TECHNICAL REPORT 161

DECIDABILITY QUESTIONS FOR PETRI NETS

Michel Henri Théodore Hack

June 1976

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LABORATORY FOR COMPUTER SCIENCE

(formerly Project MAC)

CAMBRIDGE MASSACHUSETTS 02139

DECIDABILITY QUESTIONS FOR PETRI NETS

by

Michel Henri Théodore Hack

Submitted to the Department of Electrical Engineering and Computer Science on December 22, 1975, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

An understanding of the mathematical properties of Petri Nets is essential when one wishes to use Petri Nets as an abstract model for concurrent systems. The decidability of various problems which arise in this context is an important aspect of this question. The fact that these problems also arise in the context of other mathematical theories, such as commutative semigroups, closure under linear relations, Matrix Context-Free grammars, or Weak Counter Automata, provides further motivation.

The Reachability Problem for Vector Addition Systems - whose decidability is still an open question - is of central importance. We show that a number of Petri Net problems are recursively equivalent to this problem. These include the Liveness Problem (e.g. can a given system reach a deadlocked state?), the single-place reachability problem (can a given buffer ever be emptied?), the persistence problem (can a given transition ever be disabled by the firing of another transition?), and the membership and emptiness problems for certain classes of languages generated by Petri Nets.

The power of the unrestricted Petri Net model is illustrated by various undecidable equivalence results. In particular, we show that the equality of Reachability Sets and the equivalence of two Petri Nets in terms of their language-generating capability are recursively undecidable.

It is hoped that the constructions used to prove our results will shed some light on the source of the complexities of the unrestricted Petri Net model, and may eventually permit us to achieve an optimal balance between representational transparency and analytical power of the Petri Net model.

Thesis Supervisor: Suhas S. Patil

Title: Associate Professor of Electrical Engineering and Computer Science

ACKNOWLEDGEMENTS

I wish to thank the members of my thesis committee, Professors Suhas Patil, Albert Meyer and Robert Gallager, for their helpful suggestions during the preparation of this thesis. I am grateful to my colleagues P. S. Thiagarajan and Fred Furtek for many stimulating discussions. Thanks also to Professor Jack Dennis and his Computation Structures Group for an exciting research environment at Project MAC.

I thank my parents for their patience and interest in my work, and Gloria Marshall for her continuing friendship.

I also thank Mrs. Delphine Radcliffe for her patience and accuracy in typing this document.

I am grateful to Project MAC, Massachusetts Computer Associates (Dr. A. W. Holt in particular), and the IBM Fellowship Program for financial assistance.

TABLE OF CONTENTS

		Page
List of Definitions		7
List of Theorems and Lemmas		9
CHAPTER 1	INTRODUCTION	13
	1.1 Petri Nets and Concurrent Systems	13
	1.2 The Computer Science Motivation	15
	1.3 The Mathematical Motivation	18
	1.4 Object of this Thesis	21
	1.5 Previous Work	26
CHAPTER 2	BASIC DEFINITIONS AND PROPERTIES	29
	2.1 Generalized Petri Nets	29
	2.2 Restricted Petri Nets	35
	2.3 Reachability, Coverability, Boundedness, Liveness, and Persistence	35
	2.4 Subnets and Submarkings	38
	2.5 Vector Notation for Submarkings	44
	2.6 Some Mathematical Properties of the Set of Vectors over the Augmented Integers, Ω^r	49
CHAPTER 3	DECIDABILITY OF BOUNDEDNESS AND COVERABILITY	55
	3.1 Introduction	55
	3.2 Primary Unboundedness and the Primary Coverability Tree	57
	3.3 Boundedness of a given place and the Complete Coverability Tree	65

TABLE OF CONTENTS (continued)

		Page
CHAPTER 4	REACHABILITY PROBLEMS	72
	4.1 Reachability of a given Marking or Submarking	72
	4.2 Reachability of Some Marking in a given Set of Markings	76
CHAPTER 5	LIVENESS AND PERSISTENCE	83
	5.1 Liveness	83
	5.2 Persistence	91
CHAPTER 6	UNDECIDABILITY AND WEAK COMPUTATION	94
	6.1 The First Undecidability Proofs for Vector Addition Systems	94
	6.2 Diophantine Polynomials and Hilbert's Tenth Problem	95
	6.3 Weak Computation by Petri Nets . . .	98
CHAPTER 7	INCLUSION AND EQUALITY PROBLEMS FOR REACHABILITY SETS	113
	7.1 The Decidability Problems	113
	7.2 The Subspace Inclusion Problem (SIP) . .	114
	7.3 The Inclusion Problem (IP)	118
	7.4 The Equality Problem (EP)	121
CHAPTER 8	PETRI NET LANGUAGES: DEFINITIONS AND PROPERTIES	127
	8.1 Labelled Petri Nets	127
	8.2 Standard Form	131

Page

8.3 The Relationship between Prefix and Terminal Petri Net Languages . . . 135

8.4 Closure of Petri Net Languages under Union and Intersection 137

CHAPTER 9 PETRI NET LANGUAGES: MEMBERSHIP AND EMPTINESS PROBLEMS 146

9.1 Membership Problems 146

9.2 Emptiness Problems and Finiteness Problems 149

CHAPTER 10 PETRI NET LANGUAGES: EQUIVALENCE AND INCLUSION PROBLEMS 153

10.1 Petri Net Languages can Encode Polynomial Graphs 153

10.2 Undecidable Equivalence Problems . . 156

10.3 The Equivalence Problem for Sets of Firing Sequences 161

CHAPTER 11 CONCLUSION: OPEN QUESTIONS AND CONJECTURES 170

11.1 Is Reachability Decidable? 170

11.2 Some Sufficient Conditions for the Undecidability of RP 172

11.3 Decidability Questions for Restricted Classes of Petri Nets 173

11.4 Conclusion 175

APPENDIX: SETS OF VECTORS OVER THE AUGMENTED INTEGERS 177

REFERENCES 188

List of Definitions

		Page
D2.1	Generalized Petri Net (GPN)	29
D2.2	Firing of a Transition t	31
D2.3	Firing Sequence σ	32
D2.4	Set of Reachable Markings R_N Set of Firing Sequences S_N Set of Terminal Firing Sequences T_N	33
D2.5	Hurdle $H(\sigma)$, Marking Change $\Delta(\sigma)$	33
D2.6	Reachability	35
D2.7	Coverability	35
D2.8	Boundedness	35
D2.9	Potential Firability	36
D2.10	t-Deadness	36
D2.11	Liveness	36
D2.12	Persistence	37
D2.13	Closed Subnet	40
D2.14	Submarking	40
D2.15	Agreement	41
D2.16	Support P(V) of a Submarking	41
D2.17	Reachability of a Submarking	42
D2.18	Coverability of a Submarking	42
D2.19	Firability at a Submarking	43
D2.20	Potential Liveness at a Submarking	43
D2.21	t-Deadness at a Submarking	44
D2.22	Augmented Integers $\Omega = \mathbb{N} \cup \{\omega\}$	45
D2.23	Submarking as a Vector in Ω^r	45

List of Definitions (continued)

		Page
D2.24	Initial Submarking	46
D2.25	Chain	50
D2.26	Chain-Completeness	50
D2.27	Monotonicity	50
D2.28	Set of Maximal Elements $\hat{A}$ of Set A	51
D2.29	Chain-Completion A^c of Set A	51
D2.30	Linear Set	52
D2.31	Semilinear Set	53
D3.1	Primary Coverability Tree D_N	58
D3.2	Complete Coverability Tree $\hat{D}_N$	66
D4.1	RP-Solvability of a Set	77
D6.1	Diophantine Polynomial	95
D6.2	Polynomial Graph G(P)	95
D6.3	Petri Net Weak Computer	101
D7.1	Projected Reachability Set	113
D8.1	Labelled Petri Net, Labelling Function, λ-Free	128
D8.2	Label Sequence	128
D8.3	Petri Net Language Families $\mathcal{L}$, $\mathcal{L}^\lambda$, $\mathcal{L}_0$, $\mathcal{L}_0^\lambda$	129
D8.4	Standard Form of a Labelled Petri Net	131
D10.1	Parikh Mapping #(w), #(L)	153

List of Theorems, Lemmas and Corollaries (except Appendix)

		Page
T2.1	Containment property	34
T2.2	Reachability from Submarkings	47
T2.3	Coverability of Submarkings	48
T2.4	Finiteness of Sets of Mutually Incomparable Vectors . .	51
T2.5	Characterization of Monotone Sets by their Maximal Elements	51
T2.6	Chain-Completion of a Monotone Set	52
T2.7	Characterization of the Chain-Completion of a Monotone Set	52
T2.8	Finite Characterization of Monotone Sets in $\mathbb{N}^r$ . . .	52
T2.9	Closure of Semilinear Sets under Union, Intersection and Complement	53
T2.10	Semilinearity of the Solution Space of Linear Diophantine Equations	53
T2.11	Semilinearity of Monotone Sets in $\mathbb{N}^r$	54
L3.1	Coverable Submarkings and Chain-Completion . . .	56
L3.2	Coverability and Boundedness	56
L3.3	Finiteness of Primary Coverability Trees	61
L3.4	Labels in a Primary Coverability Tree are Coverable	63
T3.1	Boundedness of a Petri Net is Decidable	64
L3.5	Transitivity of Submarking Coverability	65
L3.6	Finiteness of Complete Coverability Trees	67
L3.7	Reachable Markings Agree with some Label in the Complete Coverability Tree	67
T3.2	A Submarking is Coverable iff it is Covered by some Label in the Complete Coverability Tree	69

List of Theorems, Lemmas, etc. (continued)

		Page
T3.3	The Labels of the Complete Coverability Tree Express the Bounds on the Places	70
T3.4	Decidability of Coverability and Place Boundedness .	71
C3.1	Decidability of Potential Firability, t-Deadness, Infinite Firability, etc.	71
L4.1	SRP Reducible to ZRP	72
L4.2	ZRP Reducible to SPZRP	74
T4.1	Recursive Equivalence of RP, SRP, ZRP, SPZRP . .	76
L4.3	Reachability Sets are RP-Solvable	78
C4.1	Common Marking Problem Equivalent to RP . . .	80
L4.4	Linear Sets in $\mathbb{N}^r$ are Reachability Sets	80
L4.5	Closure of RP-Solvable Sets Under Union	81
T4.2	Semilinear Sets are RP-Solvable	81
C4.2	Examples of Semilinear Sets	82
L5.1	Set of t-Dead Markings is Monotone	83
L5.2	t-Deadness of Submarkings is Decidable	84
T5.1	Liveness Reducible to RP	85
T5.2	Liveness Equivalent to RP	86
C5.1	Recursive Equivalence of LP and SLP	85
T5.3	Persistence (PP and SPP) Reducible to RP	92
T5.4	SPP Equivalent to RP	93
T6.1	Undecidability of PGIP	96
L6.1	Weak Computability of a Form of Multiplication . .	103
L6.2	Weak Computability of Monomials	105

List of Theorems, Lemmas, etc. (continued)

		Page
L6.3	Weak Computability of Polynomials for Positive Arguments	108
T6.2	Weak Computability of Diophantine Polynomials for Non-Negative Arguments	111
L7.1	Encoding of a Polynomial Graph as a Projected Reachability Set	114
T7.1	PGIP Reducible to SIP	117
C7.1	Undecidability of SIP	117
T7.2	SIP Reducible to IP	118
C7.2	Undecidability of IP (Rabin's Theorem)	121
T7.3	IP Reducible to EP	122
T7.4	EP, IP, SIP, SEP Equivalent and All Undecidable .	124
T7.5	Change in Reachability Set due to Removing Transition is Undecidable	126
T8.1	Standard Form for Labelled Petri Nets	132
T8.2	Generation of Prefix Languages as Terminal Languages	136
C8.1		137
T8.3	Effective Closure under Union of Petri Net Languages	139
C8.2		142
T8.4	Effective Closure under Intersection of Petri Net	142
C8.3	Languages	145
T9.1	Membership for $\mathcal{L}$, $\mathcal{L}_0$ Decidable	146
T9.2	Membership for $\mathcal{L}^\lambda$ Decidable	146
T9.3	Membership for $\mathcal{L}_0^\lambda$ Equivalent to RP	147
T9.4	Emptiness for $\mathcal{L}_0$, $\mathcal{L}_0^\lambda$ Equivalent to RP	150

List of Theorems, Lemmas, etc. (continued)

		Page
T9.5	Finiteness for $\mathcal{L}$, $\mathcal{L}^{\lambda}$ Decidable	150
T9.6	RP Reducible to Finiteness for $\mathcal{L}_0$, $\mathcal{L}_0^{\lambda}$	151
T10.1	Encoding of Polynomial Graphs as $\mathcal{L}$-Language . . .	154
T10.2	Equivalence and Inclusion for $\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^{\lambda}$, $\mathcal{L}_0^{\lambda}$ Undecidable	156
T10.3	Change in Language due to Removal of Transition is Undecidable	158
C10.1	Other Changes in Language which are Undecidable . .	158
T10.4	Undecidable Whether Prefix Language is also Terminal Language of the Same Net	160
T10.5	Inclusion and Equivalence for Sets of Firing Sequences Reducible to RP	162
T10.6	Inclusion and Equivalence for Sets of Terminal Firing Sequences Equivalent to RP	164

CHAPTER 1

INTRODUCTION

1.1 Petri Nets and Concurrent Systems

Petri Nets are best known as a graphical tool for the representation and analysis of concurrent or parallel systems. They originated from the work of C. A. Petri [54] in Germany in 1962. They were introduced to the U.S.A. by A. W. Holt in 1966. The notation most commonly used is also due to Holt [27]. In 1970 the interpretation of Petri Nets was generalized to permit unboundedness, such as occurs in a priori unbounded buffers (Holt and Commoner [28]). Further generalizations - to what we call "Generalized Petri Nets" - were proposed around 1972 by several people, including Commoner [8], Keller [34] and the author. We have shown in [18, 20] that these Generalized Petri Nets can themselves be suitably modelled by "ordinary" Petri Nets (1970 definition), so that the generalization essentially only buys modelling convenience, not more modelling power.

