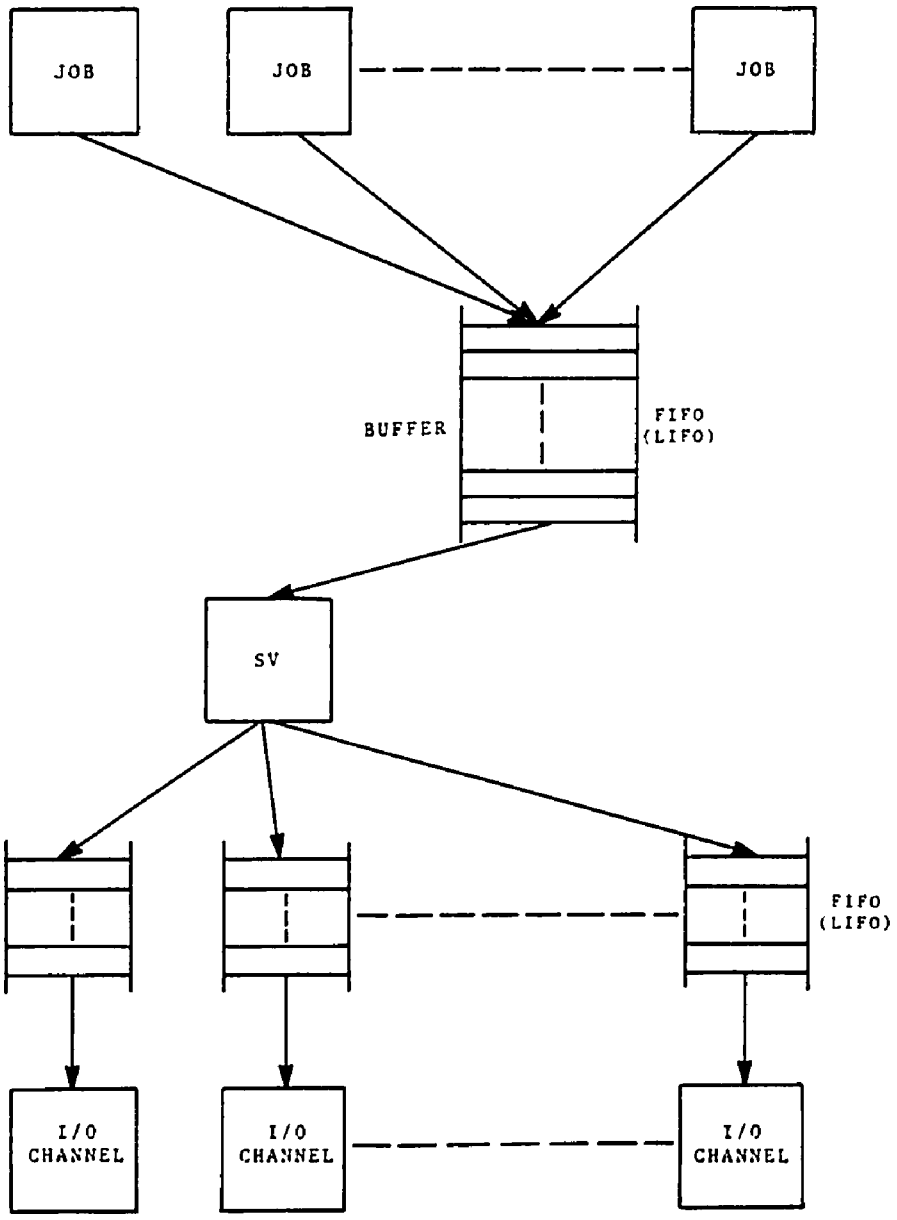


Figure 5.6.1

Transfer of Requests for I/O Operations in a Multiprogrammed Computing System

pertinent to that particular channel. Figure 5.6.1 depicts schematically the transfer of requests for I/O operations from the job processes to the I/O channel processes via the SV process.

Section 5.7 MODELLING OF THE FLOW OF CONTROL IN RECURSIVE SUBROUTINES

In this section we present a different approach to the representation of the flow of control in computer programs. The method originates with [URSC-72] and has been applied to the exploitation of maximum parallelism in flow diagrams ([URSC-72] and [URSC-73]) and to the automatic structuring of programs (URSC-75]). We slightly modify and extend this representation method in order to employ it in the modelling of the flow of control in recursive programs by the EC-CPM.

Programs are assumed to be represented by uninterpreted flow diagrams, i.e., by finite, directed, labelled graphs which satisfy the following conditions:

1. A flow diagram contains exactly one begin node (denoted by s) and one exit node (denoted by x).
2. Each node in the flow diagram is reachable from s and x can be reached from it.
3. Edges with the same source have different targets.

Figure 5.7.1 exhibits a sample flow diagram.

Ramification nodes (also called decision nodes) are the sources of at least two different edges, correspond to the decision statements of the underlying program and are represented as diamond shaped boxes. Nodes which are the source of a single edge only are called elementary blocs and correspond to "basic" (data oriented) statements of the underlying program. Elementary blocs as well as the begin and exit nodes are represented as rectangular boxes. The nodes of a flow diagram are labelled by distinct lower-case letters.

Definition 5.7.1

A node n_2 is a successor of a node n_1 if there is an edge in the graph with source n_1 and target n_2. A path is a sequence of nodes each of which (except for the first node) is a successor of the node immediately preceding it in the path.

A path is called cycle-free if it contains no node twice.

Definition 5.7.2

A node n_2 is a post dominator of a node n_1 if each path from n_1 to x contains n_2; n_2 is said to be the immediate post dominator of n_1 if in each path from n_1 to x, n_2 is the post dominator of n_1 occurring first.

The immediate post dominator of an elementary bloc is its successor. The immediate post dominator of a decision node d is the junction point of all the cycle-free paths from d to x. It has been shown ([URSC-72]) that each node has a uniquely determined immediate post dominator, except for x.

Definition 5.7.3

The scope of a decision node d is the set of all nodes which:

i. are successors of d or are post dominators of successors of d
ii. are not post dominators of d itself.

Definition 5.7.4

The control history of a node n is a list consisting of n as top element and all post dominators of n in the order in which they occur in any cycle-free path from n to x. The control history of n restricted to the scope of a decision node d is the list obtained by deleting from the control history of n the elements not belonging to the scope of d.

Using these concepts, the flow of control in a flow diagram can be expressed in terms of the productions of a context-free grammar. For this purpose, the capital labels of the nodes contained in the flow diagram are considered to be the nonterminal symbols of the grammar while the lower-case labels of the nodes define the terminal alphabet. S serves as the start symbol of the grammar. The set P of productions is formed as follows:

i. $X \rightarrow x$ is in P.
ii. If n is an elementary bloc then $N \rightarrow n$ is in P.
iii. $S \rightarrow s\delta$ is in P where δ is the tail of the control history of s (i.e., the control history of s with the top element, s, deleted) written in capital symbols.
iv. If d is a decision node then P contains a production for

each successor of d. Each such production is of the form $D \rightarrow d\gamma$ where γ is the control history of the respective successor of d restricted to the scope of d and written in capital symbols (if not empty).

The productions corresponding to the flow diagram of Figure 5.7.1 for example, are:

$S \rightarrow sABDEFX$	$A \rightarrow a$	$F \rightarrow f$	$E \rightarrow e$
$B \rightarrow bC$	$C \rightarrow c$	$X \rightarrow x$	
$B \rightarrow b$	$D \rightarrow d$	$E \rightarrow eCDE$	

As shown in [URSC-72] the construction of the production system is an effective and unambiguous procedure. The main difference between the set of productions obtained by our construction algorithm and that described in [URSC-72] is that here the grammar is obtained directly in Greibach normal form.

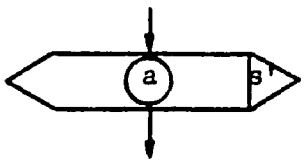
In what follows, given an arbitrary flow diagram F, we shall refer to the grammar obtained by applying the above algorithm to F as to the "context-free grammar corresponding to F."

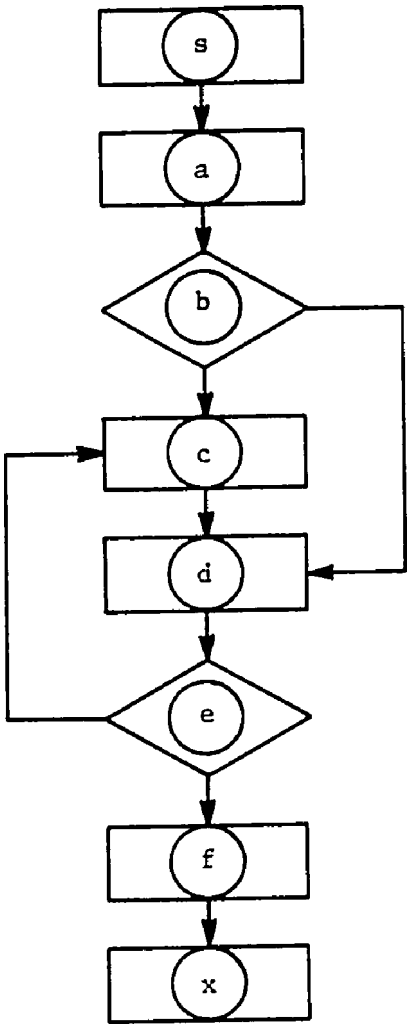
Definition 5.7.5

Given a flow diagram F, a computation in F is a path from s to x in F.

Let F be an arbitrary flow diagram and let G be the context-free grammar corresponding to F. Using arguments similar to those of Sentences 2.4 and 2.5 of [URSC-72] one can show (proof omitted) that $w \in W(G)$ if and only if w is the label sequence of a computation in F.

The method described above does not handle specifically the representation of subroutine calls. Nevertheless, it can be extended in order to accomodate this form of flow of control in computer programs, as well. Suppose we are given a calling program and the corresponding subroutine represented as the flow diagrams F and F', respectively (see Figure 5.7.2, for example). An instance of a call to subroutine statement is represented in F as a bloc of the form:





Node	s	a	b	c	d	e	f
immediate post dominator	a	b	d	d	e	f	x

Figure 5.7.1

Sample Flow Diagram

Each such bloc is called a "call" bloc and is associated with a pair of labels (a, s'), where a denotes the label of the call bloc while s' denotes the label of the begin node of F'. Note that a call bloc is the source of only one edge. Therefore, as in the case of the elementary blocs, a call bloc has only one successor which is also its immediate post dominator.

Suppose we treat the call blocs the same way as the elementary blocs. In this case, we can apply the algorithm given earlier to F and let $G = (V_N, V_T, P, S)$ be the resulting grammar. Let $G' = (V'_N, V'_T, P', S')$ denote the context-free grammar corresponding to the flow diagram F'. According to our construct, for each call bloc a G must contain a production $A \rightarrow a$, and this is the only production of G having the nonterminal A on the left hand side. The occurrence of the call bloc a in a path in F is marked in the corresponding label sequence by the occurrence of the symbol a. However, we are interested in representing execution paths of the composite system formed by the flow diagrams F and F', that is we must associate each occurrence of a call bloc with a path from s' to x' in F' (s' and x' denote the begin and the exit node of F', respectively). Let us call a flow diagram containing call blocs, such as F, a composite flow diagram. Let us also call F' an associate flow diagram of the (composite) flow diagram F.

A path in F is defined as in Definition 5.7.1 and a computation in F is defined as in Definition 5.7.5.

Definition 5.7.6

A composite computation in F is a computation in F where the occurrence of any call bloc a has been replaced by a sequence of nodes $a\gamma$, such that γ is a computation in F'.

We shall continue to call the grammar G obtained by applying our previous construct to the composite flow diagram F while treating all call blocs as elementary blocs, the context-free grammar corresponding to F. W(G) is the language of the label sequences corresponding to all possible computations in F. According to Definition 5.7.6, the language of the label sequences corresponding to all composite computations in F is $S(W(G))$ where S is the substitution mapping defined by:

$$S(a) = \begin{cases} \{a\}W(G') & \text{if } a \text{ is the label of a call bloc} \\ \{a\} & \text{otherwise} \end{cases}$$

Here, G' denotes the context-free grammar corresponding to the associate flow diagram F' of F.

One can easily construct the grammar $G'' = (V_N'', V_T'', P'', S'')$ such that $W(G'') = S(W(G))$ (we shall call G'' the composite context-free grammar corresponding to the flow diagram F). Without loss in generality we assume that $V_N \cap V_N' = \phi$. Let then $V_N'' = V_N \cup V_N'$, $V_T'' = V_T \cup V_T'$ and $S'' = S$. The set of productions P'' contains all the productions in P except for the productions $A \rightarrow a$, where a is the label of a call bloc in F. Moreover, P'' contains all the productions in P' and in addition, the set of productions:

$$\{A \rightarrow aS' \mid a \text{ is the label of a call bloc in } F\}.$$

It can easily be verified that $W(G'') = S(W(G))$. Note also that G'' remains a context-free grammar in Greibach normal form. (See Figure 5.7.2, for example.)

The construct given above can immediately be extended to the case where there are several associate flow diagrams of F. This method can also be used to represent the flow of control in recursive subroutines.

We shall call a composite flow diagram F recursive if it contains call blocs whose associated label pair is of the form (a, s) where s denotes the label of the begin bloc of F. Such a call bloc will be called a "recursive" call bloc.

A path in a recursive flow diagram F is defined as in Definition 5.7.1 and a computation in F is defined as in Definition 5.7.5. Given a sequence γ of nodes from F, the occurrence in γ of a recursive call bloc a is said to be a free occurrence of a if and only if the node immediately following a in γ (if any) is not s.

Definition 5.7.7

A composite computation is a recursive flow diagram F is:

i. a computation in F

ii. a composite computation in F in which a free occurrence of some recursive call bloc, say a, has been replaced by a sequence of nodes $a\gamma$, such that γ is a computation in F.

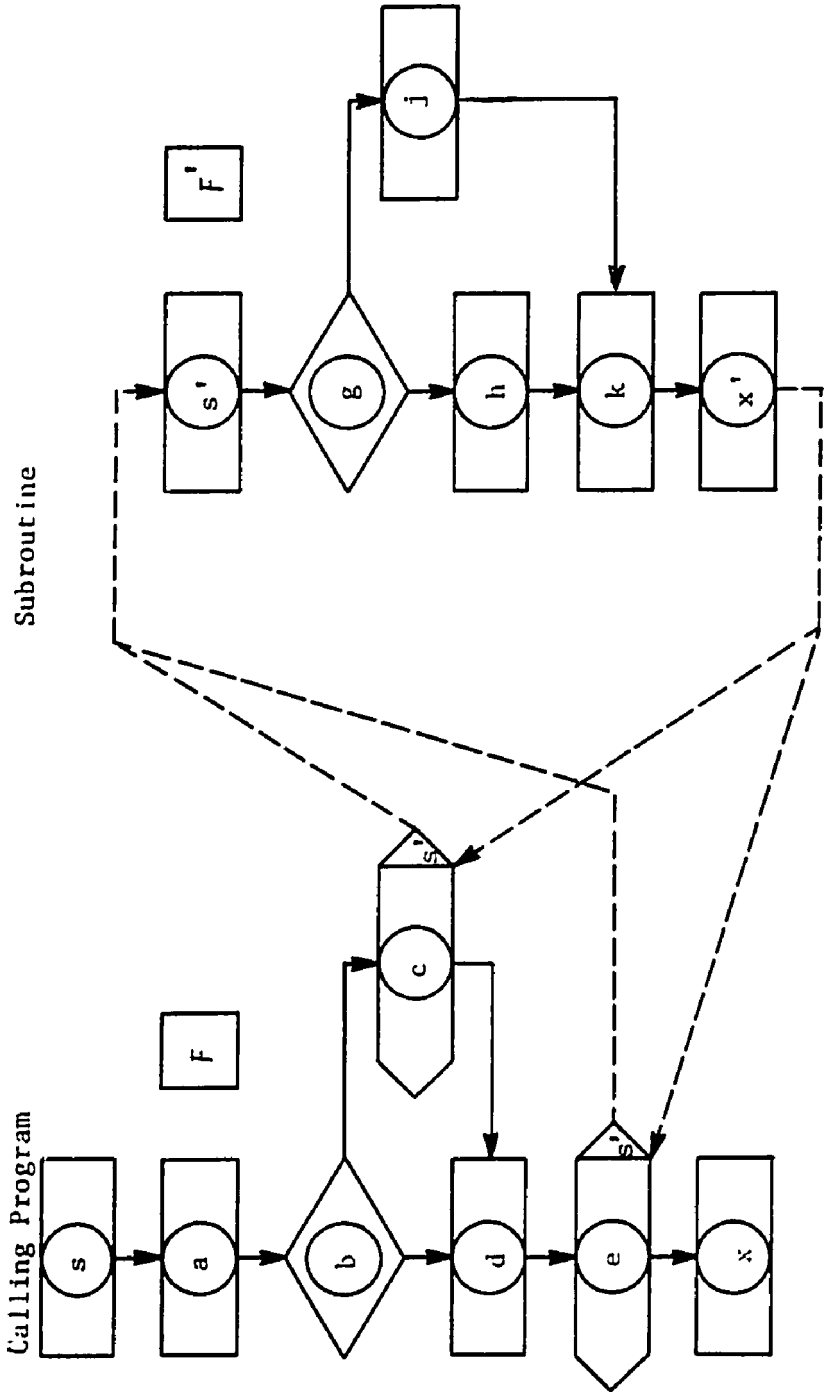


Figure 5.7.2

Representation of Subroutine Calls

P: $S \rightarrow sABDEX$

$A \rightarrow a$ $B \rightarrow bC$

$B \rightarrow b$ $D \rightarrow d$

$C \rightarrow c$ $E \rightarrow e$

$X \rightarrow x$

P': $S' \rightarrow s'GKX'$

$G \rightarrow gJ$ $G \rightarrow gH$

$H \rightarrow h$ $J \rightarrow j$

$K \rightarrow k$ $X' \rightarrow x'$

P": $S \rightarrow sABDEX$

$A \rightarrow a$

$B \rightarrow bC$

$B \rightarrow b$

$X \rightarrow x$

$C \rightarrow cS'$

$E \rightarrow eS'$

$S' \rightarrow s'GKX'$

$G \rightarrow gJ$ $J \rightarrow j$

$G \rightarrow gH$ $K \rightarrow k$

$H \rightarrow h$ $X' \rightarrow x'$

Figure 5.7.2 (Continued)

Definition 5.7.8

A composite computation δ in a recursive flow diagram F is said to be terminal if and only if δ does not contain any free occurrence of a recursive call bloc.