A Petri Net describes a concurrent system by expressing the relationship between elementary actions performed by the system and the resulting local change in the state of the system. In contrast to traditional automata theory, the state of a concurrent system is a structured entity, and "local change" means change to a specific structural component of the state of the system. Such local state changes can occur concurrently - that is to say, in a temporally independent fashion, where the concept of simultaneousness may be ill-defined - and thus the concept of "total system state" may also be ill-defined, except as an abstraction (imagine counting a moving crowd!). But this is a philosophical issue which need not concern us here.

If we want to use Petri Nets as a model for concurrent systems, we must provide analytical tools to answer the kind of questions we would like to ask about the concurrent systems. This implies a knowledge of the mathematical properties of Petri Nets.

To date, the mathematical properties are well known only for certain restricted classes of Petri Nets. In their full generality, there are still many unsolved problems. Even for bounded systems (where the number of possible configurations is finite) - which in theory can be grossly described by Finite State Automata - the problems are untractable, because the notion of total system state simply does not reflect the structure of the system, aside from any consideration of size.

We shall investigate the decidability of some important questions about the mathematical properties of Petri Nets. Specifically, we shall study whether there exist algorithms for testing whether a given Petri Net has a given property or not. For some properties, we can directly exhibit an algorithm for testing for them, but our main technique consists in proving the recursive reducibility of one problem to another: We show how to effectively construct an algorithm for one problem if we are given an algorithm (or an "oracle") for the other.

We believe that the techniques and constructions used in our proofs can also be very useful as general analytical tools for studying Petri Nets, because the reducibility proofs illustrate fundamental relationships between the various mathematical properties of Petri Nets. This is true even in the case of bounded systems, where decidability is a moot question, because the parallelism inherent in Petri Nets permits the representation of exceedingly complex finite-state systems by comparatively small Petri Nets. In fact, the complexity of bounded Petri Net

constructions can be just about as bad as for unbounded constructions.

It therefore appears that resolving the open decidability questions is not an end in itself, but a means for providing understanding and analytical tools for further questions of greater importance to the modelling of concurrent systems:

- Which restrictions are to be imposed on the general case to keep the complexity within bounds, and yet be able to model as extensive a class of systems as possible?
- Given suitable restrictions, which structural properties are important to an analysis of behavioural features of the system?
- What analytical procedures are to be used to relate such behavioural features to the identified structural features?

But there are also direct reasons for studying these decidability questions. The motivation does not come from concurrent system modelling, but rather from Automata Theory, Formal Language Theory, and Discrete Mathematics. Several open decision problems in these areas are related to the decision problems for Petri Nets. Also, Petri Nets can be formulated as a mathematical theory so simple that every undecidability result is surprising, and may shed some light on the minimal requirements to produce undecidability.

1.2 The Computer Science Motivation

Since 1963 (Estrin and Turn [13], Karp and Miller [33]), various formal systems have been developed for the purpose of modelling Concurrent Systems or Parallel Programs. The objective has been to provide models capable of answering questions peculiar to the notion of concurrency, such as non-determinacy, deadlocks, competition for

resources, critical and noncritical races, etc. These behavioural questions can often be related to structural questions about the concurrent system, such as decomposition into interacting components, the existence of critical substructures, global and local structural constraints, and the like.

The ease with which this modelling task can be accomplished depends heavily on two factors: Model transparency and analytical power. The first factor is the ability to relate structural features of the model to corresponding structural features of the concurrent system represented by the model. The second factor is the ability to use the model for answering questions about the concurrent system. It depends not only on the model itself, but also on the mathematical tools that are available to extract the desired information from the model.

When modelling parallel programs, a distinction is usually made between data flow and control flow. Program Schemata treat this discipline as a whole, and are used to answer questions about determinacy, functional equivalence, data access conflict, and the like. We wish to abstract further, and consider only the control aspect of parallel programs, i.e. the set of possible execution sequences without regard to the functional composition involved. For example, the control aspect of Karp and Miller's Parallel Program Schemata [33] and of Slutz' Flow Graph Schemata [60] is analyzed by these authors using Vector Addition Systems.

We have shown [18, 20] how Petri Nets and Vector Addition Systems can fully represent each other, and thus all questions concerning one system can be answered by studying the other.

Among parallel programming language constructs are Dijkstra's

Semaphore operations [12] and communication primitives such as fork and join. We shall only be concerned with the Semaphores and the position of the control loci in the various parallel processes; in a sense, we disregard all statements except P, V, goto (or while, with an undeterminate predicate), create and quit. Semaphore systems can be represented by Petri Nets (R. C. Holt, 1970 [29]; Patil, 1971 [51]); other references on the use of Petri Nets to represent parallel program control are [37, 38, 58, 59]. Here, the main problem of interest is the prevention of deadlock, a subject which has been extensively studied by R. C. Holt [29]. This corresponds to the Liveness Problem for Petri Nets, which is one of the open decision problems we study in this thesis.

Another field where Petri Nets have been useful is that of Asynchronous Control Structures (Dennis [11]). Some formalisms correspond to restricted classes of Petri Nets (Patil, 1969 [48]; Bruno and Altman, 1971 [5]; Jump and Thiagarajan, 1972 [31, 32]); some are slight variations (Patil, 1970 [49]; Noe and Nutt, 1972 [45]; Grandoni and Zerbetto, 1973 [15]);and others are quite general, such as Keller's Transition Systems and Vector Replacement Systems [34]. An extensive bibliography is given by Miller [42], who has also studied the relationship between some of these formalisms [43]. The interconnection of Asynchronous Modules by buffers has been studied by Patil [50]; such interconnections already generate structures with the complexity of Petri Nets in their most general sense. Problems of deadlocks are also important here; in addition, we would like to determine if a particular control state can be reached from some initial state. This is the Reachability Problem. This problem turns out to be the central decision problem, and it is not known whether it is decidable or not. Indeed, we do not know, in

general, whether the set of control states reachable from some initial configuration of the system is recursive or not.

To the extent that Petri Nets can represent the various formalisms presented so far, the decision problems for Liveness and Reachability are of concern to the computer scientist. But his main motivation in studying these problems is the insight this study may give into the structural and mathematical properties of his formalisms, as mentioned in the previous section.

1.3 The Mathematical Motivation

Vector Addition Systems - and therefore Petri Nets - turn up in several areas of automata theory and formal language theory. Minsky defined Program Machines (also known as Register Machines or Counter Automata) [43], which consist of a series of counters and a finite control which can increment or decrement individual counters and test individual counters for zero. If we have non-deterministic control and drop the zero-testing capability, we get a class of automata equivalent to Vector Addition Systems, which we call Weak Counter Automata [24]. (Baker [4] calls them Restricted Nondeterministic Counter Automata.) They are intimately related to the notion of weak computability as defined by Rabin [4, 56]. Whereas Minsky's Counter Automata can compute any partial recursive function (the arguments are the initial values in a set of input counters; the result is the contents of an output counter when the automaton halts), Restricted Nondeterministic Counter Automata can weakly compute a large class of arbitrarily fast growing monotonic primitive recursive functions - in particular, polynomials with non-negative integer coefficients (the output in a weak computation is the upper bound on the contents of the output counter over all possible (non-

deterministic) computations starting on a given input). This fact has enabled Rabin to prove the first known undecidability result about Vector Addition Systems. Following Rabin, we shall present the notion of weak computation by Petri Nets (Chapter 6), which we use to prove Rabin's result (Chapter 7) as well as some of our own undecidability results (Chapter 10).

The similarity between Petri Nets and Counter Automata can also be used to show how simple modifications to the firing rule, such as "zero-testing" arcs or "priority" firing rules, can dramatically increase the power of Petri Nets to equal the power of Turing Machines (Agerwala [2], Hack [24]). Many results in complexity theory about Vector Addition Systems and Petri Nets are also based on this relationship (Cardoza [6], Lipton [39]).

Van Leeuwen [63] has also studied the Reachability Problem for Vector Addition Systems, and points out that it is related to the recursiveness problem for Matrix Context-Free Languages, which differ from ordinary Context-Free Languages by the fact that the rules of the grammar are grouped in "Matrices", and all rules in one matrix must be applied in sequence, or else the matrix cannot be applied at all; the empty string is also allowed as a replacement for a nonterminal (otherwise the language would be trivially recursive). Also see Abraham [1], Crespi-Reghizzi and Mandrioli [9], Van Leeuwen [62]. This is one example where decidability itself is an issue: Any proof of the decidability or undecidability of the Reachability Problem for Petri Nets will also settle the emptiness and recursiveness problems for Matrix Context-Free Languages and, conversely, further research in that area may settle the Reachability Problem as well as the various Petri Net problems

that will be shown to be recursively equivalent to it.

There are in fact several ways in which Petri Nets are related to Formal Language Theory. As pointed out by Keller [34] and Crespi-Reghizzi [9], a Vector Addition System (in fact, a slight generalization thereof, due to Keller) can be considered as a commutative Semi-Thue system, and vice versa. A path in the Vector Addition System, or a control sequence in a concurrent system modelled by a Petri Net, corresponds in the Semi-Thue system to a derivation generating the vector or control state reached by that path or sequence.

A different approach has been taken by Baker [3], Peterson [52] and this author [24]. Instead of looking at the Petri Net as a grammar, let us look at it as a language-generating device. Each event occurrence - in addition to changing the control state of the system - also generates a symbol from some alphabet. We shall study the decision problems associated with these "Petri Net Languages" in Chapters 8, 9 and 10.

Another mathematical system equivalent to Petri Nets is the study of sets of integers closed under sets of linear relations of the form $R_{a,b} = \{\langle x, y\rangle \mid ax = by\}$ for integers a, b, x, y. Thus, the Reachability Problem is decidable iff*, for any finite set of pairs of integers $\langle a_i, b_i\rangle$, the closure of the set {2} under the linear relations R_{a_i, b_i}, as defined above, is effectively a recursive set. (Hack, 1973 [19])

Vector Addition Systems themselves can be formulated in the language of the mathematician. Let A be a commutative (additive) semigroup. A relation $R \subseteq A^2$ is said to be compatible iff $\forall a \in A$: $\langle x, y\rangle \in R \Rightarrow \langle x+a, y+a\rangle \in R$. The object is to study subsets of A closed under

---

* "iff" is a common abbreviation for "if and only if".

compatible relations. If A is finitely generated and finitely presented, and R has a finite number of minimal elements, we get Vector Addition Systems (Hack, 1974 [22]). Keller [34] and Van Leeuwen [63] have also pointed out that a restricted form of the Reachability Problem is related to the word problem for finitely generated and finitely presented commutative semigroups, and Cardoza [6] has studied this problem in terms of its computational complexity.

These examples show the possible impact of a solution of the decidability problems for Petri Nets. In contrast to the computer scientist, the mathematician may benefit from this result directly, and may be uninterested in the relationship to the behaviour of some underlying concurrent system.

We do in fact take this point of view in some of our proofs, where we use transformations which do not significantly change the set of reachable control states of some modelled concurrent system, but which behaviourally correspond to a total elimination of concurrency.

On the other hand, existing mathematical results in, say, the theory of commutative semigroups, may be helpful in some of our future proofs (for example, the first order theory of a given finitely generated commutative semigroup has been shown to be decidable by Taiclin [61]).

1.4 Object of this Thesis

In this section we shall briefly describe Petri Nets in the form in which they are used most frequently. But, before proceeding, we would like to state our bias in the approach to Petri Nets presented in this thesis.

Different people may have widely different views as to what

constitutes a Petri Net. To Carl Adam Petri, the Nets that we - and the computer scientists and mathematicians mentioned so far - use are only a very restricted interpretation of a much more primitive and general concept ultimately rooted in topology [55], a concept which at its coarsest level expresses the duality between actor and action, and at the finest level projects this duality into a geometry of the universe not unlike Minkowski's world lines and the conceptual pair force vs motion.

Our intentions are much less ambitious. We may use the semantic interpretation of concurrent systems modelling to motivate the various problems we wish to study, but in effect we wish to regard a Petri Net as a mathematical object, which can be defined and represented in a number of ways, depending on which properties of the model we wish to study. Thus, our vocabulary will be mainly that of sets and relations, although we also freely use the mental image of the Petri Net as a dynamic object, where things happen (occur), as in a concurrent system for example. This is actually the same attitude as that of a mathematician studying automata theory.

A Petri Net, as defined by Holt in 1970 [28], is a directed bipartite graph whose two vertex types are <u>places</u>, drawn as circles, and <u>transitions</u>, drawn as bars. This graph represents the structure of a concurrent system to be modelled: Certain collections of places may correspond to specific components in the system. The transitions then correspond to certain actions in the system which involve those components that contribute the places to which a transition is attached.

The state of a system component is described by a distribution of markers, or tokens, in the places corresponding to that system component; the occurrence of some action, which changes the state of

certain components, is modelled by the firing of a transition. This is done as follows:

A marking for a Petri Net is a function which assigns a non-negative integer to each place in the net; it can also be visualized as a vector of non-negative integers, each dimension corresponding to a specific place in the net. The marking expresses the distribution of markers over the places in the net at a given time: it indicates the number of tokens (possibly zero) on each place (drawn as dots inside the circle).

A transition is said to be firable iff every place which (in the directed bipartite graph) is connected to that transition by an arc pointing to the transition (input place of the transition) contains at least one marker. This expresses the system situation where the local configuration is such that all resources or enabling conditions for the action represented by the transition are available. A firable transition may fire; this changes the marking by removing one marker from each input place, and adding one marker to each output place (i. e. places connected to the transition by an arc pointing to the place). This models the occurrence of the enabled action in the system, and expresses the corresponding local change of configuration. In the case of a Petri Net used to recognize or generate a Petri Net Language, this transition firing can also be thought of as reading the corresponding symbol from an input tape, or printing the symbol on an output tape.

All our results will in fact be proved for the class of Generalized Petri Nets, which differ from the Ordinary Petri Nets described above only in the fact that the underlying graph is a directed bipartite multi-graph, i. e. there may be a bundle of one or more arcs from a given place to a given transition, or from a transition to a place. The firing

rule is such that each arc carries one token, so that a transition requires a token for each input arc to be enabled, and may remove or deposit several tokens in one place when it fires.

A simulation of the model then consists of a sequence of transition firings leading from a given initial marking to some reachable marking. The reachability set (also called marking class) is precisely the set of all markings that can be obtained after some firing sequence from a given initial marking.

The Reachability Problem (RP) is the problem of deciding whether a given marking is reachable (is in the reachability set) in a given Petri Net with a given initial marking. That is to say, in the concurrent system modelled by the Petri Net, we would like to know whether a particular configuration of the system can ever occur during operation.

The Reachability Problem refers to the total system state. Often a more meaningful question is whether a certain part of the system can ever be brought into a given local configuration by a sequence of actions starting from the initial configuration. In the Petri Net we ask whether any marking whose restriction to a given subset of the places is given can be reached from the initial marking. This is the Submarking Reachability Problem (SRP).

A special case of the RP is the Zero Reachability Problem, or ZRP, which asks whether all tokens can be removed from the net by some firing sequence. A special case of the SRP is the Single-Place Zero Reachability Problem, or SPZRP: does there exist a reachable marking in which a given place contains no tokens? Surprisingly, this very particular form of the Reachability Problem is recursively equivalent to the full Reachability Problem. When modelling a concurrent system,

this is the question of whether a given buffer can ever be emptied, or whether a given semaphore will ever reach zero and thus cause some process to become dormant on a P operation on that semaphore.

These Reachability Problems are studied in Chapter 4.

It can also happen that a system reaches a state after which two actions must wait for each other, creating a partial deadlock that cannot be resolved by any sequence of the remaining actions. Alternatively, some non-renewable resource may run out, also disabling a certain portion of the system. This situation is expressed in the Petri Net by a set of non-live transitions: A transition t is non-live iff a marking can be reached from which no firing sequence ever firing t is possible. A live transition thus has the property that, no matter what firing sequence has occurred so far, the transition can always eventually be fired again. A Petri Net is said to be live iff every transition is live.

The Liveness Problem (LP) is the problem of deciding, given a Petri Net and an initial marking, whether the net is live. The Subset Liveness Problem (SLP) asks whether a given transition (or set of transitions) is live.

Another important notion is that of persistence. A transition is said to be persistent iff the only way it can become disabled is by firing; no other transition firing may disable it. This corresponds to the notion of an irreversible commitment to perform a certain operation - once the decision is made to execute, nothing can remove the conditions which permit the planned operation but its own execution. The persistence problem (PP) is the question of whether a Net is persistent, i.e. whether all transitions are persistent. It is of course reducible to the SPP, which is the same question for a subset of transitions, or a single

transition.

We shall show (in Chapter 5) that all problems mentioned so far are recursively equivalent to each other, except for PP which is only known to be reducible to the others, via SPP. This may be because persistent Petri Nets have special properties - in particular the LP, when restricted to persistent Petri Nets, is decidable. It is not known yet whether PP itself is decidable or not, but we have some partial results which lead us to believe that PP is decidable, and that RP is decidable for persistent nets.

In fact, in Chapter 11 we shall present some circumstantial evidence to support our stronger conjecture that RP, and with it all problems mentioned above, are decidable.

Only one undecidability result was known for Petri Nets: Rabin's result on the undecidability of the Inclusion Problem for Reachability Sets. We shall add to this the undecidability of the Equality Problem for Reachability Sets (Chapter 7) and of the Equivalence Problems for various Petri Net Language families (Chapter 10).

We shall also consider the emptiness and membership problems for Petri Net Languages; these problems turn out to be either decidable or equivalent to RP (Chapter 9).

1.5 Previous Work

Practically all previous work done on the decision problems we are interested in has been done for Vector Addition Systems.

Vector Addition Systems were developed by Karp and Miller in 1966 to establish decidability results about their Parallel Program Schemata. In particular, they proved the decidability of boundedness and coverability for Vector Addition Systems [33]. (An improved version of this

proof, adapted to our purposes, will be presented in Chapter 3.) At the same time, M. Rabin studied the relationship between Reachability Sets and Semilinear Sets (Parikh, [46]). He concluded that there are non-Semilinear Reachability Sets, and proved that the Inclusion Problem for Reachability Sets (Is the Reachability Set of one VAS a subset of that of another VAS?) is recursively unsolvable. This proof was simplified in 1972 in response to Matijasevič's proof of the undecidability of Hilbert's 10th Problem [26,40]; an account of this can be found in Baker [4] and Hack [20]. We include an improved version of this proof in this thesis, because our own undecidability proofs use the same central idea of "weakly" computing polynomials (Chapters 6 and 7).

R. Keller discussed various decision problems for his Vector Replacement Systems [34], and considered certain restrictions under which the Reachability Problem would be decidable. He studied the Liveness Problem and showed, in particular, that the related problems of infinite firability and potential firability are decidable, and that Liveness is decidable for persistent nets. He also conjectured that the Liveness Problem was reducible to the Reachability Problem; we shall prove this conjecture (and its converse) in Chapter 5.

J. Van Leeuwen, using geometrical arguments, also proved certain decidable subcases of the Reachability Problem [63] by establishing the semilinearity of certain projections of Reachability Sets; he proved that all 3-dimensional Vector Addition Systems have Semilinear Reachability Sets.

B. O. Nash published the reducibility of the Reachability Problem to the reachability-of-zero and the reachable-from-zero problems [44]; we discovered a slightly stronger result (presented in Chapter 4) independently

at about the same time [20].

J. L. Peterson [52] studied one of the families of Petri Net Languages we consider in this thesis. Our own work on Petri Net Languages is reported in [24], and some new results can be found in [25]. In this thesis we dwell only on the decidability questions raised by Petri Nets as language generators (Chapters 9 and 10), and on the definitions and properties required for this purpose (Chapter 8).

The relationship between Petri Nets and other formalisms has been studied by many people, including Keller [34], Peterson [52], Peterson and Bredt [53], Miller [42,43], Lipton [38], etc.

Finally, let us mention some recent results about the complexity of various decision problems. Most problems are very difficult to decide. In fact, Lipton [39] has shown that both Reachability and Boundedness take at least Exponential Space to decide. The least known upper bound on the complexity of the Boundedness Problem is Ackermann's Function. The complexity of some Petri Net decision problems is studied in a paper by Jones and Lien [30].

(Added in May 1976:)

The first non-contrived problem whose complexity is non-primitive recursive arises from Petri Nets: Given two bounded Petri Nets, do they have equal reachability sets? It is reported in cardoza, Lipton and Meyer: "Exponential-Space Complete Problems for Petri Nets and Commutative Semigroups", Proceedings of the 8th Annual ACM Symposium on the Theory of Computing, Hershey, Pa., May 1976.

C. Rackoff (U. of Toronto) has also been able to improve the upper bound on the complexity of the Potential Liveness Problem, which is related to the boundedness problem, to space n!, down from Ackermann's function.

CHAPTER 2

BASIC DEFINITIONS AND PROPERTIES

2.1 Generalized Petri Nets

Definition 2.1:

A Generalized Petri Net (GPN) $N = \langle \Pi, \Sigma, F, B, M_0 \rangle$ consists of the following:

1. a finite set of places, $\Pi = \{p_1, \ldots, p_r\}$
2. a finite set of transitions, $\Sigma = \{t_1, \ldots, t_s\}$ disjoint from Π
3. a forwards incidence function, $F: \Pi \times \Sigma \to \mathbb{N}$ ($\mathbb{N}$ is the set of non-negative integers)
4. a backwards incidence function, $B: \Pi \times \Sigma \to \mathbb{N}$
5. an initial marking, $M_0: \Pi \to \mathbb{N}$

A GPN is represented graphically as follows:

1. places are represented by circles
2. transitions are represented by bars
3. circles and bars are connected by bundles of arcs: if p is a place and t is a transition, and $F(p, t) = 3$, we have a bundle of 3 arcs going from p to t; 3 is the size of the arc bundle.
4. a marking is represented by drawing a number of tokens into a place, or writing the number.

The graphical representation of a GPN is thus a directed bipartite multigraph with a marking. When we draw a bundle of arcs we expect each fibre to carry along one token when a transition fires. The firability of a transition is thus defined as follows:

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

$M_0 = \langle 5, 3, 0 \rangle$

$\Sigma = \{t_1, t_2, t_3, t_4\}$

$F(p_i, t_j)$ =

i \ j	1	2	3	4
1	1	3	0	0
2	0	3	0	0
3	0	0	1	5

$B(p_i, t_j)$ =

i \ j	1	2	3	4
1	1	0	1	0
2	2	1	0	0
3	0	2	0	0

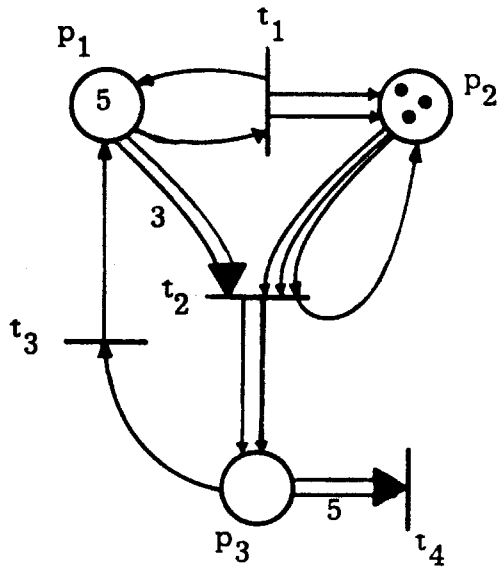


Figure 2.1

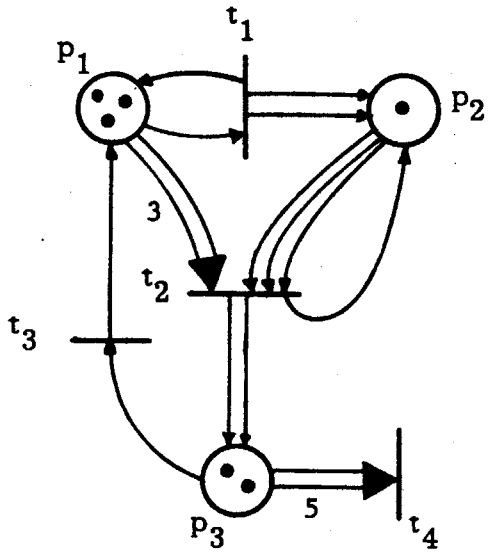


Figure 2.2

Definition 2.2:

(a) A transition t is said to be firable iff for every place $p \in \Pi$ we have $M(p) \geq F(p, t)$. Since this is always true when $F(p, t) = 0$ we need to inspect only the input places of transition t, i.e. those for which $F(p, t) > 0$.