Note that a computation in F is a terminal (composite) computation in F if and only if it is a path from s to x in F which does not incorporate any recursive call bloc.

Let $G = (V_N, V_T, P, S)$ be the context-free grammar corresponding to the recursive flow diagram F. G is constructed by applying the algorithm given at the beginning of this section to F while treating the recursive call blocs in F as elementary blocs. W(G) is the language of the label sequences corresponding to all computations in F. The composite context-free grammar $G' = (V_N, V_T, P', S)$ corresponding to F, i.e., the grammar which generates the language of the label sequences corresponding to all terminal composite computations in F, is constructed as follows. G' is obtained from G by replacing the productions $A \rightarrow a$, where a is a recursive call bloc, by the productions $A \rightarrow aS$. As an example of this construct consider the recursive subroutine given in Figure 5.7.3. The subroutine search locates the variable item in the ordered vector array using the binary search method. When a match is found, the respective entry in the vector is deleted. At the same time, the subroutine returns in the parameter count the number of items which had to be inspected in order to find a match, i.e., the number of recursion levels needed in order to locate the given item. Figure 5.7.4 exhibits the uninterpreted flow diagram of this subroutine and the corresponding composite context-free grammar.

Consider an arbitrary recursive flow diagram F and let G' be the composite context-free grammar corresponding to F. Let us examine closer the representation of the flow of control in F by means of the productions in G'. We shall first make the following notational convention. The variable A denotes the upper case label of the node a, i.e., of the node whose lower-case label is a.

We define a restricted derivation in G' to be a derivation in G' such that at no step in the derivation has a production $A \rightarrow aS$ been used, where a is a recursive call bloc in F. A terminal sentential

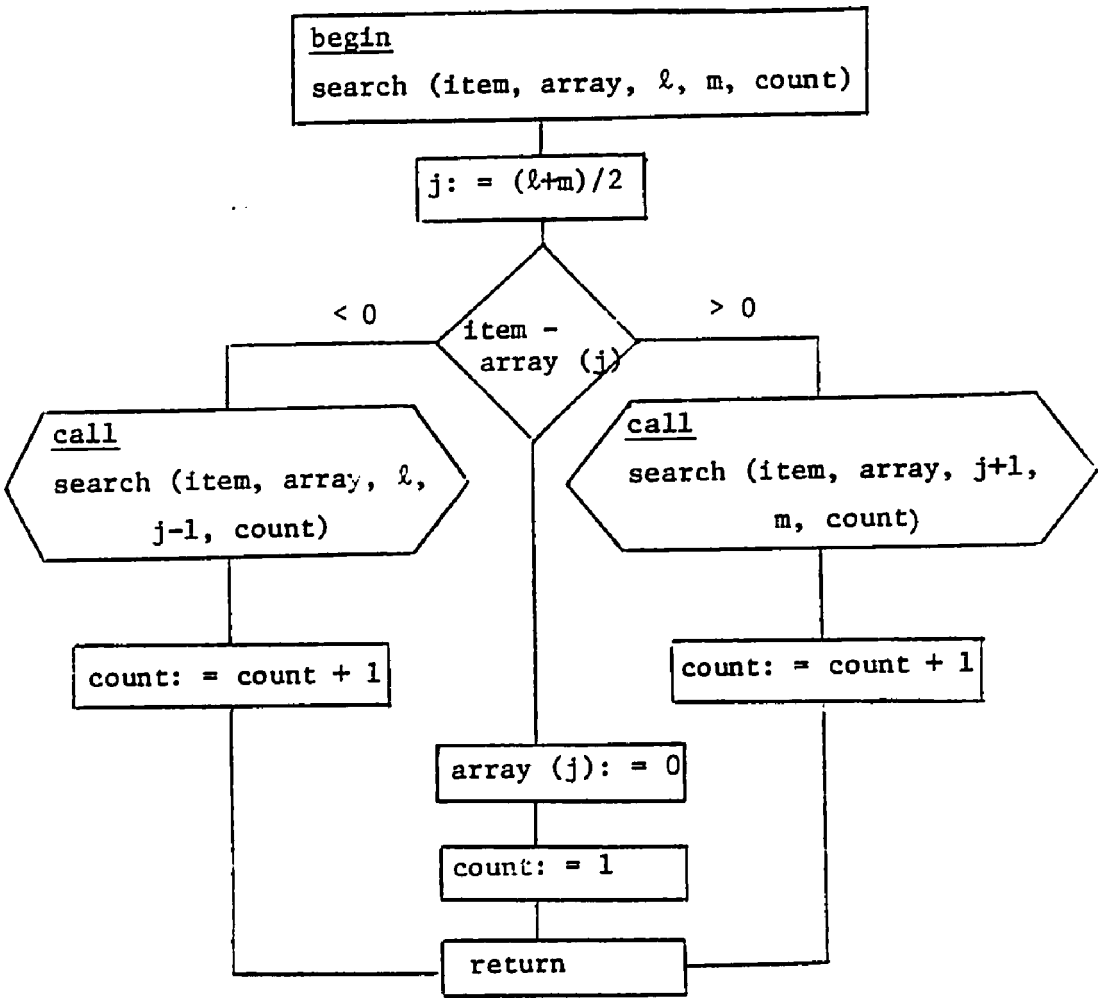
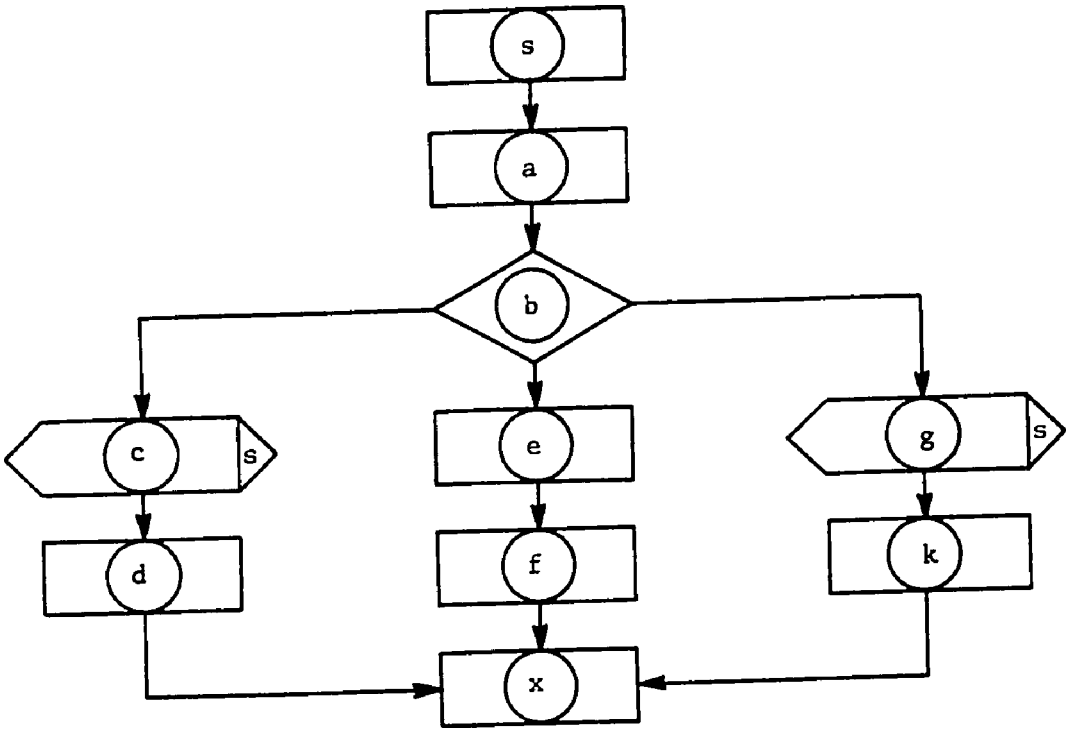


Figure 5.7.3

Sample Recursive Subroutine



P': S → sABX A → a K → k

B → bEF D → d X → x

B → bCD E → e C → cS

B → bGK F → f G → gS

Sample derivation:

S ⇒ sABX ⇒ saBX ⇒ sabCDX ⇒ sabcSDX ⇒

⇒ sabcsABXDX ⇒ sabcsaBXDX ⇒

⇒ sabcsabGKXDX ⇒ sabcsabgSKXDX ⇒

⇒ sabcsabgsABXKXDX ⇒

⇒ sabcsabgsaBXKXDX ⇒

⇒ sabcsabgsabEFXKXDX ⇒

$\overset{*}{\Rightarrow}$ sabcsabgsabefxkxdx

Figure 5.7.4

Uninterpreted Flow Diagram of the Recursive Subroutine of Figure 5.7.3

form is a sentential form δ such that:

i. A is a variable in δ only if a is a recursive call bloc in F.

ii. b is a terminal symbol in δ only if b is not a recursive call bloc in F.

Let us also define the homomorphism h on V_T by:

$$h(a) = \begin{cases} A & \text{if a is a recursive call bloc in } F \\ a & \text{otherwise} \end{cases}$$

We note that along a restricted derivation in G' only productions common to both G and G' can be used. Thus, the productions used along a restricted derivation in G' must belong to the set:

$$P - \{ A \rightarrow a \mid a \text{ is a recursive call bloc in } F\}.$$

(P denotes the set of productions of the context-free grammar G corresponding to F). Recall also that W(G) is the language of the label sequences corresponding to all possible computations in F. Hence, δ is a terminal sentential form obtained by means of a restricted derivation in G' if and only if $\delta = h(w)$, for some $w \in W(G)$. In particular, δ is a string in W(G') if and only if there exists a string $w \in W(G)$ such that $\delta = h(w) = w$. This happens, however, if and only if w is the label sequence corresponding to a computation in F which does not contain any recursive call blocs. Thus, a terminal sentential form obtained by means of a restricted derivation in G' is a string in W(G') if and only if δ is the label sequence corresponding to a computation in F which is terminal.

Consider now an arbitrary composite computation δ in F. For simplicity, let us assume that γ contains only one free occurrence of a recursive call bloc; the argument applies to the case where there are several free occurrences of recursive call blocs, as well. Thus, let $\gamma = \gamma_1 a \gamma_2$ where a denotes the free occurrence of the recursive call bloc a. Suppose that a sentential form δ can be reached in G' such that $\delta = \gamma_1 A \gamma_2$, that is the free occurrence of the recursive call bloc a has been marked in δ by the variable A. According to Definition 5.7.7 the composite computations which can be generated from γ are of the form $\gamma_1 a \gamma' \gamma_2$, where γ' is a computation in F. On the other hand, the only variable contained in the sentential form δ is A. The only

production with the nonterminal A on the left hand side contained in G' is $A \rightarrow aS$. Applying this production in δ we obtain the sentential form $\gamma_1 a S \gamma_2$. If γ' is a computation in F then there exists a restricted derivation in G' which leads to the terminal sentential form δ', such that $\delta' = h(\gamma')$. Thus, we can obtain in G' the sentential form $\gamma_1 a \delta' \gamma_2$ in which the variables contained mark the free occurrences of recursive call blocs in $\gamma_1 a \gamma' \gamma_2$. In particular, if γ' is a computation in F which does not contain any recursive call blocs, $\gamma_1 a \gamma' \gamma_2$ is a terminal composite computation in F and $\gamma_1 a \delta' \gamma_2 = \gamma_1 a \gamma' \gamma_2$ is a string in W(G').

We can conclude that if γ is a composite computation in F then there exists a sentential form δ which can be derived in G' such that the variables in δ are placeholders for the free occurrences of recursive call blocs in γ. Moreover, if γ is a terminal composite computation in F then $\delta = \gamma$ is a string in W(G'). A similar inductive argument can be used in order to show that if $w \in W(G')$ then there exists a terminal composite computation γ in F such that w is the label sequence corresponding to γ.

We note that the composite grammar G' corresponding to a recursive flow diagram F remains in Greibach normal form. The grammar G' represents the flow of control within the recursive flow diagram F. If F is invoked as a subroutine by some other composite flow diagram F', the flow of control in the composite system formed by F and F' is represented as before, namely as a substitution with W(G') in W(G''), where G'' is the composite context-free grammar corresponding to F'.

If the recursive flow diagram F also contains non-recursive call blocs then the composite grammar G' corresponding to F is further modified according to the construct presented for the representation of simple (non-recursive) subroutine calls.

We can conclude that the flow of control in programs calling subroutines, including recursive subroutines, can be represented using the substitution operation with λ-free context-free languages (note that according to our constructs the empty string λ cannot be generated by the context-free grammars associated with flow diagrams). We already know that the language family Λ_o(EC-CPM) contains all λ-free context-free languages and is closed under substitution. Consequently, the

EC-CPM can represent the flow of control in programs calling recursive subroutines. However, the construct presented in Section 3.3 for proving the closure under substitution of the language family Λ_o(EC-CPM) is rather complicated. Nevertheless, taking advantage of the special structure of context-free productions, the construct can be considerably simplified in the particular case of the stubstitution with λ-free context-free languages. In this way, the natural representation of the flow of control in programs can be preserved. In the remainder of this section we present the simplifications mentioned above. We first exhibit a language preserving modification on context-free grammars.

Let $G = (V_N, V_T, P, S)$ be a context-free grammar in Greibach normal form. Suppose S does not appear on the right hand side of any production in P (this condition can always be arranged for any context-free language). Let $V_N = \{ S, A_1, \ldots, A_m \}$, $V_N' = \{ B_1, \ldots, B_m \}$ and let us form the following set P' of productions:

i. If $S \rightarrow a_i A_{i1} \ldots A_{ik-1} A_{ik}$ is a production in P, where $k \geq 1$, then we introduce in P' the production $S \rightarrow a_i A_{i1} \ldots A_{ik-1} B_{ik}$. Any production $S \rightarrow a_i$ of P is transfered unmodified to P'.

ii. If $A_j \rightarrow a_j A_{j1} \ldots A_{jr-1} A_{jr}$ is a production in P, where $A_j \neq S$ and $r \geq 1$, then we introduce in P' the production $B_j \rightarrow a_j A_{j1} \ldots A_{jr-1} B_{jr}$. If $A_j \rightarrow a_j$ is a production in P then we introduce in P' the production $B_j \rightarrow a_j$.

iii. P' contains all the productions of the form $A_j \rightarrow a_j \gamma$ of P where $A_j \neq S$ (γ can be λ).

Let $G' = (V_N \cup V_N', V_T, P', S)$ be a new context-free grammar. Then, $W(G') = W(G)$.

Suppose $w \in W(G')$. If $w = a$, for some $a \in V_T$, then w can be generated only by the production $S \rightarrow a$, common to both G and G'. Hence, $w \in W(G)$. Suppose now that $|w| > 1$ and consider a leftmost derivation of w in G'. The first step in the derivation must be of the form:

$$S \Rightarrow a_1 A_{11} \ldots A_{1j-1} B_{1j}$$

According to the construction procedure of G', no production of G' having a variable $A \in V_N$ on the lefthand side, $A \neq S$, contains a variable $B \in V_N'$ on the right hand side. Thus, $A_{11}, \ldots, A_{1j-1}$ cannot

introduce any variable from V_N' in the derivation. Let then

$$S \Rightarrow a_1A_{11} \cdots A_{1j-1}B_{1j} \overset{*}{\Rightarrow} a_1\alpha_1B_{1j} \quad \text{where } \alpha_1 \in V_T^*.$$

On the other hand, any production in P' having B_{1j} on the left hand side is of the form $B_{1j} \rightarrow a_jA_{j1} \cdots A_{jr-1}B_{jr}$ or $B_{1j} \rightarrow a_j$. Thus, B_{1j} introduces at the most one variable from V_N' in the derivation, in the rightmost position of the sentential form. Let then

$$S \overset{*}{\Rightarrow} a_1\alpha_1B_{1j} \Rightarrow a_1\alpha_1a_2A_{21} \cdots A_{2r-1}B_{2r} \overset{*}{\Rightarrow} a_1\alpha_1a_2\alpha_2B_{2r} \overset{*}{\Rightarrow}$$
$$\overset{*}{\Rightarrow} a_1\alpha_1a_2\alpha_2 \cdots a_p\alpha_pB_{ps} \Rightarrow a_1\alpha_1a_2\alpha_2 \cdots a_p\alpha_pa_{p+1} = w$$

At the last step in the derivation, a production of the form $B_{ps} \rightarrow a_{p+1}$ has been used.

It is easy to see that we can form a leftmost derivation of w in G:

$$S \Rightarrow a_1A_{11} \cdots A_{1j-1}A_{1j} \overset{*}{\Rightarrow} a_1\alpha_1A_{1j} \Rightarrow a_1\alpha_1a_2A_{21} \cdots A_{2r-1}A_{2r} \overset{*}{\Rightarrow}$$
$$\overset{*}{\Rightarrow} a_1\alpha_1a_2\alpha_2A_{2r} \overset{*}{\Rightarrow} a_1\alpha_1a_2\alpha_2 \cdots a_p\alpha_pA_{ps} \Rightarrow a_1\alpha_1a_2\alpha_2 \cdots$$
$$\cdots a_p\alpha_pa_{p+1}$$

Hence, $w \in W(G)$ and thus $W(G') \subseteq W(G)$.

Using a similar argument it can be shown that $W(G) \subseteq W(G')$ as well and therefore $W(G) = W(G')$.

Note the following facts about the grammar G':

i. at each step of a derivation in G' the corresponding sentential form contains at the most one variable from V_N', in the rightmost position.

ii. for any $w \in W(G')$, where $|w| > 1$, the last production used in any leftmost derivation of w is of the form $B_j \rightarrow a$, for some $B_j \in V_N'$. Moreover, no production of this form has been used at any previous step in the derivation.