(b) If a firable transition fires, it changes the marking by removing $F(p, t)$ tokens if p is an input place and by adding $B(p, t)$ tokens if p is an output place ($B(p, t) > 0$). The new marking M' is now such that:

$$\forall p: \quad M'(p) = M(p) - F(p, t) + B(p, t)$$

Usually, the sets of places and transitions are indexed, i.e. $\Pi = \{p_i \mid 1 \leq i \leq r\}$ and $\Sigma = \{t_j \mid 1 \leq j \leq s\}$. In this case, it is useful to represent markings as vectors in $\mathbb{N}^r$, where the i^{th} coordinate of vector M is the number $M(p_i)$. In this context, we associate with every transition t_j its input vector $F(t_j)$ and its output vector $B(t_j)$, where the i^{th} coordinate of $F(t_j)$ and $B(t_j)$ is $F(p_i, t_j)$ and $B(p_i, t_j)$, respectively.

Now we can interpret the firing of transition t as a relation $M[t\rangle M'$ which says "transition t is firable at marking M and the firing leads to marking M'", such that:

$$M[t\rangle M' \quad \Leftrightarrow \quad M \geq F(t) \quad \& \quad M' = M - F(t) + B(t)$$

A firing sequence can now be defined as a sequence of transition names (or a string σ in Σ^*), such that each prefix leads to a marking at which the following transition is firable. Thus, Figure 2.2 shows the result of firing t_2 in the Generalized Petri Net of Figure 2.1. Since t_3

is firable at that new marking, $t_2 t_3$ is a firing sequence. Note that $t_3 t_2$ is not a firing sequence, since t_3 is not firable at the initial marking.

The dynamic aspects of the Generalized Petri Net N can now be described by the set of firing sequences $S_N(M_0)$ starting at the initial marking M_0, and by the set of reachable markings $R_N(M_0)$, i.e. the markings M' such that some firing sequence $\sigma \in S_N(M_0)$ leads from M_0 to M' (also called reachability set). This we write as $M_0[\sigma\rangle M'$, where the relation $[\sigma\rangle$ is defined as the composition of the relations $[t_i\rangle$ for the transitions t_i as they occur in the string, so that composition for the relations corresponds to concatenation for the strings of transition names.

Formally, we have:

Definition 2.3:

A firing sequence from marking M to marking M' is represented by a string of transition names $\sigma \in \Sigma^*$ such that:

(a) $M[\lambda\rangle M$ (where λ is the empty string)

(b) $M[t\rangle M' \Leftrightarrow M \geq F(t) \;\&\; M' = M - F(t) + B(t)$
(for a string of length one)

(c) $M[\sigma t\rangle M' \Leftrightarrow \exists M'' \in \mathbb{N}^r: M[\sigma\rangle M'' \;\&\; M''[t\rangle M'$
(recursive definition)

Given a "final" marking M_f, we also define the set of terminal firing sequences $T_N(M_0, M_f)$ which contains all those firing sequences which lead from M_0 to M_f.

We summarize these concepts in:

Definition 2.4:

Given a Generalized Petri Net N with initial marking M_0 and final marking M_f:

The reachability set is $R_N(M_0) = \{M \in \mathbb{N}^r \mid \exists \sigma \in \Sigma^* : M_0[\sigma\rangle M\}$

The set of firing sequences is $S_N(M_0) = \{\sigma \in \Sigma^* \mid \exists M' \in \mathbb{N}^r : M_0[\sigma\rangle M'\}$

The set of terminal firing sequences is $T_N(M_0, M_f) = \{\sigma \in \Sigma^* \mid M_0[\sigma\rangle M_f\}$

Clearly, $T_N(M_0, M_f) \subseteq S_N(M_0)$ and $M_f \notin R_N(M_0) \Leftrightarrow T_N(M_0, M_f) = \emptyset$. We notice that:

$$M_0[\sigma\rangle M \Rightarrow \sigma \in S_N(M_0) \;\&\; M \in R_N(M_0)$$

Just as the marking F(t) is the smallest marking at which a given transition t is firable, there is a smallest marking at which a given firing sequence is firable. We call this the hurdle of the firing sequence:

Definition 2.5:

Let $\sigma \in \Sigma^*$ be an arbitrary firing sequence.

(a) The smallest marking at which σ can be fired in its entirety is called the hurdle $H(\sigma)$ of the firing sequence.

(b) If $M[\sigma\rangle M'$, then $M' - M$ is called the marking change $\Delta(\sigma)$ of the firing sequence.

It is easy to see that there is indeed a unique smallest marking at which a firing sequence σ is firable. This is because each coordinate of $H(\sigma)$ can be calculated independently.

Let us define the componentwise max of two vectors as the vector $V'' = \underline{\max}\,(V, V')$, where

$$V''(i) = \underline{\text{if}}\ V(i) \geq V'(i)\ \underline{\text{then}}\ V(i)\ \underline{\text{else}}\ V'(i)$$

Then we can calculate the hurdle (and also the marking change) of a firing sequence σ recursively as follows:

$$\left.\begin{array}{l} \forall \sigma \in \Sigma^* \\ \forall t \in \Sigma \end{array}\right\} \quad \begin{array}{lll} H(\lambda) = \langle 0 \rangle^r & & \Delta(\lambda) = \langle 0 \rangle^r \\ H(\sigma t) = \max\,(H(\sigma),\ F(t) - \Delta(\sigma)) & & \\ \Delta(\sigma t) = \Delta(\sigma) - F(t) + B(t) & & \end{array}$$

Notice that $H(\sigma) \in \mathbb{N}^r$ but $\Delta(\sigma) \in \mathbb{Z}^r$. Also, if there are no self-loops, then any firing sequence σ fired from $H(\sigma)$ to $H(\sigma) + \Delta(\sigma)$ makes each coordinate reach zero at some intermediate (including initial and final) marking. If there are self-loops, a coordinate may "reach" zero "during" a firing, i.e. after removing F(t) but before adding B(t) for some transition. Finally, we observe the following effect of increasing the initial marking:

Theorem 2.1:

Let $W \in \mathbb{N}^r$.

(a) $M_0[\sigma\rangle M_1 \Rightarrow (M_0 + W)\,[\sigma\rangle\,(M_1 + W)$

(b) $S_N(M_0) \subseteq S_N(M_0 + W)$

(c) $\{M \in \mathbb{N}^r \mid (M - W) \in R_N(M_0)\} \subseteq R_N(M_0 + W)$

(d) $T_N(M_0, M_f) \subseteq T_N(M_0 + W,\ M_f + W)$

Proof:

All four statements are manifestations of the containment property, which is most easily illustrated by distinguishing tokens due to M_0 from tokens due to W, and by not moving any tokens due to W.

QED

2.2 Restricted Petri Nets

In some cases it is useful to restrict the definition of Petri Nets. Ordinary Petri Nets are GPN's where the size of arc bundles is restricted to one. This corresponds to Holt's original definition [17, 28]. Selfloop-free Petri Nets have no pairs p, t that are both forwards and backwards connected, i.e. $B(p,t) \cdot F(p,t) = 0$ for all places p and transitions t. Restricted Petri Nets† (RPN) are Selfloop-free Ordinary Petri Nets: any place-transition pair is connected by at most one arc.

The relations between these various restrictions and Vector Addition Systems are discussed in a more detailed manner in Hack [20].

2.3 Reachability, Coverability, Boundedness, Liveness and Persistence

Definition 2.6:

A marking M is said to be reachable in a Petri Net N with initial marking M_0 iff: $M \in R_N(M_0)$.

Definition 2.7:

A marking M is said to be coverable in a Petri Net N with initial marking M_0 iff: $\exists M' \in R_N(M_0) : M' \geq M$.

Definition 2.8:

(a) A place p_i is said to be bounded in a Petri Net N with initial marking M_0 iff there exists an integer b_i such that the number of tokens $M(p_i)$ at any reachable marking M never exceeds b_i: $M \in R_N(M_0) \Rightarrow M(p_i) \leq b_i$.

(b) A Petri Net N with initial marking M_0 is said to be bounded iff every place is bounded.

---

† C. A. Petri calls these nets "Pure Petri Nets".

It follows that a Petri Net is bounded iff the reachability set $R_N(M_0)$ is finite.

Definition 2.9:

A transition t is said to be potentially firable at marking M in Petri Net N iff there exists a firing sequence starting at M which includes t.

It is easy to see that potential firability is related to coverability by:

t is potentially firable at M $\Leftrightarrow$ F(t) is coverable in $R_N(M)$

Definition 2.10:

A marking M is said to be t-dead (where t is a transition) iff transition t is not potentially firable at M.

This is just another way of looking at potential firability. We have:

M is t-dead $\Leftrightarrow$ F(t) is NOT coverable in $R_N(M)$

A t-dead marking is the analogue of a hang-up state, or a "deadly embrace", in the context of concurrent systems.

Definition 2.11:

(a) A transition t is said to be live in a Petri Net N with initial marking M_0 iff it is potentially firable at every reachable marking, or equivalently, iff no t-dead marking is reachable.

(b) A Petri Net N with initial marking M_0 is said to be live iff every transition is live.

(c) A firing sequence which reaches a t-dead marking is said to be a killing sequence (for t, or for the Net).

In other words, no matter what happens, it is always possible to fire a live transition once again.

We avoid speaking of "dead" transitions since the word seems equally suitable to describe a non-live transition or a not-potentially-firable transition. R. Keller suggests the word "immortal" instead of live, since it conveys a more precise image. The word "live" seems however to be the most widely used term for this concept in the Petri Net literature. R. C. Holt calls a live marking a "safe state" in the context of deadlocks in computer systems [29].

Definition 2.12:

(a) A transition t is said to be persistent in a Petri Net N with initial marking M_0 iff the only way it can be disabled is by its own firing.

(b) A Petri Net is said to be persistent iff every transition is persistent.

Note:

This definition of persistence of a transition can lead to ambiguity in the case of self-loops. Suppose both transitions t_1 and t_2 are firable, but the firing of t_2 would, because of a self-loop, return at least as many tokens as were taken away from the input places of t_1. Can such a firing ever disable t_1? If we only look at reachable markings, it does not seem so. But the usual interpretation is that "tokens are removed before they are returned", because this interpretation is more consistent with certain interpretations of

concurrency and the notion of "set firings".

This interpretation can be made precise by the following formula:

t is persistent in $R_N(M_0)$ $\Leftrightarrow$ $\forall t' \in \Sigma - \{t\}$; $\forall M \in R_N(M_0)$:

$(M \geq F(t)$ & $M \geq F(t')$ $\Rightarrow$ $M \geq F(t) + F(t'))$

In the other interpretation, where a self-loop could prevent non-persistence, we would have replaced the clause "$M \geq F(t) + F(t')$" by "$M \geq F(t) + F(t') - B(t')$".

The notion of persistence is useful in the context of Parallel Program Schemata (Karp and Miller [33], for example), where a persistent operator, once it becomes enabled, stays enabled until it fires. Also, in a persistent Net one cannot make irreversible "mistakes" in the sense that if one tries to follow a given firing strategy and one fires the "wrong" transition, this "mistake" can be corrected because what was supposed to be fired can still be fired. (In Keller's terms [35], a persistent net has the "Church-Rosser property".) The notion of persistence is also linked to the notion of "conflict-free" Nets.

The following table (Figure 2.3) illustrates the various concepts introduced so far as they apply to the example shown in Figure 2.1.

2.4 Subnets and Submarkings

In many cases we wish to restrict our attention to only a part of a given Petri Net. For example, one may ask whether it is possible to reach a marking consisting of exactly one token in each of two places, say p_1 and p_2, without specifying a desired marking for the remaining places. In that case, we speak of reaching a given "submarking" of places p_1 and p_2.

	Yes	No
Reachable from M_0	$\langle 0, 99, 3 \rangle$	$\langle 0, 98, 2 \rangle$
Coverable from M_0	$\langle 0, 98, 2 \rangle$	$\langle 0, 0, 5 \rangle$
Bounded at M_0	p_1	p_2
Firable at M_0	t_2	t_3
Potentially firable at M_0	t_3	t_4
Live at M_0	t_1	t_3
Persistent at M_0	t_3	t_2
t_4-dead	$\langle 5, 1, 0 \rangle$	$\langle 8, 0, 0 \rangle$

Figure 2.3

For this purpose, we introduce the notion of a subnet of a Petri Net $N = \langle \Pi, \Sigma, F, B, M_0 \rangle$, where $\Pi = \{p_1, \ldots, p_r\}$ and $\Sigma = \{t_1, \ldots, t_s\}$.

A subnet is basically a subgraph, i.e. one selects a subset of the vertices - in this case, places and transitions - and all arcs that join the selected vertices - in this case, the restriction of the functions F, B and M_0 to the chosen subset of their domain.

To be mathematically useful, however, a subnet should have certain properties. A very useful property is the property of being closed. This is actually a topological property of bipartite graphs which has been studied as such by Petri [55], but for our purposes (see also Hack [17,24]) the following definition will do:

Definition 2.13:

A closed subnet of a Petri Net is a subnet consisting of a subset of the places and at least all transitions forwards or backwards connected to places in this subset. If only transitions connected to places in this subset are included in the subnet, then it is called a minimal closed subnet with respect to this subset of places; if the subnet contains all transitions of the Petri Net, it is called a maximal closed subnet.

Notation:

If $P \subseteq \Pi$ is a subset of the places of Petri Net $N = \langle \Pi, \Sigma, F, B, M_0 \rangle$, then the maximal closed subnet whose set of places is P is denoted by $N_P = \langle P, \Sigma, F', B', M'_0 \rangle$, where F' and B' are F and B restricted to $P \times \Sigma$, and M'_0 is M_0 restricted to P.

Definition 2.14:

A submarking of a Petri Net N is a marking of a subnet of N, i.e. a marking restricted to a subset of the places.

Notation:

If P is a subset of the places, a submarking defined on these places is denoted by M/P and can be considered as a marking of N_P.

Definition 2.15:

(a) Two markings M and M' agree over a set of places P if their restrictions to P are equal, i.e. if they determine the same submarking over P. We write this as:

$$M = M' \text{ mod } P \Leftrightarrow M/P = M'/P$$

(b) Two submarkings M/P and M'/P' agree if they are equal on places common to both:

$$M/P \approx M'/P' \Leftrightarrow M = M' \text{ mod } (P \cap P')$$

The notion of agreement is useful in a context where both markings and submarkings over various sets of places are referred to. In particular, a marking agrees with any of its submarkings in the sense of (b):

$$M \approx M/P$$

The notion of agreement also permits a concise formulation of the extension to submarkings of the various definitions of section 2.3.

It is often useful to refer to a submarking directly, without explicitly mentioning the set of places on which it is defined. In order to avoid confusion with markings, we use the generic letter V for submarkings, so that we may write, for example: V = M/P, where M is some marking of which V is the restriction to P. Since in this notation the set P is not explicitly shown, we introduce the notion of support:

Definition 2.16:

The support P(V) of a submarking V is the set of places over which V is defined, i.e.: $V = M/P \Rightarrow P(V) = P$.

Now we are ready to extend the definitions of section 2.3 to submarkings.

Definition 2.17:

In a Petri Net N, a submarking V over a set of places P is said to be reachable from a marking M_0 iff some marking M whose restriction to P is the submarking V is reachable in N from M_0, i.e. some marking of which V is a submarking is reachable:

$$V \text{ reachable in } R_N(M_0) \quad \Leftrightarrow \quad \exists M \approx V: \; M \in R_N(M_0)$$

This is the formal way of defining the reachability of an incompletely specified marking, as in the example at the beginning of this section.

Definition 2.18:

A submarking V is said to be coverable in a Petri Net N with initial marking M_0 iff every marking of which it is a submarking is coverable:

$$V \text{ coverable in } R_N(M_0) \quad \Leftrightarrow \quad \forall M \approx V \; \exists M' \in R_N(M_0): \; M' \geq M$$

Notice the subtle difference between the definitions of reachability and coverability as extended to submarkings. In the first case, the property is derived from some marking which agrees with the submarking, whereas in the second case, the property must be true of all markings which agree with the submarking. In the first case we speak of the weak extension of a property of markings to submarkings and in the second case we speak of strong extension. The choice is dictated by the usefulness of the resulting concept. Definitions 2.17 and 2.18 define -

in more precise terms - weak reachability* and strong coverability of submarkings.

The strong reachability of a submarking might be an interesting property, but we have not found enough interesting applications to study it further. It is a non-trivial extension of the notion of reachability, and we have as yet no evidence that it might be reducible to reachability.

On the other hand, weak coverability is simply an instance of ordinary coverability of a marking which agrees with the given submarking and is zero on the places on which the submarking is not defined.

In the following definitions, the choice of the weak or of the strong extension of various concepts is dictated by similar considerations.

Definition 2.19:

Transition t of a Petri Net is firable at submarking V iff t is firable at some marking M which agrees with V:

$$t \text{ firable at } V \quad \Leftrightarrow \quad \exists M \approx V:\ M \geq V(t)$$

Definition 2.20:

Transition t is potentially firable at submarking V iff t is potentially firable at some marking M which agrees with V:

$$t \text{ potentially firable at } V \quad \Leftrightarrow \quad \exists M \approx V:\ t \text{ potentially firable at } M.$$

It is easy to see that a transition is firable at submarking V iff it is firable in $N_{P(V)}$ at V, where V is now the marking of the subnet $N_{P(V)}$ on whose places P(V) the submarking is defined.

From Theorem 2.2, proved later in this section, it will follow that this is also true for potential firability.

---

*In Van Leeuwen [63] weak reachability means coverability.

We can rewrite Definition 2.20 in terms of t-deadness:

Definition 2.21:

A submarking V is said to be t-dead for a given transition t iff every marking which agrees with V is t-dead:

$$V \text{ t-dead} \quad \Leftrightarrow \quad \forall M \approx V: \ M \text{ t-dead}.$$

We notice that the negation of a weak extension (Definition 2.20) is a strong extension (Definition 2.21).

In the case of liveness, neither the weak extension nor the strong extension to submarkings seems to be a useful concept, partly because there is no clear relationship between liveness in a subnet and liveness in the whole Petri Net. The same holds for persistence.

2.5 Vector Notation for Submarkings

The vector notation for markings was based on a certain indexing of the set of places, namely $\Pi = \{p_1, p_2, \ldots, p_r\}$. If we now study submarkings over the set $P = \{p_2, p_4\}$, for example, should we use vectors with two coordinates or vectors with r coordinates where r-2 coordinates are "undefined"? The second alternative has the advantage that the vector notation also carries information about the support of the submarking, namely those coordinates which are defined.

We therefore include a new symbol, ω, to denote the "value" of undefined coordinates in a submarking. Since we carry out additions, subtractions and comparisons with vectors, we must extend these operations to the symbol. We would expect that adding (or subtracting) something to (or from) an undefined quantity would yield an undefined quantity, i.e. ω again. But what about order? It turns out that the following rules for dealing with ω are not only consistent with our

intended use of submarkings, but that they provide a useful mathematical structure to the set of vectors over the non-negative integers augmented by the new symbol ω, which we denote by Ω, i.e. $\Omega = \mathbb{N} \cup \{\omega\}$.

Definition 2.22:

The augmented set of non-negative integers is the set $\Omega = \mathbb{N} \cup \{\omega\}$, where ω is an element which behaves like an integer larger than any given integer and is characterized by:

$$\forall n \in \mathbb{N}: \ \omega \neq n \ \& \ \omega \geq n \ \& \ \omega + n = \omega \ \& \ \omega - n = \omega \ \&$$
$$\omega + \omega = \omega - \omega = \omega$$

Now we represent submarkings as follows:

Definition 2.23:

A submarking M/P over a subset of places $P \subseteq \{p_i \mid 1 \leq i \leq r\}$ $(r \in \mathbb{N})$ is represented by the vector $V \subseteq \Omega^r$ whose i^{th} coordinate equals $M(p_i)$, the i^{th} coordinate of M, if $p_i \in P$; otherwise it is ω:

$$(1 \leq i \leq r): \ V(i) = \underline{\text{if}}\ p_i \in P \ \underline{\text{then}}\ M(p_i) \ \underline{\text{else}}\ \omega.$$

The usefulness of this definition appears when the definition of transition firability for submarkings is rewritten in terms of vectors over Ω^r:

$$t \ \underline{\text{firable at}}\ V \quad \Leftrightarrow \quad V \geq F(t)$$

This is of course just like the corresponding definition for markings.

This notation also gives us a way of talking about firing sequences and reachability in a subnet in the same context - and place indexing - as in the whole net. Let N_P be the maximal subnet of N defined by the sub-

set of places $P \subseteq \Pi$. Let V, V' be markings of N_P (i.e. submarkings of N whose support is P). Then we write:

$$V[t\rangle V' \quad \Leftrightarrow \quad V \geq F(t) \quad \& \quad V' = V - F(t) + B(t)$$

$V[\lambda\rangle V$, for λ = the empty string

$V[\sigma t\rangle V' \quad \Leftrightarrow \quad \exists V'' \in \Omega^r$: $V[\sigma\rangle V''$ & $V''[t\rangle V'$, where $\sigma \in \Sigma^*$.

Also, if $H(\sigma)$ and $\Delta(\sigma)$ are the hurdle and the marking change (Definition 2.5) of σ, then:

$$V[\sigma\rangle V' \quad \Leftrightarrow \quad V \geq H(\sigma) \quad \& \quad V' = V + \Delta(\sigma)$$

Notice that the above relations require that the supports of V, V' and V'' be equal: $P(V) = P(V') = P(V'')$.

Now we can define a subnet reachability set:

Definition 2.24:

(a) Let V_0 be a submarking of support P in a Petri Net N. Then the subnet reachability set for the initial submarking V_0 is the reachability set of the subnet N_P, which is written as:

$$R_N(V_0) = R_{N_P}(V_0) = \{V \in \Omega^r \mid \exists \sigma \in \Sigma^*: V_0[\sigma\rangle V\}$$

(b) The notions reachable in $R_N(V_0)$, coverable in $R_N(V_0)$, bounded in $R_N(V_0)$, etc., all refer to the corresponding concept in the subnet $N_{P(V_0)}$.

It is important to note that even if $V_0 \approx M_0$, then $V \in R_N(V_0)$ does not imply that V is reachable in N from M_0 according to Definition 2.17. It only expresses reachability in the subnet N_P, where some constraints, due to places in Π-P, have been removed. But the converse is true: If V is reachable in N from M_0 and V_0 is M_0 restricted to the support of V (i.e. $V_0 = M_0/P(V)$), then $V \in R_N(V_0)$.

This can easily be verified from Definition 2.17.

On the other hand, suppose that $V \in R_N(V_0)$, and let $V_0[\sigma\rangle V$. As we have seen, this implies $V_0 \geq H(\sigma)$. If we now choose M_0 to agree with V_0 on its support P and to agree with $H(\sigma)$ on the complement Π-P, i.e. $M_0 \approx V_0$ & $M_0 \approx H(\sigma)$ mod (Π-P), then $M_0 \geq H(\sigma)$, and hence σ is firable at M_0 and $M_0[\sigma\rangle M$, where $M = M_0 + \Delta(\sigma)$. Since $V = V_0 + \Delta(\sigma)$ and $M_0 \approx V_0$, it follows that $M \approx V$.

We summarize these facts in:

Theorem 2.2:

(a) If submarking V is reachable from the initial marking M_0, then V is reachable from the initial submarking V_0, where V_0 agrees with M_0 and has the same support as V:

$$V \text{ reachable in } R_N(M_0) \Rightarrow$$
$$\exists V_0 \in \Omega^r: V_0 \approx M_0 \;\&\; P(V) = P(V_0) \;\&\; V \in R_N(V_0)$$

(b) If a firing sequence σ leads from submarking V_0 to submarking V (of same support), then there exist markings M_0 and M, agreeing with V_0 and V respectively, such that σ leads from M_0 to M:

$$V_0, V \in \Omega^r: V_0[\sigma\rangle V \Rightarrow$$
$$\exists M_0, M \in \mathbb{N}^r: (M_0 \approx V_0 \;\&\; M \approx V \;\&\; M_0[\sigma\rangle M)$$

(c) $V \in R_N(V_0) \Rightarrow$
$$\exists M_0, M: (M_0 \approx V_0 \;\&\; M \approx V \;\&\; M \in R_N(M_0))$$

A useful application of Theorem 2.2 is the following characterization of coverability in a subnet (cf. Definition 2.24(b)):

Theorem 2.3:

Submarking V is coverable in $R_N(V_0)$ if and only if for every marking M which agrees with V, there exist markings M_0 and M' such that M_0 agrees with V_0, M' exceeds M, and M' is reachable from M_0.