Consider now a λ-free context-free language W_a and suppose G' is a grammar in Greibach normal form which generates W_a. Let G be the context-free grammar obtained from G' after the application of the construct described above. G is still in Greibach normal form. According to Theorem 3.3.1 we can construct a Labelled EC-CPM ECP_{Σ_a} such that $\Lambda_o(ECP_{\Sigma_a}, UM^f) = W_a$. Moreover, UM^f is the zero marking. The typical structure of such an EC-CPM is exhibited in Figure 5.7.5. The

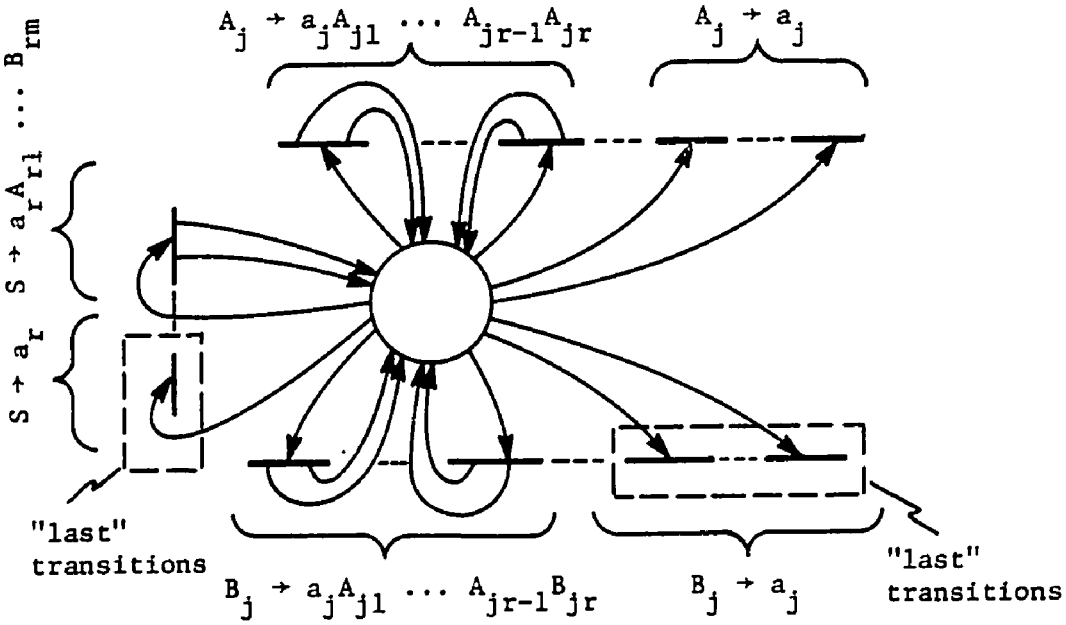


Figure 5.7.5

Sample Labelled EC-CPM Generating a Context-Free Language

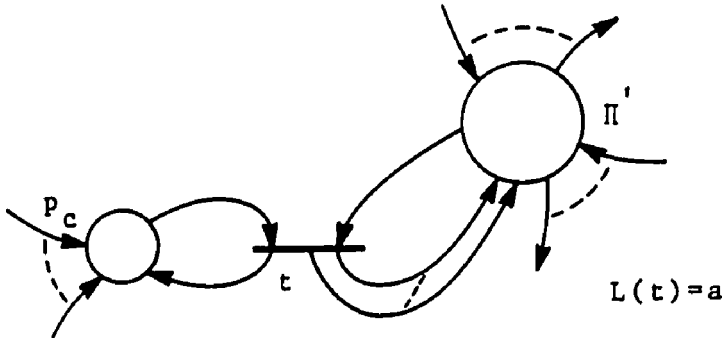


Figure 5.7.6

Sample Transition of a Labelled EC-CPM

net contains a distinct transition for each production of G. According to our discussion, given some string $w \in W_a$ the production used at the last step (and only at the last step) in any leftmost derivation of w is of the form $B_j \rightarrow a_j$ if $|w| > 1$ and $S \rightarrow a_r$ if $|w| = 1$. Recall also that the EC-CPM ECP_{Σ_a} simulates exactly leftmost derivations in G. Thus, we can identify the transitions which can fire last in any terminal firing sequence generated by ECP_{Σ_a}. More important, the firing of any such transition necessarily leads to the zero marking. This fact guarantees in case ECP_{Σ_a} is used as a substitution net that the net can be reutilized as many times as needed along a firing sequence of the EC-CPM in which the substitution is performed.

Let ECP_Σ be a Labelled EC-CPM which generates some context-free language. Let t be a transition of ECP_Σ and let a be the label of t. Suppose that we want to perform a substitution of the type encountered in this section. Thus, let $S(a) = \{a\}W_a$.

Figure 5.7.6 exhibits a sample transition t of ECP_Σ. We assume that all transitions of ECP_Σ self-loop on a control place p_c. The substitution net is given in Figure 5.7.7. Thus, we replace transition t by t' where t' has the same input and output connections as t exception making the places p_c and π. Therefore, the firing of t' has the same effect as the firing of t but t' does not restore the token in the control place p_c, leaving all the transitions of ECP_Σ disabled. Moreover t' places in π the initial marking of ECP_{Σ_a}. Only after ECP_{Σ_a} has executed a complete terminal firing sequence will the "last" transitions of ECP_{Σ_a} restore the token in the control place p_c allowing ECP_Σ to continue its firing sequence.

Note that if ECP_Σ contains several transitions with the same label a, the same construct is performed for each of them but only one copy of ECP_{Σ_a} is needed.

We can see that the construct for the substitution operation with context-free languages is in fact very simple and requires minimal modifications to the EC-CPM ECP_Σ. This fact is of importance in a top-down, hierarchial approach to the modelling of software systems.

We shall conclude this section with two final remarks. We have shown that context-free languages can represent the flow of control in flow diagrams, including recursive flow diagrams. The context-free

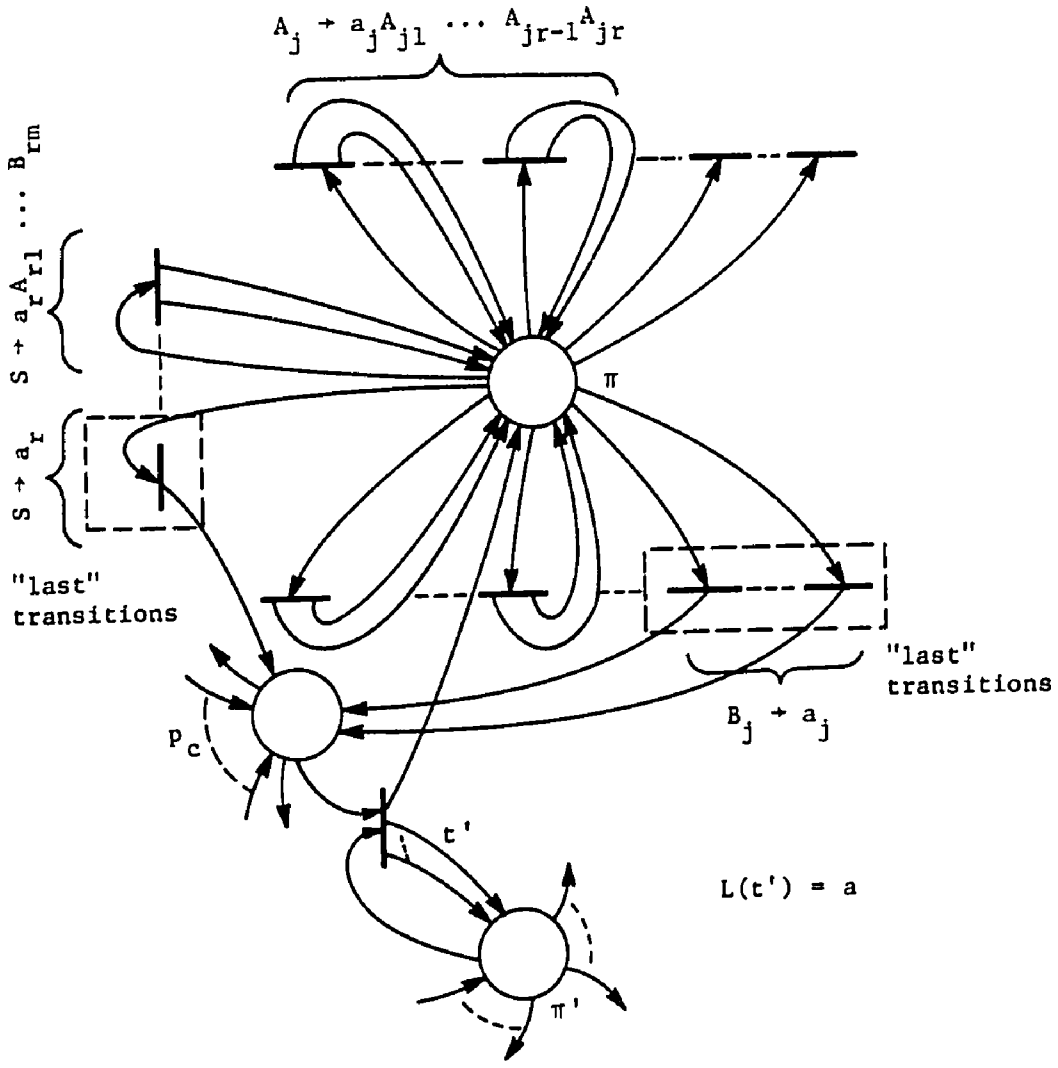


Figure 5.7.7

Substitution Labelled EC-CPM

grammars obtained by means of the algorithms presented here are in Greibach normal form. Thus, at each step in a derivation one terminal symbol is generated. Nevertheless, the order in which terminal symbols are generated along a derivation corresponds to the order in which nodes are passed by the flow of control in the corresponding computation only if the derivation is leftmost. This is exactly the approach taken in the EC-CPM.

Next, consider the recursive flow diagram F of Figure 5.7.4. We show that the language W of the label sequences corresponding to all terminal composite computations in F is not in Λ_o(C-CPM). Suppose for a moment that $W \in \Lambda_o$(C-CPM) and let us define the homomorphisms h_1 and h_2 by:

$$h_1(s) = s,\ h_1(a) = a,\ h_1(b) = b,\ h_1(e) = e,\ h_1(f) = f$$
$$h_1(x) = x,\ h_1(c) = c,\ h_1(g) = g,\ h_1(d) = c,\ h_1(k) = g$$
and $$h_2(s) = h_2(a) = h_2(b) = h_2(f) = h_2(x) = \lambda$$
$$h_2(c) = c,\ h_2(e) = e,\ h_2(g) = g$$

Obviously, h_1 is a λ-free renaming homomorphism while h_2 is a k-limited erasing on W (in fact, here k = 3).

The language family Λ_o(C-CPM) is closed under λ-free renaming homomorphism and by Theorem 4.2.2 Λ_o(C-CPM) is also closed under k-limited erasing. Hence, if $W \in \Lambda_o$(C-CPM) then $h_2(h_1(W))$ must be in Λ_o(C-CPM), as well. But $h_2(h_1(W)) = \{ wew^R \mid w \in \{c, g\}^*\}$ which we have already shown not to be Λ_o(C-CPM). Thus, we reach a contradiction.

We can conclude that the C-CPM and the other models examined in Chapter IV cannot model in a λ-free manner the flow of control in any recursive flow diagram.

Section 5.8 A PRODUCER-CONSUMER SYNCHRONIZATION PROBLEM WITH GENERALIZED QUEUEING MECHANISM.

Consider the following producer-consumer synchronization problem. Suppose the coordination system contains two producer processes, P_1 and P_2, and two consumer processes, C_1 and C_2. All processes are connected to a shared buffer B. The buffer B is structured into "slots" such that each slot may contain either a single item produced by P_1 or an arbitrary number of items produced by P_2 but no other combination of items.

A producer process may be activated at any time and it produces and

deposits one item in the buffer B. Producer P_1 always creates a new slot at the top of the buffer in which it deposits the newly produced item. On the other hand, producer P_2 first checks the content of the buffer B. If B is empty or the slot currently at the top of the buffer B contains an item produced by P_1 then P_2 creates a new slot, the new top slot of B, in which it deposits the item it has just produced. Otherwise, P_2 deposits the new item in the slot currently at the top of B. The consumer processes C_1 and C_2 become active as soon as B is nonempty and they attempt to consume items from the slot at the bottom of the buffer B. C_1 can consume only items produced by P_1 while C_2 can consume only items produced by P_2. Thus, the buffer B is accessed similar to a FIFO queue except that the slots of the queue may contain more than one element.

The producer-consumer synchronization problem described above can be further extended. The coordination system may involve several producer-consumer process pairs, possibly partitioned into two classes. The producer-consumer pairs belonging to one class behave like the process pair P_1, C_1 while the producer-consumer pairs belonging to the other class behave similar to the pair P_2, C_2. In this way, a slot of the shared buffer B may contain either one item produced by a producer process from the first class or an arbitrary number of items produced by various producer processes belonging to the second class.

The EC-CPM representation of the producer-consumer synchronization problem described at the beginning of this section is given in Figure 5.8.1. There, the place p_7 represents the buffer B. The transitions t_1 and t_2 model the processes P_1 and C_1, respectively. The transition t_6 models the consumer process C_2 while t_3, t_4 and t_5 model the producer process P_2. Thus, t_3 simulates the activation of P_2 in case the buffer B is empty and transitions t_4 and t_5 model the activation of P_2 in case the top slot of B contains an item deposited by P_1 and in case the top slot of B contains items produced by P_2, respectively. The place p_8 acts as a counter of the number of items contained in the buffer B.

The EC-CPM can model correctly this synchronization problem mainly due to its capabiltiy of structuring the color bags of its places according to some desired queueing discipline. Nevertheless, one may (justifiably) argue that the counter at the place p_8 does not have a

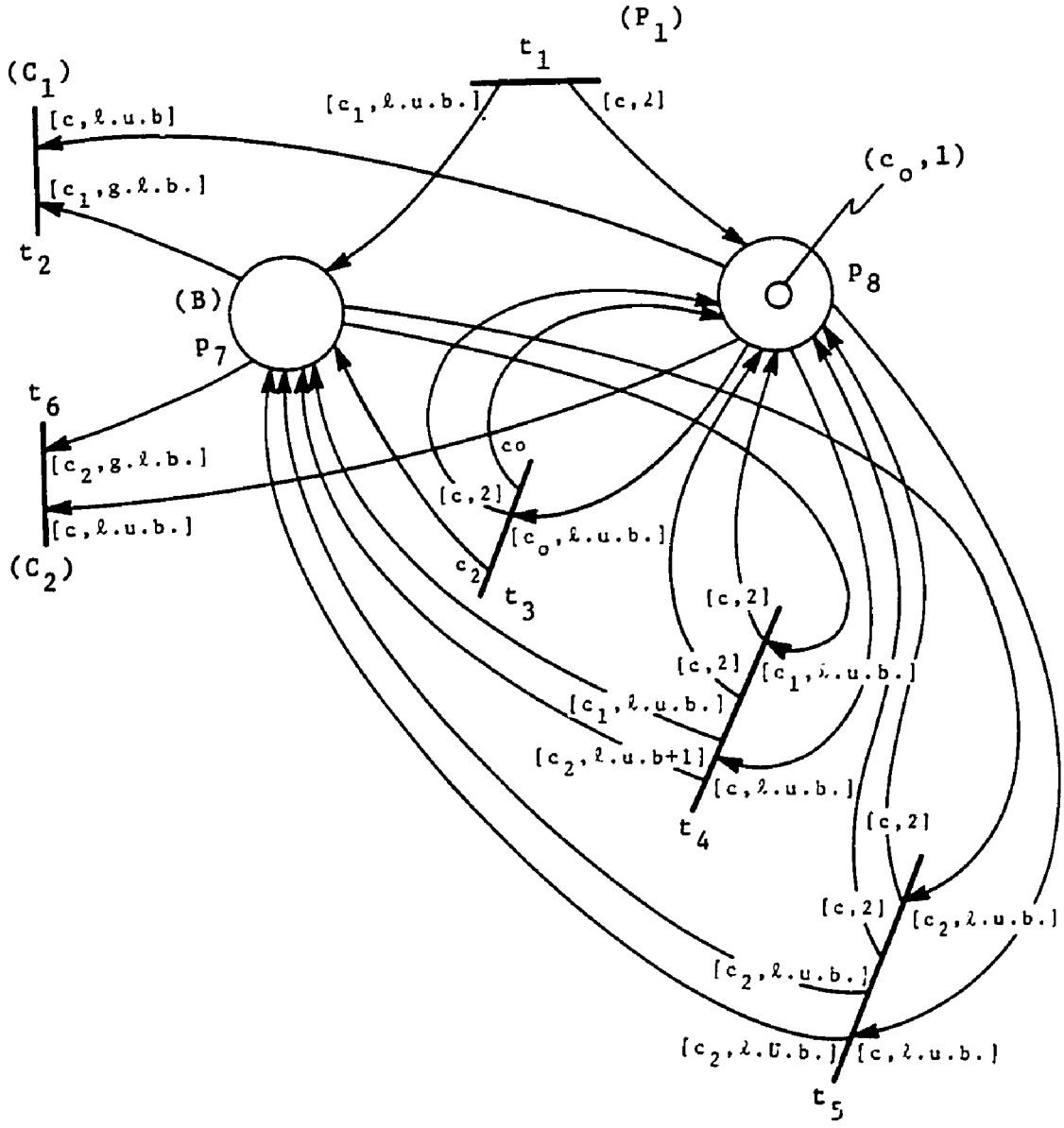


Figure 5.8.1

EC-CPM of a Producer-Consumer Synchronization System with Generalized Queueing Mechanism

direct counterpart in the modelled system. One could also object to the use of three transitions for the modelling of the producer process P_2. We note that the place p_8 has been introduced as one possible solution to the modelling of the operation of the process P_2 in case the buffer B is empty. It appears that if the EC-CPM would employ more sophisticated "f" and "g" mappings as color threshold and/or color output functions, these defficiencies could be avoided. This example thus goes to show that there are synchronization situations which could be modelled in a more natural way by classes of the CPM even more complex than the EC-CPM.

The producer-consumer synchronization problem described in this section was not artificially created, but, as in the case of the other modelling examples examined in this chapter, can actually be encountered in computing systems. Consider, for example, the computer architecture developed in [DENN-70] in order to "illustrate the principles of a machine with multiple functional units, especially the central processor of the Control Data 6600."

The processor incorporates four value registers which hold data values. Two functional units perform arithmetic operations on the values contained in the value registers. The system also incorporates a load/store unit, which transfers data between the value registers and main memory, and a transfer unit, which implements conditional control transfer instructions. An instruction queue unit fetches instructions from memory and assigns them for execution to the appropriate execution units. The system can be extended by adding more value registers and more execution units, either homogeneous or specialized.

Instructions are assigned to execution units exactly in the sequence specified by the respective program. Moreover, no instruction is allocated until it is known that its execution is required. The main control problem is to sequence the access to the value registers of the execution units operating in parallel in such a way that the intent of the program is preserved. This task is accomplished by the part of the control structure called the scoreboard. The scoreboard enforces following constraints with respect to each value register:

1. Instructions which modify the content of the register are executed in the order in which they are allocated.

2. The sequencing of instructions which modify the content of the value register and of instructions which examine the content of the value register at allocation is preserved.

The scoreboard contains one identical section for each value register. Such a section can be viewed as a queue of the type described at the beginning of this section. The instruction allocator deposits items at the top of the queue in accordance with the instructions allocated to the execution units. Each item carries a tag which identifies the execution unit and the type of access it has to perform on the respective value register (the execution unit can either modify or examine the content of the value register). Execution units consume items from the bottom slot of the queue. An execution unit can proceed only when it has collected the items corresponding to the register accesses incorporated in the instruction it has been allocated.