In other words, the following three statements are equivalent:

(1) V is coverable in $R_N(V_0)$.

(2) $P(V_1) = P(V_0)$ & $V_1 \approx V \Rightarrow \exists V_2 \in R_N(V_0)$: $V_2 \geq V_1$

(3) $M \approx V \Rightarrow \exists M_0, M'$: $M_0 \approx V_0$ & $M' \geq M$ & $M' \in R_N(M_0)$

Proof:

(a) Statement (2) is the formal definition of coverability in a subnet, as it follows from Definitions 2.7 and 2.24(b). Thus (1) and (2) are equivalent by definition. The subnet is defined by the support $P = P(V_0) \subseteq \Pi$.

(b) (2) $\Rightarrow$ (3):

Let M_1 be an arbitrary marking such that $M_1 \approx V$, and let $V_1 = M_1/P$, i.e. the restriction of M_1 to the subnet defined by the support P of V_0. By hypothesis (2), there exists $V_2 \in R_N(V_0)$ such that $V_2 \geq V_1$.

By Theorem 2.2(c), $V_2 \in R_N(V_0)$ implies the existence of markings M_0 and M_2 such that $M_0 \approx V_0$ and $M_2 \approx V_2$ and $M_2 \in R_N(M_0)$.

Now let W be a marking which is zero over all places of the subnet, and which agrees with M_1 over all other places: (cf. Definition 2.15).

$$W \approx 0 \ \underline{\text{mod}}\ P \quad \& \quad W \approx M_1 \ \underline{\text{mod}}\ (\Pi - P)$$

Then we have:

$$\left.\begin{array}{l} M_0 + W \approx V_0 \\ M_2 + W \approx V_2 \end{array}\right) \quad \text{because } W \approx 0 \ \underline{\text{mod}}\ P$$

$$M_2 + W \geq M_1 \quad \text{because } W \approx M_1 \ \underline{\text{mod}}\ (\Pi - P) \text{ and } V_2 \geq V_1$$

Finally, by the containment property (Theorem 2.1(a) or (c)):

$$(M_2 + W) \in R_N(M_0 + W)$$

If we write $M'_0 = M_0 + W$

$M' = M_2 + W$

$M = M_1$

we have shown that:

(2) & $M \approx V \Rightarrow M'_0 \approx V_0$ & $M' \geq M$ & $M' \in R_N(M'_0)$

i.e. (2) $\Rightarrow$ (3).

(c) (3) $\Rightarrow$ (2)

Let V_1 be an arbitrary submarking such that $V_1 \approx V$ and $P(V_1) = P$, and choose some marking M which agrees with V_1, i.e. $V_1 = M/P$. Then M also agrees with V. By hypothesis (3), there exist markings M_0 and M' such that $M_0 \approx V$ and $M' \geq M$ and $M' \in R_N(M_0)$.

Now let V_2 be the restriction of M' to P, i.e. $V_2 = M'/P$. Since $V_0 = M_0/P$, we have $V_2 \in R_N(V_0)$ as a consequence of $M' \in R_N(M_0)$. But now $M' \geq M$ implies $M'/P \geq M/P$, i.e. $V_2 \geq V_1$. We have shown that:

$[(3)\ \&\ (V_1 \approx V\ \&\ P(V_1) = P(V_0))] \Rightarrow V_2 \in R_N(V_0)\ \&\ V_2 \geq V_1$

i.e. (3) $\Rightarrow$ (2).

QED

2.6 Some Mathematical Properties of the Set Vectors Over the Augmented Integers, Ω

Some of our proofs will require certain results about sets of vectors in Ω^r. These results are collected in this section, and the proofs can be found in the Appendix.

Recall that $\Omega = \mathbb{N} \cup \{\omega\}$, where ω satisfies the following

(Definition 2.22):

$$\forall n \in \mathbb{N}: \ \omega \neq n \ \& \ \omega \geq n \ \& \ \omega + n = \omega \ \& \ \omega - n = \omega \ \& \ \omega + \omega = \omega - \omega = \omega$$

The relation $(V \geq V' \ \& \ V \neq V')$ is abbreviated as $V > V'$. The relation of agreement (Definition 2.15) between vectors $V, V' \in \Omega^r$ can be expressed as:

$$V \approx V' \quad \Leftrightarrow \quad (\forall i, 1 \leq i \leq r: \quad V(i) + V'(i) \neq \omega \ \Rightarrow \ V(i) = V'(i))$$

For the partial order relation $\leq$, the set $\mathbb{N}^r$ is a lattice and the set Ω^r is a complete lattice, where every subset $A \subseteq \Omega^r$ has a unique least upper bound $W = \text{lub}\,(A)$ where $W \in \Omega^r$ and:

$$(\forall V \in A: \ V \leq W') \ \Leftrightarrow \ W \leq W'$$

Definition 2.25:

A chain $C \subseteq \Omega^r$ is a subset which is totally ordered under $\leq$, i.e. $C = \{V_0, V_1, \ldots V_j, \ldots\}$ and $V_{j+1} > V_j$ (for all j if C is infinite, or up to $j = |C| - 2$ if C is finite).

Definition 2.26:

A subset $A \subseteq \Omega^r$ is chain-complete iff, for every chain $C \subseteq A$, its least upper bound is an element of A: $\text{lub}\,(C) \in A$.

Since Ω^r is a complete lattice, the lub exists for every chain. In $\mathbb{N}^r$, however, infinite chains do not have a lub in $\mathbb{N}^r$.

Definition 2.27:

A subset $A \subseteq \Omega^r$ is monotone iff $\forall V \in A: \ V' \leq V \ \Rightarrow \ V' \in A$.

An example of a monotone set is the set of all vectors less than some vector from some given finite set. In fact, we shall se that every

monotone set can be expressed in this form.

Definition 2.28:

For a set $A \subseteq \Omega^r$, its set of maximal elements $\hat{A}$ is the set:

$$\hat{A} = \{V \in A \mid \nexists V' \in A: V' > V\}$$

Definition 2.29:

For a set $A \subseteq \Omega^r$, its chain-completion A^c is the smallest chain-complete set containing A.

The theorems we shall require are:

Theorem 2.4:

(a) Every infinite subset of Ω^r contains an infinite chain.

(b) Every set of mutually incomparable vectors in Ω^r is finite.

Theorem 2.5:

If $A \subseteq \Omega^r$ is monotone and chain-complete, then its finite set of maximal elements $\hat{A}$ is uniformly reducible to A, and it characterizes A as follows:

$$A = \{V \in \Omega^r \mid \exists V' \in \hat{A}: V' \geq V\}$$

By the uniform reducibility of $\hat{A}$ to A we mean that any procedure for testing membership in A can be effectively used to completely generate the finite set $\hat{A} = \{V_j \mid 1 \leq j \leq k\}$ where k is the size of $\hat{A}$. Technically, there exists a partial recursive function which computes a canonical index for $\hat{A}$ from a characteristic index for A (Rogers, [57]).

Theorem 2.6:

The chain-completion of a monotone set $A \subseteq \Omega^r$ is monotone and consists exactly of the least upper bounds of all chains in A. (If $A \subseteq \mathbb{N}^r$, then $A^c - A$ consists exactly of the least upper bounds of all infinite chains in A.)

Note:

Every element of A is the least upper bound of a one-element chain, and thus is included in A^c.

Theorem 2.7:

The chain-completion A^c of a monotone set $A \subseteq \mathbb{N}^r$ is such that:

$$A^c = \{V \in \Omega^r \mid \forall V' \in \mathbb{N}^r:\ V' \approx V \Rightarrow V' \in A\}$$

Theorem 2.8:

If $A \subseteq \mathbb{N}^r$ is monotone, then there exists a finite set $\{V_1, \ldots, V_k\} = \widehat{A^c}$, uniformly reducible to A^c, such that:

$$A = \{V \in \mathbb{N}^r \mid V \leq V_1 \ \underline{\text{or}}\ V \leq V_2 \ \underline{\text{or}}\ \ldots \ \underline{\text{or}}\ V \leq V_k\}$$

Finally, let us mention a few results about semilinear sets. Semilinear sets were introduced by Parikh [46] to study certain problems in Formal Language Theory, and more recently have become useful in investigations about Vector Addition Systems (Van Leeuwen, [63]) and Commutative Semigroups (Cardoza, [6]).

Definition 2.30:

A set $A \subseteq \Omega^r$ (or $\mathbb{N}^r$) is said to be linear iff there exist vectors $V_0 \in \Omega^r$ (called the base of A) and $W_i \in \mathbb{N}^r$, $1 \leq i \leq n$ (called the periods of A) such that:

$$A = \{V \in \Omega^r \mid \exists x_i \in \mathbb{N},\ 1 \leq i \leq n:\ V = V_0 + \sum_{i=1}^{n} x_i \cdot W_i\}$$

Matrix Notation:

Let W be the r × n matrix whose column vectors are the periods W_i, $1 \leq i \leq n$. Then we have: $A = \{V \in \Omega^r \mid \exists X \in \mathbb{N}^n: \ V = V_0 + W \cdot X\}$.

Definition 2.31:

A set $A \subseteq \Omega^r$ (or $\mathbb{N}^r$) is said to be semilinear iff it is the union of a finite number of linear sets in Ω^r.

Theorem 2.9:

(a) The union of a finite number of semilinear sets in Ω^r ($\mathbb{N}^r$) is a semilinear set in Ω^r ($\mathbb{N}^r$).

(b) The intersection of a finite number of semilinear sets in Ω^r ($\mathbb{N}^r$) is a semilinear set in Ω^r ($\mathbb{N}^r$).

(c) The complement $\Omega^r - A$ of a semilinear set $A \subseteq \Omega^r$ is a semilinear set in Ω^r; the complement $\mathbb{N}^r - A$ of a semilinear set $A \subseteq \mathbb{N}^r$ is a semilinear set in $\mathbb{N}^r$.

(a) follows from the definition; (b) and (c) are proved in Ginsburg and Spanier [14].

Theorem 2.10:

The solution space of a set of linear diophantine equations with dummy variables is a semilinear set.

This means that if A(t × r), B(t × s) and C(t × 1) are matrices over the integers $\mathbb{Z}$, then the set $\{V \in \mathbb{N}^r \mid \exists X \in \mathbb{N}^s: \ A \cdot V + B \cdot X = C\}$ is semilinear.

The proof of this can be found in Ginsburg and Spanier [14] and in Van Leeuwen [63].

Other examples of semilinear sets are mentioned in Corollary 4.2.

We can apply Theorems 2.9 and 2.10 to the characterization of monotone sets given by Theorem 2.8:

Theorem 2.11:

(a) Every monotone set in $\mathbb{N}^r$ is semilinear.

(b) If the chain-completion A^c of a monotone set $A \subseteq \mathbb{N}^r$ is effectively recursive, then A is effectively semilinear.

CHAPTER 3

DECIDABILITY OF BOUNDEDNESS AND COVERABILITY

3.1 Introduction

The decidability of boundedness and coverability was first proved for Vector Addition Systems by Karp and Miller [33], using the notion of a coverability tree. Karp and Miller's proof was not complete in the sense that it failed to take into account the complications arising from certain firing sequences which have a large hurdle but only a small or zero marking change. In Hack [20] we have presented a more detailed version of Karp and Miller's proof to handle all such situations.

A proof using geometrical arguments in the vector space $\mathbb{N}^r$ has also been presented by Van Leeuwen [63].

In this section we shall use some of the results on monotone sets in Ω^r presented in section 2.6. We feel that this approach may relate the properties of boundedness and coverability more directly to the structure of the Petri Net in terms of its subnets and submarkings. The approach is also slightly more general in that it applies directly to submarkings. But we must warn the reader that the conciseness of this approach is deceptive, since much of the mathematical work has simply been delegated to the proofs of the results of section 2.6 (given in the Appendix).

The coverability problem is the problem of deciding, given a Petri Net N with initial marking M_0 and an arbitrary marking M, whether M is coverable in $R_N(M_0)$, i.e. whether there exists a marking $M' \in R_N(M_0)$ such that $M' \geq M$.

Let us thus define the set of coverable markings $C_N(M_0)$:

$$C_N(M_0) = \{M \in \mathbb{N}^r \mid \exists M' \in R_N(M_0): M' \geq M\}$$

This set is clearly monotone by construction. Its chain-completion is, from Theorem 2.7:

$$C_N^c(M_0) = \{V \in \Omega^r \mid \forall M \in \mathbb{N}^r: M \approx V \Rightarrow (\exists M' \in R_N(M_0): M' \geq M)\}$$

Recalling the definition of submarking coverability (Definition 2.18), we have:

$$C_N^c(M_0) = \{V \in \Omega^r \mid V \text{ is coverable in } R_N(M_0)\}$$

Thus:

Lemma 3.1:

The chain-completion of the set of coverable markings is the set of coverable submarkings.

From Theorems 2.5 and 2.8 we can conclude that there exists a finite set of maximal coverable submarkings $\widehat{C_N^c}(M_0) = \{V_1, \ldots, V_k\}$ such that:

$$C_N^c(M_0) = \{V \in \Omega^r \mid V \leq V_1 \text{ or } \ldots \text{ or } V \leq V_k\}$$

$$C_N(M_0) = \{M \in \mathbb{N}^r \mid M \leq V_1 \text{ or } \ldots \text{ or } M \leq V_k\}$$

It is thus clear that the coverability problem for a fixed Petri Net is decidable, and quite efficiently so as a matter of fact.

Boundedness is related to coverability by:

Lemma 3.2:

A place p_i is bounded iff the submarking ($\forall j$, $1 \leq j \leq r$: $V(j) =$ if $j = i$ then ω else 0) is not coverable.

Proof:

If p_i is bounded, then there exists a bound b such that the marking

$(\forall j,\ 0 \leq j \leq r\colon M(j) = \underline{\text{if}}\ j = i\ \underline{\text{then}}\ b\ \underline{\text{else}}\ 0)$ is not coverable, hence V is not coverable. Conversely, if V is not coverable, then for some b there exists such a marking M, which determines a bound for p_i.

QED

If we now want to prove that the Boundedness and Coverability Problems are uniformly decidable, we have to effectively construct the finite set of maximal coverable submarkings. The Karp and Miller Coverability Tree is such a construction: the labels of the nodes in this tree constitute a finite set of coverable submarkings which contains all maximal coverable submarkings. In the following sections, we shall also construct coverability trees, in a step-by-step approach designed to illustrate more clearly the relationship between the coverability tree and various subnets of the Petri Net.

3.2 Primary Unboundedness and the Primary Coverability Tree

One way a place p_i may become unbounded is the following:

Let M_0 be the initial marking, and suppose there exists a firing sequence $\sigma_1\sigma_2$ such that:

$$M_0[\sigma_1\rangle M_1 \ \&\ M_1[\sigma_2\rangle M_2 \ \&\ M_2 \geq M_1 \ \&\ M_2(p_i) > M_1(p_i)$$

Because of $M_2 \geq M_1$, every firing sequence possible from M_1 is also possible from M_2; in particular, σ_2 can be repeated, and therefore $\sigma_1(\sigma_2)^*$ is a legal set of firing sequences. But then it is clear that by repeating σ_2 arbitrarily often, the marking in p_i can grow without bounds. In particular, after the firing sequence $\sigma_1(\sigma_2)^n$, the marking will be $M_1 + n \cdot (M_2 - M_1)$. All places p_j for which $M_2(p_j) - M_1(p_j) > 0$ will be unbounded.

This is called primary unboundedness.

But this is not the only way a place can become unbounded. For example, in the Petri Net of Figure 3.1 place p_4 is unbounded: given any number n, the firing sequence $(t_1)^n\, t_2 (t_3)^n$ yields the marking $\langle 0, 1, 0, n \rangle$. But for no pair of reachable markings such that $M_2 \geq M_1$ do we also have $M_2(p_4) > M_1(p_4)$. This net incidentally has the interesting property that t_3 can fire any finite number of times, but cannot fire indefinitely (see the "reachability graph" of this net in Figure 3.2).

However, in this case the unboundedness of p_4 follows from that of p_3, for which we do find two markings having the property described here: $M_0[t_1\rangle M_1$ and $M_1 \geq M_0$ and $M_1(p_3) > M_0(p_3)$. Because of this dependency, the unboundedness of p_4 may be called secondary unboundedness. In the next section we shall see how this is related to primary unboundedness in a subnet.

The following construction, which we call a primary coverability tree, is useful for investigating primary unboundedness. We define it in the general case of a subnet with an initial submarking.

Definition 3.1:

The primary coverability tree $D_N(V_0)$ of a given Petri Net with a given initial submarking V_0 (or subnet defined by the support $P(V_0)$ of the initial submarking) is a labelled rooted tree defined iteratively as follows:

base: The root node ρ is labelled V_0: $L_\rho = V_0$.

step: Let α be a node with label L_α which has not yet been declared as a leaf-node. There are four cases.

(a) No transition is firable at submarking L_α, i.e. $\forall t \in \Sigma$: $L_\alpha \not\geq F(t)$. In that case α is a leaf-node called a dead-end.

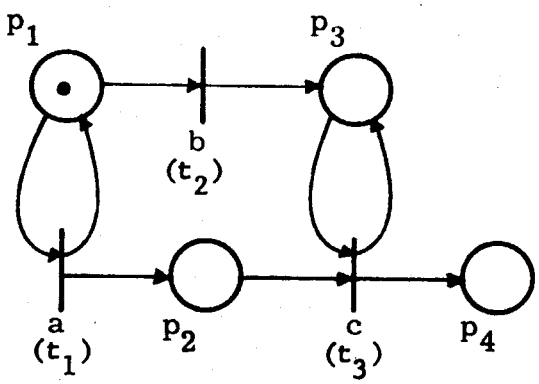


Figure 3.1

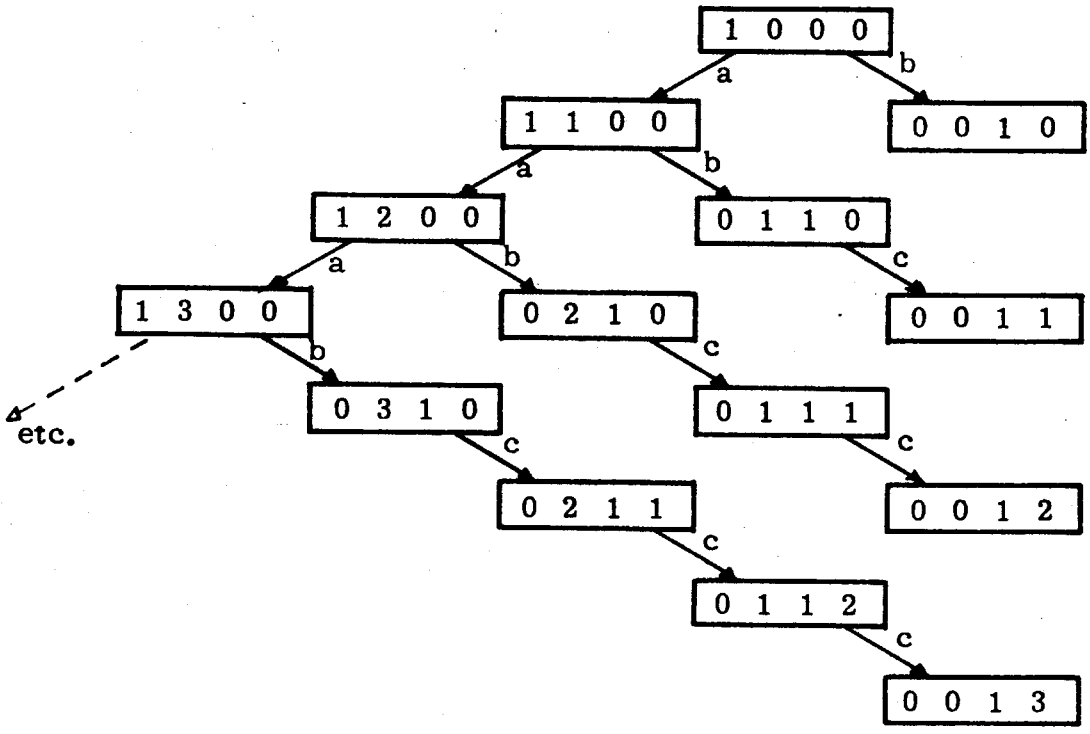


Figure 3.2

(b) There exists a node $\gamma \neq \alpha$ on the path from ρ to α such that $L_\alpha = L_\gamma$. In that case α is a leaf-node called a λ-loop-end, and a λ-backpointer points from α back to γ. This pointer is for record-keeping only and is not an arc of the tree.

(c) There exists a node γ on the path from ρ to α such that $L_\alpha > L_\gamma$. In that case α is a leaf node called an ω-loop-end, and an ω-backpointer (also for record-keeping only) is directed from α back to γ. In addition, the label L_α is modified by setting those coordinates in which L_α strictly exceeds L_γ to ω.

(d) If neither of the above cases holds, then α is an interior mode, and it has a successor node whose label is $L_\alpha - F(t) + B(t)$ for every transition t firable at L_α. The arcs pointing to the successor nodes are labelled with the transition whose firing they express.

Note 1:

This definition differs from that of a full coverability tree given in Hack [20], Karp and Miller [33] or Keller [34] essentially in the fact that only primary unboundedness is found (relative to a subnet in case of an initial submarking), and so nodes where new ω's are introduced are leaf-nodes, i. e. nodes without successors in the tree.

Note 2:

Step (c) in this definition may be interpreted in several ways if there exist more than one node γ on the path from ρ to α such that $L_\alpha > L_\gamma$. We may choose one arbitrarily, in which case the primary coverability tree is not unique, or we may choose all such nodes and generate appropriately many ω-backpointers, each causing some set of new ω-coordinates. The proofs which follow do not essentially

depend on which interpretation we choose. The proof of Lemma 3.4 is written for a single ω-backpointer, and the argument only has to be repeated for the other ω-backpointers, if any.

Figure 3.3 shows two primary coverability trees for the Net of Figure 3.1.

Lemma 3.3:

Every primary coverability tree is finite and can be effectively constructed.

Proof:

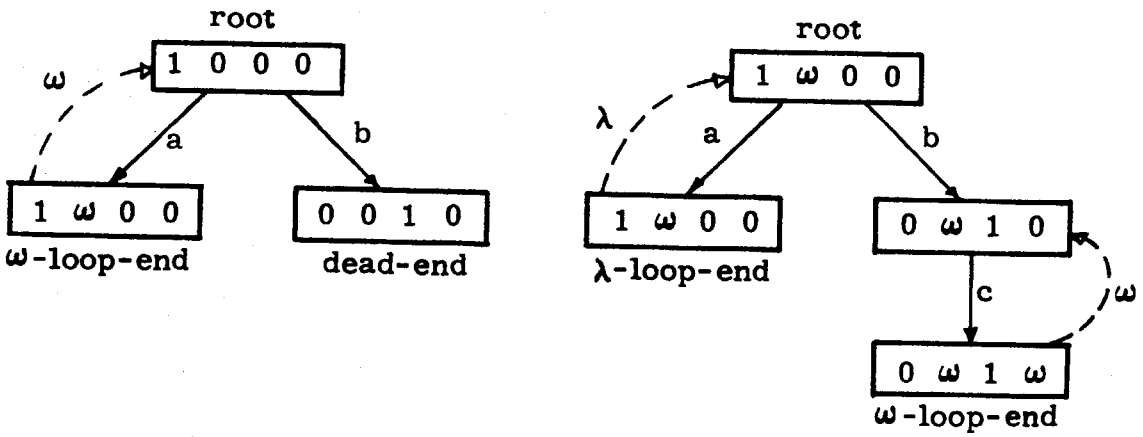
Suppose the tree is infinite. By construction, every node has at most as many immediate successors as there are transitions in the Petri Net, a finite number. Then, by König's Infinity Lemma for rooted trees, there must be an infinite path in the tree, i.e. a path which does not eventually end at a leaf node. But then, by Theorem 2.4(a), there must be an infinite subsequence non-decreasing in each coordinate of the sequence of node labels along that infinite path. This implies the existence of two nodes α and β along the path, where α is reached before β, such that $L_\beta \geq L_\alpha$. But then node β should be a leaf-node - either a λ-loop-end or an ω-loop-end, which contradicts the existence of an infinite path.

Since the tree must be finite, the iterative definition can be used as a terminating algorithm to construct it.