Note that instructions which only examine the content of a value register may be executed in any order without altering the intent of the program. Therefore, items corresponding to register content examine operations which are issued in consecutive order by the instruction allocator are placed in the same slot at the top of the queue. In this way, when that slot advances to the bottom position of the queue, the respective items are simultaneously available to the pertinent execution units. On the other hand, items which correspond to register accesses which modify the content of the register are always deposited in a new slot at the top of the queue in order to satisfy constraint 1 above. In order to satisfy constraint 2, two items corresponding to register accesses which examine and modify the content of the register, respectively and which are issued in consecutive order by the instruction allocator are also deposited in distinct slots at the top of the queue (in the order in which they were issued).

In this perspective, the instruction allocator can be viewed as a collection of producer processes, where each producer process generates items with a fixed tag. Moreover the producer processes are divided into two disjoint classes. The processes in the first class are the "write" processes, i.e., they generate items corresponding to register accesses which modify the content of the value register. The processes

in this class behave like the producer process P_1 of our synchronization problem. The processes in the second class are the "read" processes, that is they generate items corresponding to register accesses which examine but do not alter the content of the value register. The read processes operate the same way as the producer process P_2. On the other hand, each execution unit can be viewed as a collection of consumer processes, such that there exists a consumer process which can be paired with either a write or a read producer process.

CHAPTER VI

CONCLUSIONS

Section 6.1 OVERVIEW OF THE RESEARCH

At the beginning of this report we have suggested that the use of colors to represent distinct control flow streams operating simultaneously in concurrent systems would enable a simpler and more natural representation of the flow of control in various synchronization structures, in particular of reentrancy and recursivity in programs. Our objective was to develop a model of concurrent systems having the capability of handling "colored" representations of control flow and to analyze the modelling power of the respective model. Let us now examine to what extent we have succeeded in achieving these goals.

We have defined the Colored Petri Model as a general framework for the modelling of systems exhibiting concurrency. Deliberately, as few constraints as possible were incorporated in the CPM in order to obtain a general model which then can be further restricted according to the modeller's needs. We in turn, starting with synchronization structures of practical significance, have singled out two classes of the CPM, namely the C-CPM and the EC-CPM. The representation powers of the C-CPM and of the EC-CPM have been analyzed by studying the closure properties and the extents of the family of computation sequence sets and of the family of terminal computation sequence sets corresponding to these two classes of the CPM.

From our study it follows that the Λ and Λ_0 language families generated by the EC-CPM preserve and for certain operations extend the closure properties of the corresponding language families generated by the GPN's. Thus, even though the EC-CPM is a more complex model it not only preserves but in certain cases amplifies the modelling flexibility offered by the GPN's. This fact is important when one is interested in the top-down hierarchical modelling of large concurrent systems. More important, from the analysis of the extents of the language families Λ(EC-CPM) and Λ_0(EC-CPM) it follows that the EC-CPM is strictly more powerful, as far as its representation capabilities are concerned, than the C-CPM, EPM, PPM, COPM and the

Petri Net Model with switches, disjunctive logic and token absorbers. Hence, there are synchronization systems whose coordination sequences can be modelled without the aid of λ-transitions by the EC-CPM but not by any of the latter models (recall also that the C-CPM and the other formal models examined in Chapter IV are already strictly more powerful than the GPN's, the Complex GMC, the Finite State Parallel Program Schemata, etc.). Various coordination structures which can be modelled in a natural manner by the EC-CPM and which do require its additional representation power were presented in Chapter V. In particular, we have shown that the EC-CPM can indeed model in a natural and systematic way the flow of control in recursive programs.

Regarding the C-CPM, we have related the language families Λ(C-CPM) and Λ_0(C-CPM) to the corresponding language families generated by the EPM, PPM, COPM and the Petri Net Model with switches, disjunctive logic and token absorbers. During our study of the C-CPM, we have introduced various variants of the above models such as the EPM(I), EPM(II), PPM(I) and the C-CPM(I). These new models were not only useful in the constructs we have employed but also contain a few interesting features of their own. Our study of the C-CPM has led to some rather interesting results. Thus, we have concluded that the "conflict structure" as well as the "choice" in the selection of enabling colors incorporated in the C-CPM do not extend the representation capabilities of this model with respect to the representation capabilities of the less complicated PPM, for example. Also, the representation capabilities of the C-CPM are not altered if more sophisticated color threshold and color output functions than the constant mappings of Definition 2.4.1 are used. Other features, such as the positive testing arcs or the token absorbers do not affect the language families in question, either. We have also shown that the C-CPM and the other formal models considered in Chapter IV can simulate the behavior of arbitrary EC-CPM's only with considerable difficulty. Nonetheless, we have presented examples of coordination systems which can be modelled in a natural way by the C-CPM and which do not require the additional representation power of the EC-CPM. At the same time, we have argued that the respective

coordination systems cannot be modelled in a simple and natural manner by the EPM, PPM, COPM or the Petri Net Model with switches, disjunctive logic and token absorbers. In particular, the C-CPM has been shown to be well-suited to the modelling of reentrancy.

Inevitably, our work has not covered all aspects of the problem under study. The following are a few of the more challenging open research problems:

1. Find a characterization of the language families Λ(EC-CPM) and Λ_0(EC-CPM). One possibility would be to eventually relate these language families to restricted types of scattered context grammars ([GREI-69]).

2. Determine whether $\Lambda(\text{EPM(II)}) \subseteq \Lambda(\text{C-CPM})$.

3. Determine whether it is possible to convert Labelled EC-CPM's and/or Labelled C-CPM's in a λ-transition free, language preserving manner into restricted form, i.e., no multiple arcs and no self-loops.

APPENDIX A

BAG NOTATIONS AND DEFINITIONS

This appendix presents the concept of a bag and pertinent notations and definitions. For a more detailed discussion on this topic, the reader is referred to [CERF-72]. We mention, however, that we have introduced a few new definitions and that we have altered some of the definitions with respect to [CERF-72] in order to better serve our purpose.

Let P(x) denote any predicate and let S be the set defined by it, i.e. $S = \{x \mid P(x)\}$. Let $F: S \rightarrow Z^0$ be a function.

Definition A.1

The bag B over the set S determined by the function F is a collection of elements of S where:

i. If $F(b) > 0$ then b is an element of the bag B, denoted $b \in B$.

ii. If $F(b) = 0$ then b is not an element of the bag B, denoted $b \notin B$.

For each $b \in S$, F(b) is the number of occurrences of b in the bag B and is also denoted by #(b, B). Formally, the bag B is specified as follows:

$B = \langle x^y \mid x \in S \text{ and } y = \#(x, B) = F(x) \rangle$.

A bag is similar to a set except that it may contain more than one occurrence of the same element. The order of occurrence of the elements in the collection which forms the bag is not important.

Alternate ways of writing the bag B are:

$B = \langle x^{F(x)} \mid P(x) \rangle$

or $B = \langle x^y \mid P(x), y = F(x) \rangle$

or $B = \langle x \mid P(x), F(x) \rangle$.

If the function F is understood, it may be omitted from the specification of the bag B. In this case:

$B = \langle x \mid P(x) \rangle$.

Sometimes, it may be more convenient to specify the bag B by simply listing the collection of elements which forms the bag B. For example, let $S = \{a,b,c,d\}$ and let $F(a) = F(b) = 2$, $F(c) = 3$ and

$F(d) = 0$. The following are equivalent descriptions of the bag B over the set S determined by the function F:

$\langle a,a,b,b,c,c,c\rangle$,

$\langle a,b,a,b,c,c,c\rangle$,

$\langle c,c,a,b,c,b,a\rangle$,

$\langle a^2,b^2,c^3,d^0\rangle$,

$\langle a^2,c^3,b^2\rangle$,

$\langle b,b,a^2,c^3\rangle$, etc.

Definition A.2

The domain of the bag B is the set:

$\mathcal{D}(B) = \{b \mid b \in B\} = \{b \mid b \in S \text{ and } F(b) = \#(b, B) > 0\}$

The empty bag ϕ is a bag which contains no items.

Definition A.3

The bag closure of the set S is the set of bags:

$\langle S\rangle^* = \{B \mid \mathcal{D}(B) \subseteq S\}$

Obviously, the bag closure of the set S contains all possible bags which can be formed with the elements of S. The bags in $\langle S\rangle^*$ differ only by the corresponding functions $F: S \to Z^o$.

Definition A.4

Two bags, B_1 and B_2, are equal if and only if:

i. $\mathcal{D}(B_1) = \mathcal{D}(B_2) = D$.

ii. For each element $b \in D$, $\#(b, B_1) = \#(b, B_2)$.

Definition A.5

A bag B_1 is a subbag of a bag B_2, denoted by $B_1 \subseteq B_2$, if and only if for each item $b \in \mathcal{D}(B_1)$

$\#(b, B_1) \leq \#(b, B_2)$

Obviously, $\mathcal{D}(B_1) \subseteq \mathcal{D}(B_2)$ if $B_1 \subseteq B_2$.

B_1 is a proper subbag of B_2, denoted by $B_1 \subset B_2$, if and only if:

i. $B_1 \subseteq B_2$

ii. There exists an element $b \in \mathcal{D}(B_2)$ such that $\#(b, B_1) < \#(b, B_2)$.

Definition A.6

Given two bags B_1 and B_2 the bag B is the union of B_1 and B_2, i.e., $B = B_1 \cup B_2$, if and only if for each element $b \in \mathcal{D}(B)$

$\#(b, B) = \#(b, B_1) + \#(b, B_2)$

Obviously, $\mathcal{D}(B) = \mathcal{D}(B_1) \cup \mathcal{D}(B_2)$.

B is the <u>intersection</u> of B_1 and B_2, i.e., $B = B_1 \cap B_2$, if and only if for each element $b \in \mathcal{D}(B)$

$\#(b, B) = \min\{\#(b, B_1), \#(b, B_2)\}$

Obviously, $\mathcal{D}(B) = \mathcal{D}(B_1) \cap \mathcal{D}(B_2)$

B is the <u>difference</u> of B_1 and B_2, i.e., $B = B_1 - B_2$, if and only if for each element $b \in \mathcal{D}(B)$

$\#(b, B) = \max\{\#(b, B_1) - \#(b, B_2), 0\}$

Obviously, $\mathcal{D}(B) = \{b \mid \#(b, B_1) > \#(b, B_2)\}$

APPENDIX B

TAPE AND TIME COMPLEXITY OF THE LANGUAGE FAMILIES Λ(EC-CPM) AND Λ_o(EC-CPM)

We begin this appendix by giving a concise definition of the Turing machine model employed in our proofs.

Definition B.1

An n-tape Turing machine is a (n+6)-tuple $TM = (K, \Sigma, \Gamma_1, \ldots, \Gamma_n, \delta, q_o, F, n)$ where:

i. $K, \Sigma, \Gamma_1, \ldots, \Gamma_n$ are finite sets, $q_o \in K$, $F \subseteq K$ and n is a non-negative integer.

ii. δ is a function from $K \times (\Sigma \cup \{\lambda\}) \times ((\Gamma_1 \cup \{\lambda\}) \times \ldots \times (\Gamma_n \cup \{\lambda\}))$ into the finite subsets of $K \times ((\Gamma_1 \times \{1, 0, -1\}) \cup \Gamma_1^*) \times \ldots \times ((\Gamma_n \times \{1, 0, -1\}) \cup \Gamma_n^*)$.

TM is deterministic if for all $q \in K$, $A_i \in \Gamma_i \cup \{\lambda\}$

i. $\delta(q, \lambda, A_1, \ldots, A_n) \neq \phi$ implies $\delta(q, a, A_1, \ldots, A_n) = \phi$, for all $a \in \Sigma$, and

ii. for all $a \in \Sigma \cup \{\lambda\}$, $|\delta(q, a, A_1, \ldots, A_n)| \leq 1$.

Definition B.2

A configuration of TM is a (2n+2)-tuple $(q, w, y_1, \ldots, y_n, i_1, \ldots, i_n)$ where:

i. $q \in K$, $w \in \Sigma^*$

ii. $y_k \in \Gamma_k^*$, $0 \leq i_k \leq |y_k|$ and $i_k = 0$ if and only if $y_k = \lambda$, for all k, $1 \leq k \leq n$.

The relation $\vdash$ between configurations is defined as follows. Let $(q, aw, y_1, \ldots, y_n, i_1, \ldots, i_n)$ be a configuration, $a \in \Sigma \cup \{\lambda\}$. Let $(q', \sigma_1, \ldots, \sigma_n)$ be in $\delta(q, a, A_1, \ldots, A_n)$. Then, $(q, aw, y_1, \ldots, y_n, i_1, \ldots, i_n) \vdash (q', w, y_1', \ldots, y_n', i_1', \ldots, i_n')$ if for each k, $1 \leq k \leq n$, either:

1. $A_k \in \Gamma_k$, $\sigma_k = (B, i)$, $y_k = xA_ky$, $y_k' = xBy$, $i_k = |x| + 1$, $0 < i_k' = i_k + i \leq |y_k|$.
2. $A_k = \lambda = y_k$, $\sigma_k = Z_k \in \Gamma_k^*$, $y_k' = Z_k$ and $i_k' = |Z_k|$
3. $A_k \in \Gamma_k$, $\sigma_k = Z_k \in \Gamma_k^*$, $y_k = xA_k$, $i_k = |x| + 1$, $y_k' = xZ_k$ and $i_k' = |xZ_k|$.

The relations $\vdash^{m}$ ($m \geq 0$) and $\vdash^{*}$ are defined as follows. For any configuration C, $C \vdash^{0} C$. If $C_0 \vdash C_1 \vdash \ldots \vdash C_m$ then $C_0 \vdash^{m} C_m$ and if $C_0 = (q_o, w, \lambda, \ldots, \lambda, 0, \ldots, 0)$ then $C_0, C_1, \ldots, C_m$ is called an m-step computation on w. If $C_m = (q, \lambda, \ldots, \lambda, 0, \ldots, 0)$ with $q \in F$ then it is an accepting computation (on w). For configurations C_1 and C_2, $C_1 \vdash^{*} C_2$ if $C_1 \vdash^{m} C_2$, for some $m \geq 0$. If $C_0 \vdash C_1 \vdash \ldots \vdash C_m$ and $k > 0$ is a constant such that for each configuration $C_j = (q, w_j, y_1, \ldots, y_n, i_1, \ldots, i_n)$ every y_r is such that $|y_r| \leq k$ then TM visits no more than k tape squares on any of its storage tapes in the computation $C_0, \ldots, C_m$.

Definition B.3

The language accepted by TM is the set:

$$W(TM) = \{w \in \Sigma^* \mid (q_o, w, \lambda, \ldots, \lambda, 0, \ldots, 0) \vdash^{*} (q, \lambda, \lambda, \ldots, \lambda, 0, \ldots, 0) \text{ for some } q \in F\}$$

Definition B.4

A multitape Turing acceptor TM operates within tape bound f if for each input string w accepted by TM, every accepting computation of TM on w visits no more than $\max\{|w|, f(|w|)\}$ tape squares on any of its storage tapes (f is a real-valued function of a real variable).

Let

$$\text{DetTAPE}(f) = \{W(TM) \mid TM \text{ is a deterministic multitape Turing acceptor which operates within tape bound } f\}.$$

Definition B.5

An online multitape Turing acceptor TM operates within time bound f if for each input string w accepted by TM, every accepting computation of TM on w has no more than $\max\{|w|, f(|w|)\}$ steps.

Let us now determine a suitable "Turing machine encoding" for an EC-CPM. Let $ECP_\Sigma = (ECP, \Sigma, L)$ be an arbitrary Labelled EC-CPM where $ECP = (CN, UM^o)$, $CN = (N, U(C), H)$ and $N = (T, P, I, O)$ is the underlying GPN.

We have already mentioned that given a color marking UM^i and a transition t_k of ECP_Σ if $UM^i[t_k > UM^{i+1}$ then the color marking UM^{i+1} is uniquely determined by UM^i and t_k. This remark can be extended to firing sequences as well. Thus, let $\gamma_k = t_{k_1} \ldots t_{k_m}$ be a nonempty

firing sequence in S(UM°). Suppose

$$UM^{\circ}[t_{k_1} > UM^{1} \ldots UM^{m-1}[t_{k_m} > UM^{m}$$

The color markings UM^j, $0 < j \leq m$, are uniquely determined by UM° and γ_k. In this sense, an EC-CPM behaves similarly to a deterministic automaton. On the other hand, in each of the color markings UM^{j-1} above, other transitions than t_{k_j} may also be enabled, some of which may be in conflict. Nevertheless, all enabled transitions have the same priority to be selected to fire in the respective color marking. From this point of view, ECP_Σ behaves like a nondeterministic automaton.

Let us assume that $T = \{t_0, t_1, \ldots, t_n\}$. Let $\tilde{\gamma_k}$ be the m-digit number (in the basis n+1) $k_1 k_2 \ldots k_m$ formed from the indices of the transitions contained in the firing sequence γ_k. Obviously, $0 \ldots 0 \leq \tilde{\gamma_k} \leq n \ldots n$. If γ_1 and γ_2 are distinct firing sequences then $\tilde{\gamma_1} \neq \tilde{\gamma_2}$. Moreover, by virtue of the remarks made above, one can associate with $\tilde{\gamma_k}$ the (unique) sequence of reachable color markings $UM^1, \ldots, UM^m$.

Let us now make a few observations regarding the way in which the color bags of the places of ECP_Σ vary along the firing sequence γ_k. This will enable us to determine the amount of Turing machine tape required in order to simulate γ_k.

Let:

$$O = \max\{O(t_k, p_j) \mid t_k \in T, p_j \in P\}$$

and $M^{\circ} = \max\{M^{\circ}(p_j) \mid p_j \in P\}$

Then, for all places $p_j \in P$,

$$M^m(p_j) \leq M^{\circ} + |\gamma_k| \cdot O$$

M° and O are constants depending on the initial color marking of ECP_Σ and the structure of the underlying GPN N, respectively. We conclude that the number of tokens present in any place of ECP_Σ at the end of the firing sequence γ_k is linearly bounded by the length of γ_k.

Let A denote the set of arcs of ECP_Σ and let:

$$Z = \max\{z_r \mid H(a^r_{jk}) = [c_r; z_r] \text{ for some arc } a^r_{jk} \in A, z_r \in Z^+\}$$

$$\ell = \max\{\ell.u.b.(\mathscr{D}^2(U^o_j)) \mid p_j \in P\}$$

$$R = \max\{Z, \ell\}$$

$$S = \max\{s_r \mid H(a^r_{jk}) = [c_r; \ell.u.b. + s_r] \text{ or } H(a^r_{jk}) = [c_r; g.\ell.b. + s_r] \text{ for some arc } a^r_{jk} \in A, s_r \in Z^{\circ}\}$$

Then, for all places $p_j \in P$,

$$\ell.u.b.(\mathcal{D}^2(U_j^m)) \leq R + |\gamma_k| \cdot S,$$

i.e. $\ell.u.b.(\mathcal{D}^2(U_j^m))$ is bounded by a linear function of the length of the firing sequence γ_k.

Let $C = (X, \leq)$ be the finite set of colors whose extension U(C) is the set of colors associated with ECP_Σ. Let us assume that $X = \{c_1, \ldots, c_u, c_m, c_M\}$. Let also U_j^r be the bag of colors corresponding to some place $p_j \in P$ in one of the color markings UM^r reachable along the firing sequence γ_k. We want to represent U_j^r on a Turing machine tape whose associated tape alphabet Γ contains the symbols $c_1, \ldots, c_u$ and the special markers $\$$ and $\not\in$. Let for all k, $1 \leq k \leq \ell.u.b.(\mathcal{D}^2(U_j^r)) = TOP$

$$w_k^r = < c_q^{n_{k_q}} \mid (c_q, k) \in \mathcal{D}(U_j^r) \text{ and } n_{k_q} = \#(c_q, w_k^r) =$$

$$= \#((c_q, k), U_j^r) >$$

Then the color bag U_j^r can be represented on tape as follows:

$$\$ \not\in w_1^r \not\in \ldots \not\in w_k^r \not\in \ldots \not\in w_{TOP}^r \not\in \$$$

If for some k $w_k^r = \phi$ then on tape we have the following configuration:

$$\ldots \not\in w_{k-1}^r \not\in \not\in w_{k+1}^r \not\in \ldots$$

Thus, the tape can contain substrings of adjacent $\not\in$ markers. For reasons to become apparent later on, we assume that within the bags w_k^r the colors are grouped together and placed in order, i.e.:

$$w_k^r = < c_{q_1}^{n_{k_{q_1}}}, c_{q_2}^{n_{k_{q_2}}}, \ldots, c_{q_s}^{n_{k_{q_s}}} >$$

where $1 \leq q_1 < q_2 < \ldots < q_s \leq u$.

With this preparation we can now determine the number X_j^m of tape squares needed in order to encode the color bag U_j^m, for any place $p_j \in P$. Thus,

$$X_j^m = M^m(p_j) + \ell.u.b.(\mathcal{D}^2(U_j^m)) + 3$$

Therefore,

$$X_j^m \leq M^\circ + |\gamma_k| \cdot 0 + R + |\gamma_k| \cdot S + 3 \leq \alpha \cdot |\gamma_k|$$

for some nonnegative integer constant α.

Thus, the number of tape squares needed in order to store the bag of colors of some place $p_j \in P$ along the simulation of some firing

sequence γ_k is linearly bounded by the length of γ_k. Note that this bound depends only on the length of the firing sequence γ_k and not on the particular firing sequence. Therefore, all equal-length firing sequences require the same amount of tape.

With the above preparation we can now proceed to construct a deterministic Turing acceptor TM such that $W(TM) = \Lambda(ECP_\Sigma)$. Based on our construct we shall determine the amount of tape required in order to accept any language from the language family Λ(EC-CPM). Finally, we shall extend our discussion to the language family Λ_o(EC-CPM).

Given a string $w \in \Sigma^*$, $w \in \Lambda(ECP_\Sigma)$ if and only if there exists a firing sequence $\gamma \in S(UM^\circ)$ such that $L(\gamma) = w$. Moreover, all such firing sequences must have the same length, $|w|$. Consequently, in order to check whether $w \in \Lambda(ECP_\Sigma)$, TM will simulate in turn firing sequences of ECP_Σ of length $|w|$ until either a firing sequence γ with $L(\gamma) = w$ is found or all such firing sequences are exhausted. In the latter case, obviously $w \notin \Lambda(ECP_\Sigma)$.

TM has $2 \cdot |P| + 3$ tapes, where P denotes the set of places of ECP_Σ. Tape k, $1 \leq k \leq |P|$, is used to store the color bag of the place p_k along the simulation of any firing sequence of ECP_Σ. For this purpose, the encoding presented earlier is used. Tape $|P| + k$, $1 \leq k \leq |P|$, is used as an auxiliary tape for rearranging the content of the tape k. In general, the pair of tapes k and $|P| + k$, for each k, $1 \leq k \leq |P|$, will work in a "head-to-head", pushdown manner.

Tape $2 \cdot |P| + 3$ is used to store the input string w. Tape $2 \cdot |P| + 2$ will contain at the end of the simulation of a firing sequence δ the corresponding label sequence $L(\delta)$. By comparing the contents of the tapes $2 \cdot |P| + 2$ and $2 \cdot |P| + 3$ we can determine whether the firing sequence δ generates the given string w.

Finally, TM will generate in turn on tape $2 \cdot |P| + 1$ all $|w|$-digit numbers in the basis $|T|$, starting with 0 ... 0. For each such number $k_1 \ldots k_{|w|}$, TM will attempt to deterministically simulate the firing sequence γ_k, where $\gamma_k^{\sim} = k_1 \ldots k_{|w|}$. This mechanism permits us to simulate all firing sequences of ECP_Σ of length $|w|$ in a deterministic manner.

In the initial configuration TM has all its storage tapes empty. TM starts its computation on w by depositing in one move the initial

color marking UM° of ECP_{Σ} on the tapes k, $1 \leq k \leq |P|$. Next, TM copies the input string w from its read-only input tape onto tape $2 \cdot |P| + 3$. At each step during this copy-process, TM also writes a digit 0 on tape $2 \cdot |P| + 1$ while moving the head on this tape to the right. Thus, at the end of this phase:

- the head on each tape k corresponding to some place $p_k \in P$ for which $UM^{\circ}(p_k) \neq \phi$ scans the rightmost nonempty position on the respective tape.

- the tapes $|P| + k$, for all k, $1 \leq k \leq |P|$, and the tape $2 \cdot |P| + 2$ are empty.

- tape $2 \cdot |P| + 1$ contains $|w|$ digits 0 and the respective head is positioned on the $|w|$-th tape square.

- tape $2 \cdot |P| + 3$ contains the input string w and the corresponding tape head is positioned on the tape square containing the rightmost symbol of w.

At this point TM is ready to simulate the firing sequence $\gamma = t_0 \ldots t_0$ ($\gamma \tilde{} = 0 \ldots 0$). The simulation of the firing sequence γ is performed by simulating in turn the firing of the transitions contained in γ, in the order in which the corresponding indices occur on the tape $2 \cdot |P| + 1$ from right to left (in this particular case, TM will attempt to simulate $|w|$ consecutive firings of the transition t_0). During the simulation of each transition firing, TM will write the corresponding label on the tape $2 \cdot |P| + 2$. At the end of the simulation of the firing sequence γ the tapes k, $1 \leq k \leq |P|$, will contain the color marking UM^m, where $UM^{\circ}[\gamma > UM^m$. On each such tape corresponding to some place $p_k \in P$ for which $UM^m(p_k) \neq \phi$, the corresponding tape head will be positioned on the rightmost nonempty tape square. At the same time, the tape $2 \cdot |P| + 2$ will contain the label sequence $L(\gamma)$ and the respective tape head is positioned on the rightmost symbol of $L(\gamma)$. TM will now scan in parallel the tapes $2 \cdot |P| + 2$ and $2 \cdot |P| + 3$ in order to compare $L(\gamma)$ with w. If $L(\gamma) = w$ then TM stops in an accepting configuration. Otherwise, TM "jumps" to the subroutine EXIT, preparing for the simulation of another firing sequence of length $|w|$. In the subroutine EXIT, TM increments the $|w|$-digit number contained on tape $2 \cdot |P| + 1$ by 1. Let N be the

new number contained on tape $2 \cdot |P| + 1$. If N is less than or equal to the $|w|$-digit number $|T|\ldots|T|$ then TM proceeds to simulate the firing sequence δ, for which $\delta^{\sim} = N$. Before doing so, however, TM moves the head on the tape $2 \cdot |P| + 3$ back to the right end of the tape, erases all its tapes but the tapes $2 \cdot |P| + 1$ and $2 \cdot |P| + 3$ and deposits on the tapes k, $1 \leq k \leq |P|$, the initial color marking UM^o.

The above algorithm is executed repeatedly until either a firing sequence δ' with $L(\delta') = w$ is found or the number reached on the tape $2 \cdot |P| + 1$ exceeds the $|w|$-digit number $|T|\ldots|T|$. In the latter case, obviously $w \notin \Lambda(ECP_\Sigma)$ and TM stops without accepting the input w.

We note that there may exist $|w|$-digit numbers N' such that $\gamma' \notin S(UM^o)$, where $\gamma'^{\sim} = N'$. Let us suppose that $\gamma' = t_{k_1} \ldots t_{k_r} t_{k_{r+1}} \ldots t_{k_{|w|}}$ where $UM^o[t_{k_1} \ldots t_{k_r} > UM^r$ and the transition $t_{k_{r+1}}$ is not enabled in the color marking UM^r. In this case, TM will be able to simulate the prefix firing sequence $t_{k_1} \ldots t_{k_r}$ but will not be able to complete the simulation of the firing of the transition $t_{k_{r+1}}$ in the color marking UM^r (the simulation of the firing of a transition is explained below). Consequently, TM will simply abort the simulation of the firing sequence γ' and will proceed to execute the subroutine EXIT.

We conclude the description of TM by explaining the simulation of the firing of some aribtrary transition t_r of ECP_Σ. First, TM writes the label $L(t_r)$ on the tape $2 \cdot |P| + 2$ while advancing the respective tape head one position to the right. Next, TM checks whether t_r is enabled in the current color marking. Let p_k be an input place of t_r. For simplicity, let us assume that $I(p_k, t_r) = 3$ and that $H(a^1_{kr}) = [c^1_r;\ \ell.u.b. - s^1_r]$, $H(a^2_{kr}) = [c^2_r;\ g.\ell.b + s^2_r]$ and $H(a^3_{kr}) = [c^3_r;\ s^3_r]$. Assuming that in the current color marking, let it be UM^j, the place p_k is not empty, the head on the corresponding tape k is placed on the rightmost nonempty tape square (which contains a \$ marker). TM checks first whether there exists an enabling token (of color $(c^1_r,\ \ell.u.b.(\mathscr{D}^2(U^j_k)) - s^1_r) = (c^1_r,\ TOP - s^1_r)$) for the arc a^1_{kr}. In order to do so, TM scans the tape k from right

to left, counting the number of ¢ markers encountered. The fixed integer s_r^1 is stored in the finite memory of the control of TM. After the head on tape k has passed $s_r^1 + 1$ ¢ markers it reaches the color bag $w_{TOP} - s_r^1$. The head continues to scan through the bag $w_{TOP} - s_r^1$ until it encounters the first symbol c_r^1, in which case it replaces that symbol by a special "erase" marker #. We recall that the colors in the bag $w_{TOP} - s_r^1$ are grouped together and placed in order. This rule is meant to facilitate the search through the color bag $w_{TOP} - s_r^1$ if, for example, there are several input arcs of t_r from p_k which request the same enabling color, $(c_r^1, TOP - s_r^1)$. After the symbol c_r^1 has been "marked" TM continues to move left on the tape k, without counting the ¢ markers, until the left end marker $ is reached.

Next, TM checks whether there exists an enabling token (of color $(c_r^2, g.\ell.b.(\mathscr{D}^2(U_k^j)) + s_r^2) = (c_r^2, BOT + s_r^2)$) for the input arc a_{kr}^2. In order to accomplish this, the head on tape k is moved right. Adjacent ¢ markers immediately following the left end marker $ (if any) are ignored. As soon as the respective tape head encounters a ¢ marker immediately followed by a symbol other than a ¢ marker, TM starts counting the number of ¢ markers passed by the tape head. After it has passed $s_r^2 + 1$ ¢ markers, the head on tape k reaches the color bag $w_{BOT + s_r^2}$. The first symbol c_r^2 encountered is replaced by the "erase" marker #. From this point on, TM simply moves the head on the tape k to the right, without counting ¢ markers, until the right end marker $ is encountered.

In order to check whether there exists an enabling token for the input arc a_{kr}^3, TM first moves the head on the tape k to the left end of that tape. Next, TM proceeds as in the case of the input arc a_{kr}^2 except that adjacent ¢ markers immediately following the left end marker $ are also counted.

After TM has marked the "enabling tokens" for the input arcs of t_r from the place p_k it proceeds to erase the marked symbols (this corresponds to t_r removing its enabling tokens when firing). The head of the tape k scans the content of that tape from right to left. At each step, the symbol read on the tape k is erased from the tape k and written on the tape $|P| + k$ (the tape $|P| + k$ is empty at this

stage). Obviously, the head on the tape $|P| + k$ moves in opposite direction i.e., from left to right. Whenever a # marker is encountered, it is erased from the tape k but it is not copied on the tape $|P| + k$. When the head on the tape k erases the left end marker \$, tape k becomes empty while tape $|P| + k$ contains the color bag:

$$U_k^j - \langle (c_r^1, \text{TOP} - s_r^1), (c_r^2, \text{BOT} + s_r^2), (c_r^3, s_r^3) \rangle .$$

At this point the roles of the tapes k and $|P| + k$ are reversed, that is the content of the tape $|P| + k$ is copied back on the tape k while the tape $|P| + k$ is erased. Note that the process of removing "the bag of enabling tokens of t_r from the input place p_k" may generate adjacent ¢ markers immediately preceding the right end marker \$. All such ¢ markers are now erased from the tape k (without using the auxiliary tape $|P| + k$). This operation may require an additional scan of the tape k.

We also note that "enabling tokens" of t_r cannot be erased from the tape k before all such "tokens" have been marked. This is so because the removal of an enabling token may generate an intermediate color bag $U_k^{j'}$ for which g.ℓ.b.($\mathscr{D}^2(U_k^{j'})$) and/or ℓ.u.b.($\mathscr{D}^2(U_k^{j'})$) are different from g.ℓ.b.($\mathscr{D}^2(U_k^j)$) and ℓ.u.b.($\mathscr{D}^2(U_k^j)$) respectively, which in turn would influence the selection of the remaining enabling tokens.

If TM cannot mark an "enabling token" for some input arc of the transition t_r then t_r is not enabled in the current color marking. In this case, as mentioned earlier, TM aborts the simulation of the firing of t_r and proceeds to execute the subroutine EXIT. The mechanism for the simulation of the firing of the transition t_r can easily be extended to the case when there are more than three input arcs of t_r from the place p_k. Nonetheless, still only four scans of the tape k are sufficient in order to mark all the "enabling tokens" of t_r from p_k. Also, only up to three more scans are required in order to erase (remove) these "enabling tokens."

In case t_r has other input places than p_k, these operations are performed concurrently on the corresponding tapes.

If "enabling tokens" could be marked (and subsequently removed) for all input arcs of t_r, TM proceeds to deposit the "output tokens" of t_r. At this stage, the tapes k, $1 \leq k \leq |P|$, contain the

intermediate color marking UM^{j-}. For each place $p_k \in P$ for which $UM^{j-}(p_k) \neq \phi$, the head on the tape corresponding to p_k scans the rightmost nonempty tape square.

TM deposits the "output tokens" of the transition t_r following an algorithm similar to that used for removing the "enabling tokens" of t_r. Let p_m be an output place of t_r. TM determines first the positions on the tape m in which the corresponding "output tokens" of t_r have to be inserted by counting ¢ markers. Next, TM writes in the selected tape squares symbols corresponding to the first coordinates of the colors of the respective tokens. In this case, however, it is not necessary for TM to execute separate scans of the tape m during which the affected tape squares are marked and then additional scans during which the insertion of the respective "output tokens" is actually performed. These two operations can in fact be performed simultaneously. Thus, using the auxiliary tape $|P| + m$ as shown earlier (except that here symbols are added rather than eliminated) each "output token" can be inserted as soon as its position has been determined. In order, however, for the "output tokens" to be inserted in the correct "slots", "tokens" corresponding to arcs of t_r carrying output color functions of the form $[c_r^i; \ell.u.b. + s_r^i]$ must be deposited first. "Tokens" corresponding to output arcs carrying output color functions of the form $[c_r^i; g.\ell.b. + s_r^i]$ must be deposited second while "tokens" corresponding to output arcs carrying output color functions of the form $[c_r^i; s_r^i]$ must be deposited last. It can be verified that four scans of the tape m are sufficient in order to deposit all "output tokens" of t_r corresponding to the place p_m.

This completes the description of the Turing acceptor TM. It can be verified that the operations described above can be executed by TM deterministically. Moreover, in view of our earlier discussion regarding the amount of tape required by our encoding of the color bags corresponding to the places of ECP_Σ and of the fact that all firing sequences which generate the given input string w (if any) have the same length, $|w|$, it follows that every accepting computation of TM on w visits no more than $\alpha \cdot |w|$ tape squares on any tape, where α is a nonnegative integer constant. Hence

$$\Lambda(\text{EC-CPM}) \subseteq \text{DetTAPE}(\alpha \cdot i)$$

where $i(x) = x$ is the identity function on the set of real numbers. According to [BOOK-70a], for any real-valued function f of a single real variable and for any constant $k > 0$,

$$\text{DetTAPE}(k \cdot f) = \text{DetTAPE}(f).$$

Therefore, $\Lambda(\text{EC-CPM}) \subseteq \text{DetTAPE}(i)$. But DetTAPE(i) is the family of deterministic context-sensitive languages. Moreover, the language $\{a^n b^n c^n \mid n \geq 0\}$ is contained in $\mathscr{DCS}$ but not in Λ(EC-CPM). Thus, we actually have:

$$\Lambda(\text{EC-CPM}) \subset \mathscr{DCS}$$

Let us now consider the language family Λ_o(EC-CPM). Suppose UM^f is the final color marking of the Labelled EC-CPM ECP_Σ considered at the beginning of this appendix. For any string $w \in \Sigma^*$, $w \in \Lambda_o(ECP_\Sigma, UM^f)$ if and only if there exists a firing sequence $\gamma \in T(UM^o, UM^f)$ with $L(\gamma) = w$. Moreover, all firing sequences which generate w must have the same length, $|w|$. Since $T(UM^o, UM^f) \subset S(UM^o)$ the deterministic Turing acceptor TM can easily be modified in order to accept the language $\Lambda_o(ECP_\Sigma, UM^f)$. Thus, whenever TM finds a firing sequence γ with $L(\gamma) = w$ it checks in addition whether the color marking UM^m stored on the tapes k, $1 \leq k \leq |P|$, at the end of the simulation of the firing sequence γ equals UM^f. This check can be performed deterministically and certainly does not require any additional tape space. Hence, we can conclude that

$$\Lambda_o(\text{EC-CPM}) \subseteq \mathscr{DCS}$$

The Corollary 1 of Theorem 3.4.2 follows in a rather straightforward manner from the construct presented so far in this appendix. Without going into details, let us tranform the deterministic multi-tape Turing acceptor TM into a nondeterministic multi-tape Turing acceptor TM' which recognizes the same language as TM. For this purpose, we eliminate the tapes $2 \cdot |P| + 1$, $2 \cdot |P| + 2$ and $2 \cdot |P| + 3$. The tapes k and $|P| + k$, $1 \leq k \leq |P|$, are used the same way.

TM' incorporates a distinct subroutine for each transition of ECP_Σ. Each such subroutine simulates the firing of the corresponding transition. For any string $w \in \Sigma^*$, TM' checks whether $w \in \Lambda(ECP_\Sigma)$ ($w \in \Lambda_o(ECP_\Sigma, UM^f)$) by simulating firing sequences of length $|w|$ from

$S(UM^o)$ $(T(UM^o, UM^f))$. Here, the firing sequence simulated by TM' is determined by the input string w. Thus, TM' scans the input tape from left to right. For each symbol a read by TM' from its input tape, TM' enters the subroutine corresponding to some transition of ECP_Σ labelled a. During the execution of the respective subroutine, the head on the input tape is not advanced. In case there are several transitions of ECP_Σ carrying the same label a, the choice of the subroutine entered by TM' is nondeterministic.

We have seen that the simulation of the firing of a transition requires a fixed number of scans of the tapes k and $|P| + k$, $1 \leq k \leq |P|$. Since the number of nonempty tape squares on any of these tapes is bounded by $\alpha \cdot |w|$ and since all firing sequences which generate the input string w have the same length, $|w|$, it follows that every accepting computation of TM' on w requires no more than $\alpha' \cdot |w|^2$ steps, where α and α' are nonnegative integer constants. Note that in the case of the language $\Lambda_o(ECP_\Sigma, UM^f)$ after the simulation of any firing sequence γ, TM' must check whether γ leads to the given final color marking UM^f. This operation requires, however, only a fixed number of steps. Since for any real-valued function f of a single real variable and for any constant $k > 0$, $TIME(f) = TIME(k \cdot f)$ ([BOOK-70a]) it follows that:

$$\Lambda(\text{EC-CPM}) \subseteq TIME(x^2)$$

and

$$\Lambda_c(\text{EC-CPM}) \subseteq TIME(x^2)$$

APPENDIX C

MULTI-COUNTER E-ACCEPTORS

The objective of this appendix is to show that $\Lambda_o(\text{EPM(II)}) \subseteq \Lambda_o(\text{EPM})$ and to discuss the relationship between $\Lambda(\text{EPM(II)})$ and $\Lambda(\text{EPM})$. In order to do so we shall first introduce the family of multi-counter E-acceptors, a variant of the multi-counter acceptors ([BOOK-72]). The notations employed in this appendix are compatible with those of [BOOK-72].

Definition C.1

An n-counter E-acceptor is a system $A = (K, \Sigma, Z, \delta, q_o, F, n)$ where:

1. K is a finite set of states, $q_o \in K$ is the initial state and $F \subseteq K$ is the set of final states.

2. Σ is a finite set of symbols, called the input alphabet, Z is an abstract symbol called the storage symbol, n is a positive integer.

3. δ is a (possibly partial) mapping from $K \times (\Sigma \cup \{\lambda\}) \times \{Z, \lambda\}^{(n)}$ into the finite subsets of $K \times \{0, \lambda, Z, E\}^{(n)}$.

Definition C.2

A configuration of $A = (K, \Sigma, Z, \delta, q_o, F, n)$ is an n + 2-tuple $(q, w, y_1, \ldots, y_n)$ where $q \in K$ denotes the state of A, $w \in \Sigma^*$ denotes the input remaining to be read and for each k, $1 \leq k \leq n$, $y_k \in Z^*$ denotes the content of the k-th storage tape.

Let $\vdash$ denote the following relation among the configurations of A. For configurations $C_1 = (q, aw, y_1, \ldots, y_n)$ and $C_2 = (q', w, y_1', \ldots, y_n')$ where $a \in \Sigma \cup \{\lambda\}$, $C_1 \vdash C_2$ if there exists $(q', \sigma_1, \ldots, \sigma_n)$ in $\delta(q, a, Z_1, \ldots, Z_n)$ such that for each k, $1 \leq k \leq n$, one of the following conditions holds:

a) $Z_k = Z$, $y_k = x_k Z_k$, $\sigma_k = 0$ and $y_k' = y_k$
b) $Z_k = \lambda$, $y_k = \lambda$, $\sigma_k \in \{Z, \lambda\}$ and $y_k' = \sigma_k$
c) $Z_k = Z$, $y_k = x_k Z_k$, $\sigma_k = \lambda$ and $y_k' = x_k$
d) $Z_k = Z$, $y_k = x_k Z_k$, $\sigma_k = Z$ and $y_k' = y_k Z$

e) $Z_k = \lambda$, $y_k = \lambda$, $\sigma_k = E$ and $y_k' = \lambda$

f) $Z_k = Z$, $y_k = x_k Z_k$, $\sigma_k = E$ and $y_k' = \lambda$

Informally, a multi-counter E-acceptor A is a modified one-way, nondeterministic multi-counter acceptor. Each storage tape is a pushdown tape of the form Z^i (Z fixed). It is assumed that whenever $y_k \neq \lambda$, A scans the rightmost nonempty square of the k-th storage tape, for all k, $1 \leq k \leq n$. On the other hand, A also has the capability to sense if $y_k = \lambda$. In addition, however, a multi-counter E-acceptor can execute ganeralized erase moves (conditions e), f)) also called E-moves. From this perspective, a multi-counter accepter ([BOOK-72]) can be viewed as a multi-counter E-acceptor which does not perform E-moves.

Definition C.3

A computation of A of length m is a sequence of configurations $C_0, C_1, \ldots, C_m$ such that $C_i \vdash C_{i+1}$, $0 \leq i < m$. If A moves from some configuration C_i to a configuration C_j via a computation of length $m \geq 0$, we shall denote this fact by $C_i \vdash^* C_j$. In case we want to stress the length of the particular computation we shall use the notation $C_i \vdash^m C_j$.

Definition C.4

The language accepted by an n-counter E-acceptor $A = (K, \Sigma, Z, \delta, q_o, F, n)$ is the set:

$$L(A) = \{w \in \Sigma^* \mid (q_o, w, \{\lambda\}^{(n)}) \vdash^* (q, \lambda, \{\lambda\}^{(n)}) \text{ for some } q \in F\}$$

Definition C.5

Let d be a nonnegative integer. An n-counter E-acceptor $A = (K, \Sigma, Z, \delta, q_o, F, n)$ is said to operate with delay d or, equivalently, has delay d, if for all configurations $C_1 = (q, \lambda, y_1, \ldots, y_n)$ and $C_2 = (q', \lambda, y_1', \ldots, y_n')$ such that $C_1 \vdash^m C_2$, we must have $m \leq d$. If A has delay d for some $d \geq 0$, then A is said to be quasi-realtime.

The definition of a quasi-realtime multi-counter acceptor is analogous.

For each $n \geq 1$ let us define the language families:

$CE_n = \{L(A) \mid A \text{ is a quasi-realtime n-counter E-acceptor}\}$

$C_n = \{L(A) \mid A \text{ is a quasi-realtime n-counter acceptor}\}$

$$CE = \bigcup_{n=1}^{\infty} CE_n, \quad C = \bigcup_{n=1}^{\infty} C_n$$

We note that the definition of acceptance (Definition C.4) demands both final state and empty storage tapes. In Section 4 of [BOOK-72] it was shown that acceptance by final state alone of a quasi-realtime n-counter acceptor is equivalent to acceptance by final state and empty store. The arguments presented there can actually be extended in order to prove a more general result, namely that $CE = C$:

Lemma C.1

$CE_1 \subseteq C_1$.

Proof: Let $A = (K, \Sigma, Z, \delta, q_0, F, 1)$ be a quasi-realtime 1-counter E-acceptor. Let $A' = (K', \Sigma, Z, \delta', q_0, F, 1)$ be a 1-counter acceptor where:

1. $K' = K \cup K''$ where:

$K'' = \{[q, q'] \mid (q', E) \in \delta(q, a, y)$ for q, q' in K, $a \in \Sigma \cup \{\lambda\}$, $y \in \{\lambda, Z\}\}$

2. δ' is defined by:

2i. for all $q \in K$, $a \in \Sigma \cup \{\lambda\}$, $y \in \{\lambda, Z\}$

$\delta'(q, a, y) = \delta(q, a, y) - \{(q', E) \mid q' \in K\}$

2ii. if $(q', E) \in \delta(q, a, y)$ where $q' \in K$ then

$([q, q'], \lambda) \in \delta'(q, a, y)$

2iii. for all $[q, q'] \in K''$ and all $b \in \Sigma$

$\delta'([q, q'], \lambda, Z) = \{([q, q'], \lambda)\}$

$\delta'([q, q'], \lambda, \lambda) = \{(q', \lambda)\}$

$\delta'([q, q'], b, y) = \phi$.

Thus, A' simulates the moves of A until A performs an E-move. In that case A' enters an "erase loop" in which it erases its storage tape. As soon as the storage tape is empty, A' resumes the simulation of A.

Clearly, $L(A') = L(A)$ but A' may not be quasi-realtime. Nevertheless, any language accepted by a 1-counter acceptor is accepted by some quasi-realtime 1-counter acceptor ([GINS-74]). Thus, $L(A') \in C_1$.

□

Lemma C.2

If $L \in CE_n$, $n \geq 1$, then there exist $L_1, \ldots, L_n$ in CE_1, a λ-free renaming homomorphism h_2 and k-limited erasing homomorphism h_1 on $h_2(L_1 \cap \ldots \cap L_n)$ such that $L = h_1(h_2(L_1 \cap \ldots \cap L_n))$.

Proof: Let $A = (K, \Sigma, Z, \delta, q_o, F, n)$ be an n-counter E-acceptor which operates with delay k, for some $k \geq 0$. Let $A' = (K, \Sigma', Z, \delta', q_o, F, n)$ be an n-counter E-acceptor which has delay 0 and for which:

i. $\Sigma' = \Sigma \cup \{c\}$ where c is a new symbol, not in Σ.

ii. For all $q \in K$, $a \in \Sigma$ and $Z_j \in \{\lambda, Z\}$, $1 \leq j \leq n$

$$\delta'(q, a, Z_1, \ldots, Z_n) = \delta(q, a, Z_1, \ldots, Z_n)$$

and $$\delta'(q, c, Z_1, \ldots, Z_n) = \delta(q, \lambda, Z_1, \ldots, Z_n)$$

Let h_1 be a homomorphism on $(\Sigma \cup \{c\})^*$ defined by $h_1(c) = \lambda$ and $h_1(a) = a$, for all $a \in \Sigma$. Clearly, $h_1(L(A')) = L(A)$ and since A is quasi-realtime, h_1 is a k-limited erasing on $L(A')$.

Consider now an n-counter E-acceptor $A'' = (K'', \Sigma'', Z, \delta'', q_o'', F'', n)$ which operates with delay 0. We shall define n 1-counter E-acceptors A_k, $1 \leq k \leq n$, such that A_k simulates the k-th counter of A''. Let

$$\Sigma_2 = \{[q_1, a, Z_1, \ldots, Z_n, q_2, \sigma_1, \ldots, \sigma_n] \mid (q_2, \sigma_1, \ldots, \sigma_n) \in \delta(q_1, a, Z_1, \ldots, Z_n) \text{ where } q_1, q_2 \in K'', a \in \Sigma'', Z_i \in \{\lambda, Z\}, \sigma_i \in \{0, \lambda, Z, E\}, 1 \leq i \leq n\}$$

Then, $A_k = (K'', \Sigma_2, Z, \delta_k, q_o'', F'', 1)$ where δ_k is defined by:

i. $(q_2, \sigma_k) \in \delta_k(q_1, [q_1, a, Z_1, \ldots, Z_n, q_2, \sigma_1, \ldots, \sigma_n], Z_k)$

for all $[q_1, a, Z_1, \ldots, Z_n, q_2, \sigma_1, \ldots, \sigma_n]$ in Σ_2.

Let h_2 be the homomorphism on Σ_2^* defined by:

$$h_2([q_1, a, Z_1, \ldots, Z_n, q_2, \sigma_1, \ldots, \sigma_n]) = a$$

for all $[q_1, a, Z_1, \ldots, Z_n, q_2, \sigma_1, \ldots, \sigma_n] \in \Sigma_2$. Evidently, h_2 is a λ-free renaming homomorphism. It is easy to see that $L(A'') = h_2(L_1 \cap \ldots \cap L_n)$ where $L_k = L(A_k)$, $1 \leq k \leq n$.

□

Lemma C.3

$CE = C$.

Proof: For any $L \in CE_n$, $n \geq 1$, we have $L = h_1(h_2(L_1 \cap \ldots \cap L_n))$ where h_1, h_2 and L_k, $1 \leq k \leq n$ are as in Lemma C.2. By Lemma C.1,

$L_k \in C_1$, $1 \leq k \leq n$. On the other hand, $C_n = H(C_1 \wedge \ldots \wedge C_1)$ ∓ where C_1 occurs n times in the last expression ([BOOK-72]). Consequently, $h_2(L_1 \cap \ldots \cap L_n) \in C_n$. Also, C_n is a principal AFL ([BOOK-72]). By Theorem 2.1 of [GINS-69], C_n is closed under k-limited erasing. Thus, $h_1(h_2(L_1 \cap \ldots \cap L_n) \in C_n$ and therefore $CE_n \subseteq C_n$.

On the other hand, n-counter acceptors are a particular case of the n-counter E-acceptors and therefore $C_n \subseteq CE_n$.

□

We note that the family of languages realtime recognizable by deterministic multi-counter acceptors with generalized erase moves properly includes the class of languages accepted in realtime by deterministic multi-counter acceptors without E-moves ([FISH-68]). For the nondeterministic case, quasi-realtime acceptance is equivalent to (nondeterministic) realtime acceptance ([BOOK-72]). In light of Lemma C.3, E-moves do not increase the recognition power of nondeterministic multi-counter acceptors.

Lemma C.4

$\Lambda_o(EPM(II)) \subseteq CE$.

Proof: Let $EN_\Sigma = (EN, \Sigma, L)$ be a Labelled EPM(II) where $EN = (EN, M^o)$ and EN = (T, P, I, O). Let M^f denote the final marking of EN_Σ. It is rather straightforward to show that $\Lambda_o(EN_\Sigma, M^f) \in CE$. We shall construct a quasi-realtime n-counter E-acceptor $A = (K, \Sigma, Z, \delta, q_o, F, n)$ where $n = |P|$, which simulates firing sequences of EN_Σ. A contains a distinct storage tape for each place in P; tape j, $1 \leq j \leq |P|$, will store the markings of $p_j \in P$ along any firing sequence of EN_Σ.

Given an input string $w \in \Sigma^*$ A will check whether there exists a firing sequence $\gamma \in T(M^o, M^f)$ such that $L(\gamma) = w$. The simulation of a firing sequence requires three phases.

Phase I Let $m_o = \max\{M^o(p_j) \mid p_j \in P\}$. Phase I requires $m_o + 1$ states, denoted by $q_o, q_1, \ldots, q_{m_o-1}, q_{m_o} = q_p$. For all i, $1 \leq i \leq m_o$,

---

∓ For each family $\mathcal{A}$ of languages $H(\mathcal{A}) = \{h(L) \mid L \in \mathcal{A}$, h is a λ-free homomorphism$\}$. For all families $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of languages $\mathcal{A}_1 \wedge \ldots \wedge \mathcal{A}_n = \{L_1 \cap \ldots \cap L_n \mid L_i \in \mathcal{A}_i, 1 \leq i \leq n\}$.

$\delta(q_{i-1}, \lambda, Z_{i-1\,1}, \cdots, Z_{i-1\,n}) = \{(q_i, \sigma_{i1}, \cdots, \sigma_{in})\}$
where for each r, $1 \leq r \leq n$, $Z_{or} = \lambda$,

$$Z_{ir} = \begin{cases} Z & \text{if } M^\circ(p_r) > 0 \\ \lambda & \text{if } M^\circ(p_r) = 0 \end{cases}$$

and

$$\sigma_{ir} = \begin{cases} Z & \text{if } i \leq M^\circ(p_r) \text{ and } M^\circ(p_r) > 0 \\ 0 & \text{if } i > M^\circ(p_r) \text{ and } M^\circ(p_r) > 0 \\ \lambda & \text{if } M^\circ(p_r) = 0 \end{cases}$$

The function δ is not defined for tuples from the set $\{q_o, q_1, \cdots, q_{m_o-1}\} \times \Sigma \times \{\lambda, Z\}^{(n)}$ (A does not accept any inputs from Σ in Phase I). Thus, in Phase I A performs the computation $(q_o, w, \{\lambda\}^{(n)}) \vdash^{m_o} (q_p, w, y_{o1}, \cdots, y_{on})$ where $y_{or} = Z^{M^\circ(p_r)}$, $1 \leq r \leq n$. Evidently, this phase is completely deterministic and requires exactly m_o λ-input moves. At the end of Phase I A is in a distinguished state, q_p.

Phase II During Phase II, A performs the actual simulation of a firing sequence. For each transition $t_k \in T$, the control of A contains a separate subroutine. Suppose A is currently in some configuration $(q_p, aw', y_1, \cdots, y_n)$ where $a \in \Sigma$ and $y_r \in Z^*$, $1 \leq r \leq n$. At this stage A can enter (nondeterministically) the subroutine corresponding to either transition t_k such that $L(t_k) = a$. Let t_s be such a transition. Let $m_s = \max\{I(p_j, t_s) \mid p_j \in P_s^n\}$ and $n_s = \max\{O(t_s, p_j) \mid p_j \in P - P_s^a\}$. The subroutine corresponding to t_s requires exactly $m_s + n_s$ states, denoted by $q_1^i, \cdots, q_{m_s}^i, q_1^o, \cdots, q_{n_s}^o$. Let $q_{m_s+1}^i = q_1^o$.

First, A checks whether "t_s is enabled". Thus

$$(q_1^i, \sigma_1, \cdots, \sigma_n) \in \delta(q_p, a, Z_1, \cdots, Z_n)$$

where for each r, $1 \leq r \leq n$, $Z_r = Z$ and $\sigma_r = 0$ if $p_r \in P_s^n$, $Z_r = \lambda$ and $\sigma_r = \lambda$ if $p_r \in P_s^z$, $Z_r \in \{\lambda, Z\}$, $\sigma_r = 0$ (if $Z_r = Z$) and $\sigma_r = \lambda$ (if $Z_r = \lambda$) if $p_r \notin \mathscr{D}(I_s)$.

For all j, $1 \leq j \leq m_s$,

$$\delta(q_j^i, Z_{j1}, \cdots, Z_{jn}) = \{(q_{j+1}^i, \sigma_{j1}, \cdots, \sigma_{jn})\}$$

where for each r, $1 \leq r \leq n$, $Z_{jr} = Z$ and $\sigma_{jr} = \lambda$ if $I(p_r, t_s) \geq j$, $Z_{jr} \in \{Z, \lambda\}$, $\sigma_{jr} = 0$ (if $Z_{jr} = Z$) and $\sigma_{jr} = \lambda$ (if $Z_{jr} = \lambda$) if $I(p_r, t_s) < j$ or $p_r \notin \mathscr{D}(I_s)$, $Z_{jr} = \lambda$ and $\sigma_{jr} = \lambda$ if $I(p_r, t_s) = \zeta$.

Also, for all j, $1 \leq j \leq n_s - 1$,

$$\delta(q_j^o, \lambda, Z_{j1}, \ldots, Z_{jn}) = \{(q_{j+1}^o, \sigma_{j1}, \ldots, \sigma_{jn})\}$$

where for each r, $1 \leq r \leq n$, $Z_{jr} \in \{\lambda, Z\}$, $\sigma_{jr} = Z$ if $O(t_s, p_r) \geq j$, $\sigma_{jr} = 0$ (if $Z_{jr} = 0$) and $\sigma_{jr} = \lambda$ (if $Z_{jr} = \lambda$) if $O(t_s, p_r) < j$ or $p_r \in P_s^a$.

The last move of the subroutine is:

$$\delta(q_{n_s}^o, \lambda, Z_{j1}, \ldots, Z_{jn}) = \{(q_p, \sigma_1, \ldots, \sigma_n)\}$$

such that for each r, $1 \leq r \leq n$, $Z_{jr} \in \{\lambda, Z\}$, $\sigma_{jr} = E$ if $p_r \in P_s^a$, $\sigma_{jr} = Z$ if $O(t_s, p_r) = n_s$ and $\sigma_{jr} = 0$ (if $Z_{jr} = Z$) or $\sigma_{jr} = \lambda$ (if $Z_{jr} = \lambda$) if $O(t_s, p_r) < n_s$.

A does not accept in the subroutine any inputs other than λ. Thus, the subroutine attempts to perform the computation

$(q_p, aw', y_1, \ldots, y_n) \vdash^{m_s} (q_1^o, w', y_1', \ldots, y_n') \vdash^{n_s} (q_p, w', y_1'', \ldots, y_n'')$ where for each r, $1 \leq r \leq n$, $y_r' Z^{I(p_r, t_s)} = y_r$ if $p_r \in P_s^n$ or $y_r' = y_r$ if $p_r \notin P_s^n$ and $y_r'' = y_r' Z^{O(t_s, p_r)}$ if $p_r \notin P_s^a$ or $y_r'' = \lambda$ if $p_r \in P_s^a$.

The subroutine is executed deterministically and requires exactly $m_s + n_s$ λ-input moves.

After the entire input string w is read, A reaches a configuration of the form $(q_p, \lambda, y_{f1}, \ldots, y_{fn})$ and enters Phase III.

Phase III Let $m_f = \max\{M^f(p_j) \mid p_j \in P\}$. Phase III requires $m_f + 1$ states, denoted by $q_p = q_o^f, q_1^f, \ldots, q_{m_f-1}^f, q_{m_f}^f = q_f$. Let $F = \{q_f\}$. For all i, $1 \leq i \leq m_f$,

$$\delta(q_{i-1}^f, \lambda, Z_{i-1\,1}, \ldots, Z_{i-1\,n}) = \{(q_i^f, \sigma_{i1}, \ldots, \sigma_{in})\}$$

where for each r, $1 \leq r \leq n$, $\sigma_{ir} = \lambda$ and

$$Z_{i-1\,r} = \begin{cases} \lambda & \text{if } M^f(p_r) < i \\ Z & \text{if } M^f(p_r) \geq i \end{cases}$$

In this phase, A does not accept any input symbols from Σ. Thus, A attempts to perform deterministically the computation $(q_p, \lambda, y_{f1}, \ldots, y_{fn}) \vdash^{m_f} (q_f, \lambda, \{\lambda\}^{(n)})$. Evidently, the configuration $(q_f, \lambda, \{\lambda\}^{(n)})$ can be reached only if $y_{fr} = Z^{M^f(p_r)}$, $1 \leq r \leq n$. Exactly m_f λ-input moves are required for this purpose.

Let $m = \max\{m_s + n_s \mid t_s \in T\}$ and $\mu = \max\{m, m_o, m_f\}$. Evidently,

$w \in \Sigma^*$ is accepted by A if and only if there exists a firing sequence $\gamma \in T(M^o, M^f)$ such that $L(\gamma) = w$. Moreover, A operates with delay μ. □

We shall now present a refinement of the result given in Theorem 9.2 of [HACK-75]:

Lemma C.5

$\Lambda_o(EPM) = \{W \mid W \in C \text{ and } \lambda \notin W\}$

Proof: By Theorem 3.1 of [BOOK-72], if $W \in C$ then there exists an n-counter acceptor A, which operates with delay 0, such that $W = L(A)$. Without loss in generality, let us then consider such an n-counter acceptor $A = (K, \Sigma, Z, \delta, q_o, F, n)$ and suppose $\lambda \notin L(A)$. We shall build a Labelled EPM(I) $EN_\Sigma = (EN, \Sigma, L)$ such that $\Lambda_o(EN_\Sigma, M^f) = L(A)$ where M^f denotes the final marking of EN_Σ. Let $EN = (EN, M^o)$ where $EN = (T, P, I, O)$ is the underlying EPN(I). EN_Σ simulates computations of A.

We shall introduce a distinct place in P for each of the n storage tapes of A; place $p_j \in P$ "remembers" the contents of tape j, $1 \leq j \leq n$, along any computation of A. In addition, we shall introduce in P a place denoted by π, whose markings encode the current state of A. Suppose we have ordered the states in K such that each state q_s can be uniquely represented by an integer tag $|q_s|$, where $1 \leq |q_s| \leq |K|$. Thus, the markings of π, i.e. $M^i(\pi)$, are always bounded through our construct by: $1 \leq M^i(\pi) \leq |K|$.

Let $(q_s, a, \alpha) \in K \times \Sigma \times \{\lambda, Z\}^{(n)}$ be a tuple such that $\delta(q_s, a, \alpha)$ is defined. Let $\alpha = (Z_1, \ldots, Z_n)$. Suppose $(q_m, \beta) \in \delta(q_s, a, \alpha)$ and let $\beta = (\sigma_1, \ldots, \sigma_n)$. We shall introduce in T a transition t_k such that:

i. $I(\pi, t_k) = (\sigma, |q_s|)$; $O(t_k, \pi) = |q_m|$

ii. For each r, $1 \leq r \leq n$,

$$I(p_r, t_k) = \begin{cases} 1 & \text{if } Z_r = Z \\ \zeta & \text{if } Z_r = \lambda \end{cases}$$

$$O(t_k, p_r) = \begin{cases} 2 & \text{if } \sigma_r = Z \text{ and } Z_r = Z \\ 1 & \text{if } \sigma_r = 0 \text{ or } \sigma_r = Z \text{ and } Z_r = \lambda \\ 0 & \text{if } \sigma_r = \lambda. \end{cases}$$

Let also $L(t_k) = a$. We shall introduce such a transition for all tuples $(q_m, \beta) \in \delta(q_s, a, \alpha)$, for all tuples (q_s, a, α) for which the mapping δ is defined. Evidently, each transition simulates one possible move of A and at the same time generates the symbol $a \in \Sigma$ which is read in by A during that move (recall that A must advance its input head at each move, i.e. λ-input is not permitted). The initial marking M° is defined by $M^\circ(\pi) = |q_o|$ and $M^\circ(p_r) = 0$ for all $p_r \in P$, $p_r \neq \pi$.

If $w \in L(A)$, then there exists a computation of A $(q_o, w, \{\lambda\}^{(n)}) \vdash^* (q_f, \lambda, \{\lambda\}^{(n)})$, for some $q_f \in F$. In EN_Σ, the firing sequence which simulates this computation leads to a marking M where $M(\pi) = |q_f|$ and $M(p_r) = 0$ for all $p_r \in P$, $p_r \neq \pi$. Since A may have several final states, firing sequences which simulate distinct accepting computations of A may lead to distinct markings. In order to secure a unique final marking M^f, we shall introduce a set of "stop" transitions. Thus, let $q_f \in F$ and suppose t_n is a transition such that $O(t_n, \pi) = |q_f|$, i.e., t_n has been introduced to simulate a move of A which leads to q_f. We shall introduce a transition t_{nstop} in T which has exactly the same input and output connections as t_n except for π; we shall set $O(t_{nstop}, \pi) = 0$. We note that the firing of t_{nstop} in any marking leaves the place π empty and therefore disables all the transitions of *EN* (all transitions of *EN*, including t_{nstop}, are connected to π by positive testing arcs). Moreover, the firing of t_{nstop} in some marking leads to the zero marking (all places of P are empty) if and only if the firing of t_n in the same marking leads to a marking in which π contains $|q_f|$ tokens and the remaining places of P are empty (which corresponds to the last move of an accepting computation). Let also $L(t_{nstop}) = L(t_n)$.

This construct is applied for all transitions of T such as t_n, for all $q \in F$. Finally, we shall set M^f to be the zero marking. Then, $\Lambda_o(EN_\Sigma, M^f) = L(A)$.

Since the multi-counter acceptors are similar to the multi-counter E-acceptors in every respect except for the E-moves, the construct of Lemma C.4 can immediately be particularized in order to show that if $W \in \Lambda_o(EPM)$ then there exists a quasi-realtime n-counter acceptor A such that $W = L(A)$. Since for any such language W, $\lambda \notin W$ by definition, the theorem follows.

If $W \in \Lambda_o(\text{EPM(II)})$ then by definition, $\lambda \notin W$. By Lemma C.4, we actually have $\Lambda_o(\text{EPM(II)}) \subseteq \{W \mid W \in CE \text{ and } \lambda \notin W\}$. But by Lemma C.3, $\{W \mid W \in CE \text{ and } \lambda \notin W\} = \{W \mid W \in C \text{ and } \lambda \notin W\}$. Hence, by Lemma C.5, $\Lambda_o(\text{EPM(II)}) \subseteq \Lambda_o(\text{EPM})$.

Let us now examine the relationship between the language families $\Lambda(\text{EPM(II)})$ and $\Lambda(\text{EPM})$. For each n-counter acceptor $A = (K, \Sigma, Z, \delta, q_o, F, n)$ let

$$L_f(A) = \{w \in \Sigma^* \mid (q_o, w, \{\lambda\}^{(n)}) \vdash^* (q, \lambda, y_1, \ldots, y_n) \text{ for } q \in F, y_r \in Z^*, 1 \leq r \leq n\}.$$

Suppose A is an n-counter acceptor with delay 0. A sufficient condition for $L_f(A)$ to be in $\Lambda(\text{EPM})$ is that $K = F$, i.e. each state of A is a final state. In this case, for any $w \in \Sigma^*$, $w \in L_f(A)$ if and only if there exists a computation of A on w, that is if and only if $(q_o, w, \{\lambda\}^n) \vdash^* (q, \lambda, y_1, \ldots, y_n)$. Equivalently, each computation of A is an accepting computation in the sense of $L_f(A)$. Note that if $w \in L_f(A)$, then all prefixes of w, including λ, are in $L_f(A)$ as well (q_o is also a final state).

Let A be such an n-counter acceptor and let us apply to A the construct of Lemma C.5 while ignoring the final marking and the pertinent set of "stop" transitions. Let $EN_\Sigma = (EN, \Sigma, L)$ be the resulting Labelled EPM(I). A straightforward induction argument can show that for any $w \in \Sigma^*$ there exists a computation of A on w if and only if there exists a firing sequence $\gamma \in S(M^\circ)$ of EN_Σ such that $L(\gamma) = w$. Hence, $w \in L_f(A)$ if and only if $w \in \Lambda(EN_\Sigma)$ and thus $L_f(A) \in \Lambda(\text{EPM})$.

Let us now consider a language $W \in \Lambda(\text{EPM(II)})$. By Corollary 1 of Theorem 4.7.2, $W - \{\lambda\} \in \Lambda_o(\text{EPM})$. By Lemma C.5, $W \in C$. By Theorem 3.1 of [BOOK-72] there exists an n-counter acceptor A, with delay 0, such that $L(A) = W$. Evidently, there exists an n-counter acceptor A', with delay 0, such that $L_f(A') = L(A)$ (A' simulates A but enters an accepting state if and only if A enters an accepting state with all counters empty). Let us consider the computation tree of A' with input w. Each path in the tree corresponds to a computation of A' with input w. Each node corresponds to a configuration of A' along one such computation. Let us organize the nodes in levels, that is level k, $0 \leq k \leq |w|$, contains the set of configurations which can be reached by A' after reading in the k leftmost symbols of w.

We note that if $w \in W$ then all prefixes of w must be in W. Consequently, in the computation tree of w each level must contain at least one accepting configuration. Conversely, if there exists a level such that no node on that level corresponds to an accepting configuration then $w \notin W$. Unfortunately this fact does not imply that all states of A' are final states or that there exists an n-counter acceptor with delay 0 which has this property.

APPENDIX D

THE PETRI NET MODEL WITH SWITCHES DISJUNCTIVE LOGIC AND TOKEN ABSORBERS

This appendix defines the Petri Net Model with switches, disjunctive logic and token absorbers, an extension of the model described in [BAER-73].

Definition D.1

A Petri Net with switches, disjunctive logic and token absorbers is a system SPN = (SN, Log^+, Log^-) where:

1. SN = (T, P, I, O) is a modified Generalized Petri Net, where:

1a. T is a finite set of transitions.

1b. P is a finite set of places. Let $S \subset P$ be the set of switches, a distinguished category of places.

1c. I: $P \times T \rightarrow Z^\circ$ is the input incidence function.

1d. O: $T \times P \rightarrow Z^\circ \cup \{e, f, \xi\}$ is the output incidence function.

Following restrictions are imposed on the input and output incidence functions. Let t_k be an arbitrary transition in T:

i. If there exists a switch $s_j \in S$ such that $I(s_j, t_k) > 0$ then $I(s_r, t_k) = 0$ for all $s_r \in S$, $s_r \neq s_j$. Moreover, $I(s_j, t_k) = 1$. Thus, a transition can have at most one switch as input place, in which case the switch is connected to the respective transition by a single input arc. In order to distinguish the switches from the "regular" places, switches are represented graphically as triangles, rather than circles (in Figure D.1, for example, s_j denotes the input switch of t_k).

ii. If t_k has a switch as input place then there exist $p_r \in P$ and $p_s \in P$ such that $O(t_k, p_r) = e$ and $O(t_k, p_s) = f$. The transition t_k is connected to p_r and p_s by special directed arcs, labelled e and f, respectively (see Figure D.1). In addition, for all $p_q \in P$, $p_q \neq p_r$ and $p_q \neq p_s$, either $O(t_k, p_q) = 0$ or $O(t_k, p_q) = \xi$.

If $O(t_k, p_q) = \xi$, for some pair $(t_k, p_q) \in T \times P$, then t_k and p_q are connected by a special type of arc, called a token absorber. A token absorber is represented graphically as indicated in Section 4.7.

2. Log^{+}: T → $\mathcal{P}$(P - S) is the input logic function. Here $\mathcal{P}$(P - S) denotes the power set of the set of "regular" places. For each transition $t_k \in T$, $Log^{+}(t_k) = \omega$ where $\omega \subseteq \mathcal{D}(I_k)$.

3. Log^{-}: T → $\mathcal{P}$(P) is the output logic function. For each transition $t_k \in T$, $Log^{-}(t_k) = \omega$ where

$$\omega \subseteq \{p_r \mid O(t_k, p_r) = s, s \in Z^{+}\}.$$

A marking of a Petri Net with switches, disjunctive logic and token absorbers is a total, single-valued mapping from the set of places P into Z°.

Definition D.2

A transition $t_k \in T$ is enabled in the marking M^i if and only if:

1. For all places p_r such that $I(p_r, t_k) > 0$, $p_r \notin S$ and $p_r \notin Log^{+}(t_k)$, $M^i(p_r) \geq I(p_r, t_k)$.

2. If $Log^{+}(t_k) \neq \phi$, there exists a place $p_s \in Log^{+}(t_k)$ such that $M^i(p_s) \geq I(p_s, t_k)$ and $M^i(p_q) = 0$, for all $p_q \in Log^{+}(t_k)$, $p_q \neq p_s$.

Condition 1 above is also referred to as conjunctive, or AND, input logic while condition 2 is referred to as disjunctive, or EOR, input logic. Note that if t_k has a switch as input place, the marking of the

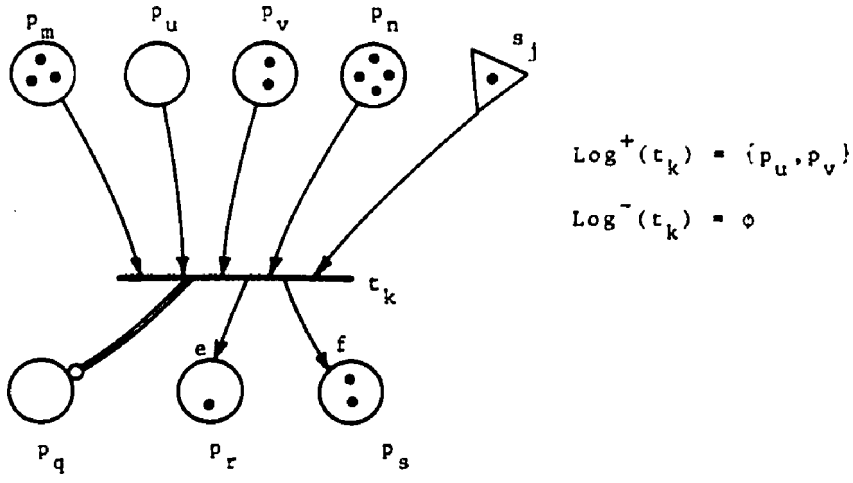


Figure D.1
Sample Transition

respective switch does not influence the enabled status of t_k. For example, in Figure D.2 transition t_m is enabled while transition t_n is disabled (because condition 2 is not satisfied).

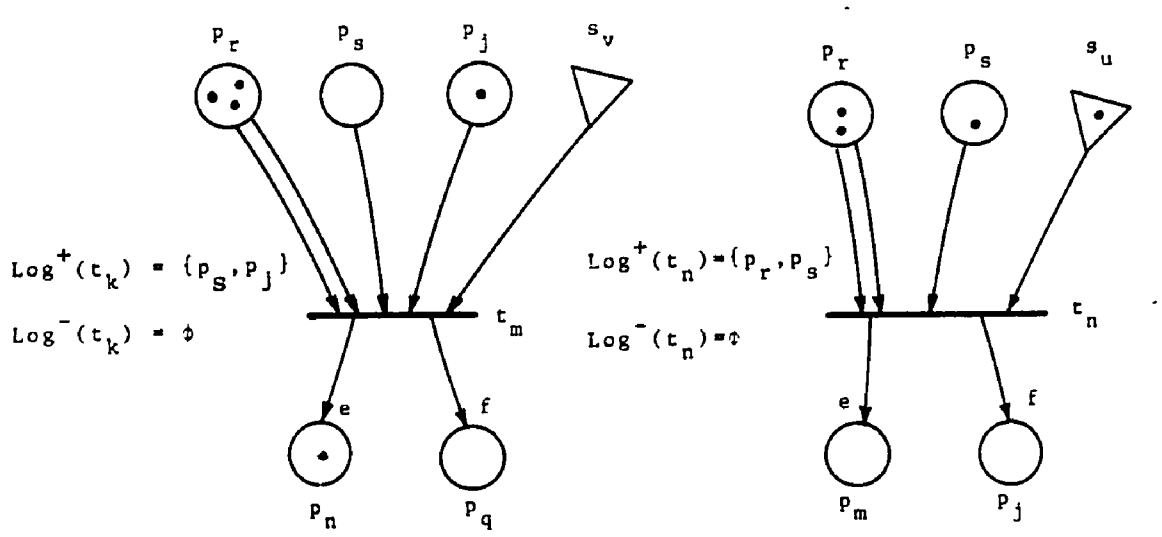


Figure D.2
Sample Transitions

Any transition enabled in some marking M^i may be selected to fire in M^i. Suppose t_k is enabled in M^i and let M^{i+1} denote the marking obtained after firing t_k in M^i. Let M^{i-} denote the "intermediate marking" obtained after collecting the corresponding tokens from the input places of t_k but before tokens are placed in the output places of t_k. Thus:

i. for all places $p_j \in P$ such that $I(p_j, t_k) = 0$, $M^{i-}(p_j) = M^i(p_j)$.

ii. for all places p_r such that $I(p_r, t_k) > 0$, $p_r \notin S$ and $p_r \notin Log^+(t_k)$, $M^{i-}(p_r) = M^i(p_r) - I(p_r, t_k)$.

iii. for all places $p_s \in Log^+(t_k)$, $M^{i-}(p_s) = 0$ if $M^i(p_s) = 0$ or $M^{i-}(p_s) = M^i(p_s) - I(p_s, t_k)$ if $M^i(p_s) > 0$ (recall, however, that all places in $Log^+(t_k)$ but one must have a zero marking in order for t_k to be enabled in M^i).

iv. if s_u is the input switch of t_k then

$$M^{i-}(s_u) = \begin{cases} 0 & \text{if } M^i(s_u) = 0 \\ M^i(s_u)-1 & \text{if } M^i(s_u) > 0 \end{cases}$$

We can now define the marking M^{i+1} as follows:

i. for all places $p_j \in P$ such that $O(t_k, p_j) = 0$, $M^{i+1}(p_j) = M^{i-}(p_j)$.

ii. for all $p_r \in P$ such that $O(t_k, p_r) = \xi$, $M^{i+1}(p_r) = 0$.

iii. if t_k has a switch as input place, let it be s_j, then let $p_m \in P$ be such that $O(t_k, p_m) = e$ and let $p_n \in P$ be such that $O(t_k, p_n) = f$. Then

$$M^{i+1}(p_m) = \begin{cases} M^{i-}(p_m) & \text{if } M^i(s_j) > 0 \\ M^{i-}(p_m) + 1 & \text{if } M^i(s_j) = 0 \end{cases}$$

$$M^{i+1}(p_n) = \begin{cases} M^{i-}(p_n) + 1 & \text{if } M^i(s_j) > 0 \\ M^{i-}(p_n) & \text{if } M^i(s_j) = 0 \end{cases}$$

iv. if t_k does not have a switch as input place then for all places p_q such that $O(t_k, p_q) > 0$ and $p_q \notin Log^-(t_k)$, $M^{i+1}(p_q) = M^{i-}(p_q) + O(t_k, p_q)$. If $Log^-(t_k) \neq \phi$, for some place $p_s \in Log^-(t_k)$, $M^{i+1}(p_s) = M^{i-}(p_s) + O(t_k, p_s)$ while for all p_u, $p_u \in Log^-(t_k)$ and $p_u \neq p_s$, $M^{i+1}(p_u) = M^{i-}(p_u)$. The place p_s is arbitrarily selected.

Notice that if t_k has the switch s_j as input place then p_m receives a token upon the firing of t_k in M^i only if $M^i(s_j) = 0$, i.e. only if s_j is "empty" in the marking M^i (whence the label e of the arc connecting t_k to p_m). On the other hand, if $M^i(s_j) > 0$, i.e. s_j is "full" in the marking M^i, then p_n receives a token upon the firing of t_k in M^i (whence the label f of the arc connecting t_k to p_n). Thus, the place s_j acts indeed as a "switch" directing the output token of t_k to p_m or p_n (but not both) depending on its marking prior to the firing of t_k. Figure D.3 examplifies the operation of a switch.

Figure D.4 examplifies the firing of a transition which does not have a switch as input place and for which $Log^+(t_k)$ and $Log^-(t_k)$ are not empty. We emphasize that upon the firing of t_k only one place from $Log^-(t_k)$ receives tokens, the selection of that place being arbitrary. This constraint is referred to as disjunctive, or EOR, output logic.

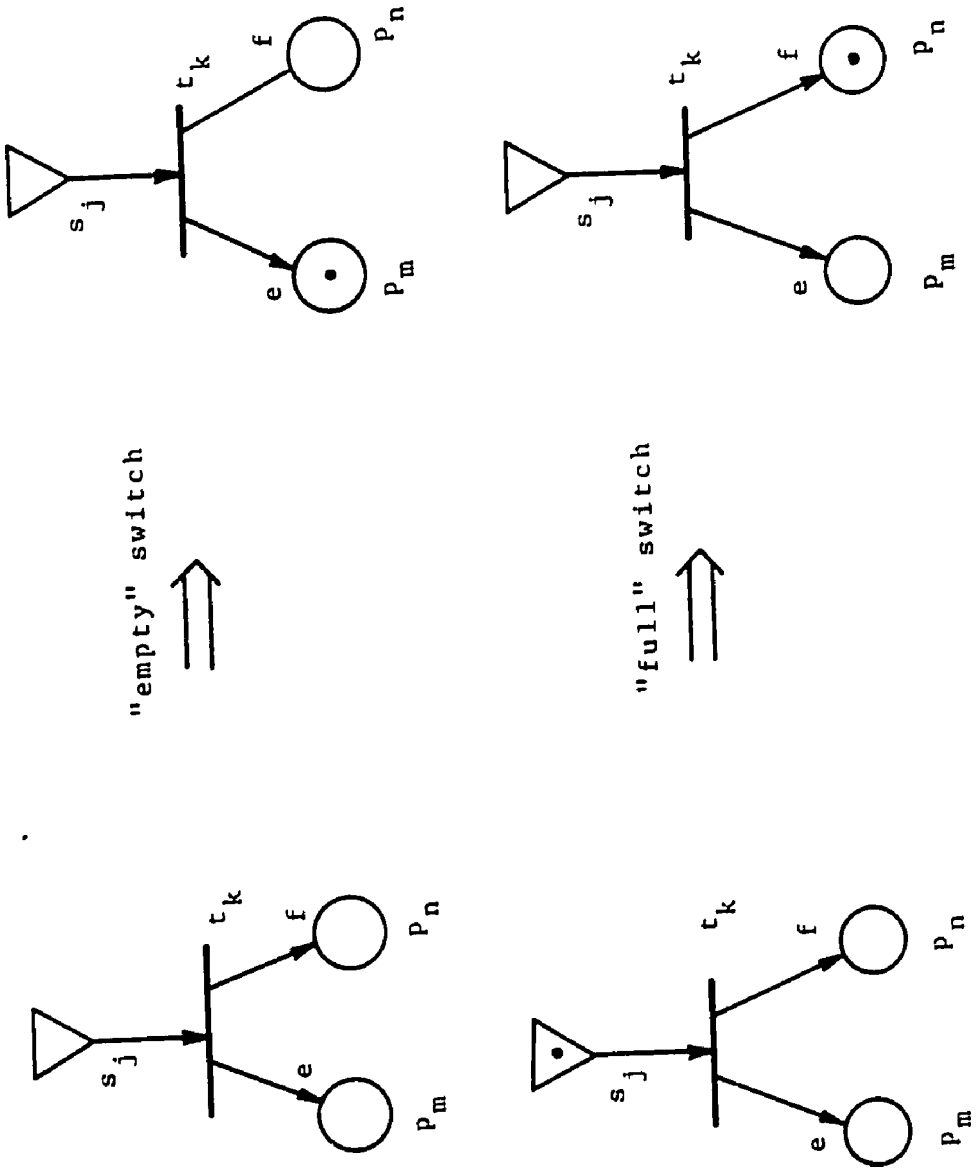


Figure D.3
Operation of a Switch

Figure D.4 also examplifies the operation of a token absorber. Note that a token absorber performs exactly as described in Section 4.7. Thus, if $O(t_k, p_r) = \xi$, for some place $p_r \in P$, then upon the firing of t_k in M^i all tokens present in p_r are "absorbed" such that $M^{i+1}(p_r) = 0$, irrespective of $M^i(p_r)$.

Definition D.3

A Petri Net Model with switches, disjunctive logic and token absorbers (SPM) is a system: $SPM = (SPN, M^\circ)$ where:

1. $SPN = (SN, Log^+, Log^-)$ is a Petri Net with switches, disjunctive logic and token absorbers.
2. M° is the initial marking of SN.

Definition D.4

A Labelled Petri Net Model with switches, disjunctive logic and token absorbers is a system $SPM_\Sigma = (SPM, \Sigma, L)$ where:

1. $SPM = (SPN, M^\circ)$ is a Petri Net Model with switches, disjunctive logic and token absorbers.
2. Σ is a finite label alphabet.
3. $L: T \to \Sigma$ is a total single-value mapping, called the labelling function of SPM_Σ.

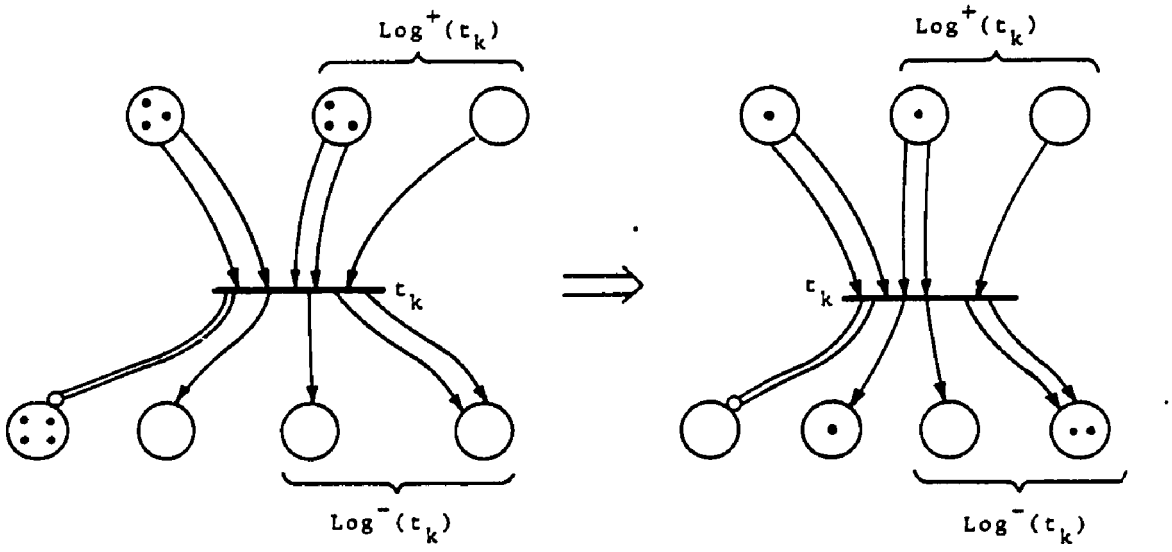


Figure D.4
The Firing of a Transition without a Switch as Input Place

The families of Languages Λ(SPM) and Λ_o(SPM) are defined analogous to Λ(C-CPM) and Λ_o(C-CPM), respectively.

BIBLIOGRAPHY

[AGAR-75] Agarwala, T.K.M., "Towards a Theory for the Analysis and Synthesis of Systems Exhibiting Concurrency," Ph.D. Thesis, The Johns Hopkins University, Baltimore, Maryland, 1975.

[AHO-72] Aho, A.V. and Ullman, J.D., "The Theory of Parsing, Translation and Compiling," Vol. I, Prentice-Hall, Series in Automatic Computation, 1972.

[BAER-73] Baer, J.L., "Modeling for Parallel Computation: A Case Study," Sagamore Computer Conference on Parallel Processing, 1973.

[BIRK-67] Birkhoff, G., "Lattice Theory," American Mathematical Society Colloquium Publications, Vol. XXV, 1967.

[BOOK-70a] Book, R.V., Greibach, S.A., and Wegbreit, B., "Time- and Tape-Bounded Turing Acceptors and AFL's," Journal of Computer and System Sciences: 4, 1970.

[BOOK-70b] Book, R.V. and Greibach, S.A., "Quasi-Realtime Languages," Mathematical Systems Theory, Vol. 4, No. 2, 1970.

[BOOK-72] Book, R.V. and Ginsburg, S., "Multi-Stack-Counter Languages," Mathematical Systems Theory, Vol. 6, No. 1, 1972.

[CERF-72] Cerf, V.G., "Multiprocessors, Semaphores, and a Graph Model of Computation," Ph.D. Thesis, Computer Science Department, UCLA, 1972 (UCLA-10P14-110).

[COUR-71] Courtois, P.J., Heymans, F., and Parnas, D.L., "Concurrent Control with "Readers" and "Writers"," CACM, Vol. 14, No. 10, October 1971.

[DENN-70] Dennis, J.B., "Modular, Asynchronous Control Structures for a High Performance Processor," Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, Woods Hole, Massachusetts, June 2-5, 1970.

[DENN-75] Dennis, J.B., "First Version of a Data Flow Procedure Language," MAC Technical Memorandum 61, Project MAC, MIT, Cambridge, Massachusetts, May 1975.

[FISH-68] Fisher, P.C., Meyer, A.R., and Rosenberg, A.L., "Counter Machines and Counter Languages," Mathematical Systems Theory, Vol. 2, No. 3, 1968.

[GINS-69] Ginsburg, S. and Greibach, S.A., "Abstract Families of Languages," Memoires Americ. Math. Soc., No. 87, 1969.

[GINS-74] Ginsburg, S. and Rose, G.F., "The Equivalence of Stack-Counter Acceptors and Quasi-Realtime Stack-Counter Acceptors," Journal of Computer and System Sciences, 8, 1974.

[GOST-71] Gostelow, K.P., "Flow of Control, Resource Allocation, and the Proper Termination of Programs," Ph.D. Thesis, Computer Science Department, UCLA 1971 (UCLA ENG-7179/UCLA-10P14-106).

[GREI-69] Greibach, S.A. and Hopcroft, J.E., "Scattered Context Grammars," Journal of Computer and System Sciences, Vol. 3, No. 3, August 1969.

[GRIE-71] Gries, D., "Compiler Construction for Digital Computers," John Wiley, 1971.

[HACK-75] Hack, M., "Petri Net Languages," Computation Structures Group Memo 124, Project MAC, MIT, Cambridge, Massachusetts, June 1975.

[HOPC-69] Hopcroft, J.E. and Ullman, J.D., "Formal Languages and their Relation to Automata," Addison-Wesley, 1968.

[KOSA-73] Kosaraju, S.R., "Limitations of Dijkstra's Semaphore Primitives and Petri Nets," The Johns Hopkins University, Baltimore, Maryland, May 1973.

[MILL-74] Miller, R.E., "Some Relationships between Various Models of Parrallelism and Synchronization," RC 5074 (#22391), Mathematical Sciences Department, IBM Thomas J. Watson Research Center, October 1974.

[NOE-73] Noe, J.D. and Nutt, G.J., "Macro E-Nets for Representation of Parallel Systems," IEEE Transactions on Computers, Vol. C-22, No. 8, August 1973.

[NUTT-72] Nutt, G.J., "The Formulation and Application of Evaluation Nets," Ph.D. Thesis, Computer Science Group, Univ. of Washington, Seattle, 1972 (TR. 72-07-02).

[PATI-70] Patil, S.S., "Coordination of Asynchronous Events," Ph.D. Thesis, Project MAC, MIT, Cambridge, Massachusetts, 1970 (MAC-TR-72).

[PETE-73] Peterson, J.L., "Modelling of Parallel Systems," Ph.D. Thesis, Department of Electrical Engineering, Stanford University, 1973.

[RUMB-75] Rumbaugh, J.E., "A Parallel Asynchronous Computer Architecture for Data Flow Programs," Ph.D. Thesis, Project MAC, MIT, Cambridge, Massachusetts, May 1975 (MAC-TR-150).

[SALO-73] Salomaa, A., "Formal Languages," ACM Monograph Series, Academic Press, 1973.

[SONN-75] Sonnenburg, C.R., "A Configurable Parallel Computing System," Ph.D. Thesis, Program of Computer, Information and Control Engineering, The University of Michigan, Ann Arbor, Michigan, February 1975 (RADC-TR-74-353).

[URSC-72] Urschler, G., "The Inherent Parallelism of Flow Diagrams," Technical Report TR 25.129, IBM Laboratory Vienna, July 1972.

[URSC-73] Urschler, G., "The Transformation of Flow Diagrams into Maximally Parallel Form," Sagamore Computer Conference on Parallel Processing, 1973.

[URSC-75] Urschler, G., "Automatic Structuring of Programs," IBM J. Res. Develop., March 1975.