QED

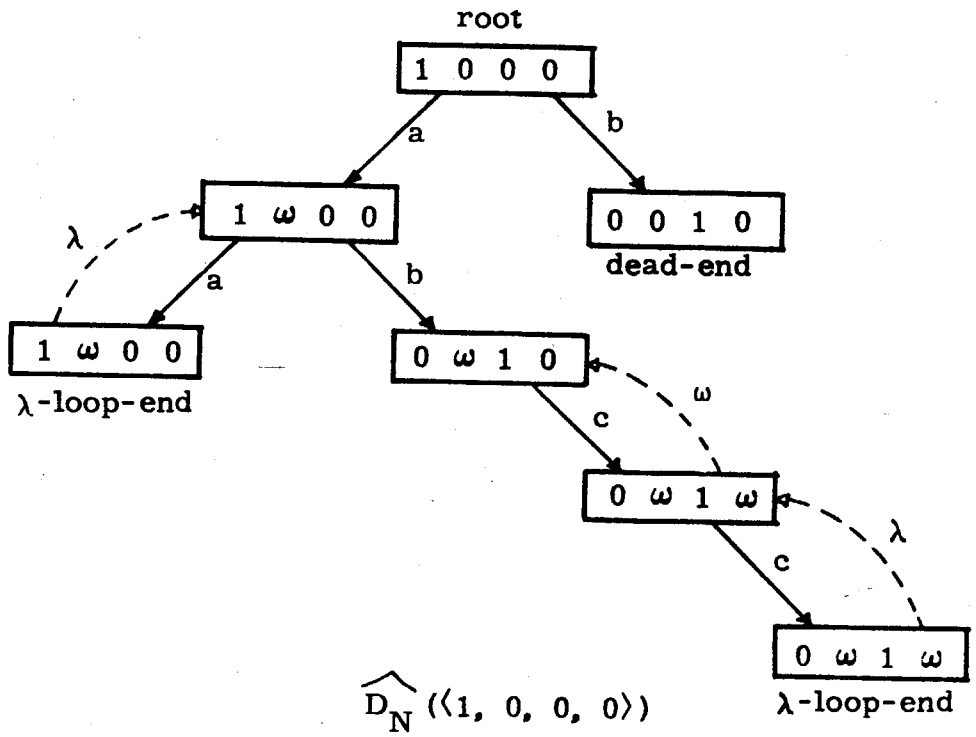
Note:

König's Infinity Lemma for rooted trees can easily be proved non-constructively. Assume the rooted tree is infinite, yet at each node there is a finite number of branches. Then at least one of the



$D_N(\langle 1, 0, 0, 0\rangle)$ $D_N(\langle 1, \omega, 0, 0\rangle)$

Figure 3.3



$\widehat{D_N}(\langle 1, 0, 0, 0\rangle)$

Figure 3.4

branches from the root node must point to the root of an infinite subtree. The path traced out by the root nodes of such successive infinite subtrees must be an infinite path -- QED. König's original Infinity Lemma [36] is more general. We provide a translation of his proof in Hack [20].

The reason for introducing new ω-coordinates in the label of an ω-loop-end, which indicates primary unboundedness, becomes clear from:

Lemma 3.4:

If V is the label of some node α in the primary coverability tree $D_N(V_0)$, then V is coverable in $R_N(V_0)$.

Proof:

Let us adopt the convention that if a path (a forwards sequence of labelled arcs in the tree $D_N(V_0)$) from node α to node β spells out a sequence σ of arc labels, we write $\alpha[\sigma\rangle\beta$. From the construction of $D_N(V_0)$ it follows that if β is not an ω-loop-end, then the firing sequence σ also leads from L_α to L_β:

$$\sigma \in \Sigma^*;\ \alpha,\beta \text{ nodes in } D_N(V_0):\ \alpha[\sigma\rangle\beta \ \Rightarrow\ L_\alpha[\sigma\rangle L_\beta$$

Thus, if α is not an ω-loop-end, then $\rho[\sigma\rangle\alpha$ for some path σ implies $V_0[\sigma\rangle V$, i.e. V is in fact reachable in $R_N(V_0)$.

If α is an ω-loop-end, then there exists, by construction, an internal node γ such that:

$$\gamma[\sigma\rangle\alpha \ \ \&\ \ L_\alpha > L_\gamma \ \ \&\ \ L_\gamma \in R_N(V_0)$$

Since $L_\alpha > L_\gamma$, σ is also firable at L_α, in fact arbitrarily often, and each repetition of σ increases the marking in the coordinates corresponding to the new ω-coordinates, whereas the marking in the

finite coordinates agrees with L_α. Thus the new ω-coordinates are unbounded in $R_N(V_0)$, and L_γ is coverable in $R_N(V_0)$.

QED

Before proceeding to search for all unbounded places (in the next section), we show that the primary coverability tree is sufficient to decide boundedness of the whole Petri Net:

Theorem 3.1:

It is decidable whether a given Petri Net with its initial marking M_0 is bounded.

Proof:

If the primary coverability tree contains ω-loop-ends, then the net is unbounded, by Lemma 3.4 above. Now suppose there are no ω's, i.e. every leaf node is either a dead-end or a λ-loop-end. If we fold all λ-loop-ends along their λ-backpointers (by identifying the λ-loop-end node with the interior node), we obtain a finite graph where the vertices are labelled with markings, and where for every node α whose label is M, and for every transition t which is firable at M, there exists an arc labelled t which leads from α to a node β whose label is M', such that $M[t\rangle M'$. In other words, every firing sequence σ starting at M_0 and leading to $M \in R_N(M_0)$ can be spelled by the arcs along a path from ρ to some node α labelled M. So every reachable marking is represented in the graph. Since the graph is finite, the number of reachable markings is finite, so the net must be bounded. In fact, the bounds for the various places can be found by inspecting the labels of the graph.

QED

3.3 Boundedness of a Given Place and the Complete Coverability Tree

To establish the unboundedness of a Petri Net, it is sufficient to establish the existence of some unbounded place by constructing the primary coverability tree for the initial marking.

But if we also construct primary coverability trees for the submarkings which label ω-loop ends, we can find more unbounded places, including places which are not primary unbounded. Indeed, we have:

Lemma 3.5:

If V is coverable in $R_N(V_0)$, and V' is coverable in $R_N(V)$, then V' is coverable in $R_N(V_0)$.

Proof:

Let M' be an arbitrary marking which agrees with V':

$$(1)\quad M' \approx V'$$

By Theorem 2.3, since V' is coverable in $R_N(V)$, there exist markings M and M_1 such that:

$$(2)\quad M \approx V$$

$$(3)\quad M_1 \geq M'$$

$$(4)\quad M_1 \in R_N(M)$$

Since V is coverable in $R_N(V_0)$, Theorem 2.3 applied to (2) implies the existence of M_0 and M_2 such that:

$$(5)\quad M_0 \approx V_0$$

$$(6)\quad M_2 \geq M$$

$$(7)\quad M_2 \in R_N(M_0)$$

Now rewrite (6) as:

$$(8)\quad M_2 = M + W, \text{ where } W \geq 0$$

and define:

(9) $M'' = M_1 + W$, where $W = M_2 - M \geq 0$

From Theorem 2.1 (containment) applied to (4) and (9) we deduce:

(10) $M'' \in R_N(M_2)$

Thus, given $M' \approx V'$ (1), we deduce the existence of M'' and M_0 such that:

(11) $M_0 \approx V_0$ (5)

(12) $M'' \geq M'$ from (3) and (9)

(13) $M'' \in R_N(M_0)$ from (7) and (10)

But then Theorem 2.3 implies that V' is indeed coverable in $R_N(V_0)$.

QED

This Lemma justifies the construction of the Complete Coverability Tree out of primary coverability trees as follows:

Definition 3.2:

The Complete Coverability Tree $\hat{D}_N(M_0)$ of a Petri Net N with initial marking M_0 is constructed iteratively as follows:

basis:

Construct the primary coverability tree $D_N(M_0)$. Its λ-loop-ends and its dead-ends are leaf nodes of $\hat{D}_N(M_0)$, but all other nodes are interior nodes; the ω-loop ends are still distinguished, but they are considered interior nodes.

step:

If α is an ωloop-end with label $L_\alpha = V$, append the primary coverability $D_N(V)$ by identifying α with the root node of $D_N(V)$. All nodes of $D_N(V)$ except λ-loop-ends and dead-ends become interior nodes of $\hat{D}_N(M_0)$.

If there are no ω-loop-ends left, the construction of $\hat{D}_N(M_0)$ is complete.

Figure 3.4 shows the complete coverability tree for the Net of Figure 3.1.

This construction terminates and is effective, because:

Lemma 3.6:

The complete coverability tree $\hat{D}(M_0)$ is finite and can be effectively constructed.

Proof:

Any branch in $\hat{D}(M_0)$ consists of a sequence of finite branches from primary coverability trees, and each time a new primary coverability tree is encountered, the number of ω-coordinates of the labels increases, and the support of the corresponding submarkings strictly decreases. A branch of $\hat{D}(M_0)$ therefore consists of a finite number of finite segments, and is finite. Since branching at every node is finite, the tree is finite by König's Lemma.

QED

In the proof of Theorem 3.1 we showed that if a primary coverability tree contains no ω-loop-ends, then every firing sequence from the initial marking (or submarking) can be folded onto the graph obtained by closing the λ-loops.

The same construction can be applied to complete coverability trees, because in a complete coverability tree the only leaf-nodes are λ-loop-ends and dead-ends.

Lemma 3.7:

If a marking M is reachable from M_0 in a Petri Net N, then the complete coverability tree $\hat{D}_N(M_0)$ contains a node α whose label agrees with M:

$$M \in R_N(M_0) \Rightarrow \exists \alpha \in \hat{D}_N(M_0): L_\alpha \approx M$$

Proof:

Let σ be a firing sequence leading to M, i.e. $M_0[\sigma\rangle M$. The proof is by induction on the length of σ.

basis: $\sigma = \lambda$ (the empty firing sequence)

Then $M = M_0$ and $\alpha = \rho$, the root node: $L_\rho = M_0$.

step:

$\sigma = \sigma' \cdot t$ and there exists a node α' such that $L_{\alpha'} \approx M'$, where $M_0[\sigma'\rangle M'$. We have $M'[t\rangle M$, so α' is not a dead-end. We may also assume that α' is not a λ-loop-end; if it were, its λ backpointer would point to a node γ with the same label, and we could have chosen that node instead.

It follows that α' is an interior node, and there exists a successor node α, joined to α' by an arc labelled t, whose label is obtained from $V = L_{\alpha'} - F(t) + B(t)$.

If α is not an ω-loop node in some component primary coverability tree, then L_α is simply equal to V (step d in Definition 3.1). Since $L_{\alpha'} \approx M'$ and $M = M' - F(t) + B(t)$, we have $V \approx M$, and hence also $L_\alpha \approx M$.

If α is an ω-loop node in a component primary coverability tree, then its label L_α is obtained from V by replacing certain coordinates by ω. But this still permits us to infer $L_\alpha \approx M$ from $V \approx M$.

In every case, we have proved the existence of a node α whose label agrees with M. Moreover, the firing sequence σ can be spelled out by a sequence of paths in $\hat{D}_N(M_0)$ from ρ to α linked by λ-backpointers. This, incidentally, is the reason for "labelling" these backpointers with the symbol for the empty string, λ.

QED

Now we can use the Complete Coverability Tree to answer questions about coverability and boundedness:

Theorem 3.2:

A submarking V is coverable in $R_N(M_0)$ if and only if some node α in $\hat{D}_N(M_0)$ carries a label which covers V: $L_\alpha \geq V$.

Proof:

(a) if:

Every label in the primary coverability tree $D_N(M_0)$ is coverable in $R_N(M_0)$, by Lemma 3.4. Because of Lemma 3.5, this property extends inductively to all nodes in $\hat{D}_N(M_0)$. Indeed, let α be an ω-loop-end whose label V is already known to be coverable in $R_N(M_0)$. Then every node in the primary coverability tree appended to α in the construction of $\hat{D}_N(M_0)$ is coverable in $R_N(V)$ by Lemma 3.4, and hence coverable in $R_N(M_0)$ by Lemma 3.5.

Thus, if $L_\alpha \geq V$ for some node α and some given submarking V, then the coverability in $R_N(M_0)$ of L_α implies the coverability of V.

(b) only if:

If V is coverable in $R_N(M_0)$, then every marking M which agrees with V is coverable in $R_N(M_0)$, by definition. So let us choose M such that its unspecified coordinates (those corresponding to ω-coordinates in V) are larger than any finite coordinate of all labels in $\hat{D}_N(M_0)$. Since M is coverable in $R_N(M_0)$, there exists $M' \geq M$ such that $M' \in R_N(M_0)$. By Lemma 3.7, there exists a node $\alpha \in \hat{D}_N(M_0)$ such that $L_\alpha \approx M'$. The finite coordinates of V are covered by M' and hence by L_α. The ω-coordinates of V correspond to coordinates which, in M

and thus also in M', are larger than any finite coordinates of all labels, such as L_α. Thus L_α must have ω-coordinates where V has ω-coordinates: L_α exceeds (or equals) V in all coordinates: $L_\alpha \geq V$.

QED

Theorem 3.3

(a) A place p_i is unbounded in $R_N(M_0)$ if and only if some node α in $\hat{D}_N(M_0)$ has a label L_α whose i^{th} coordinate is ω.

(b) The largest number of tokens b_i that can ever accumulate in place p_i is the largest value taken by the i^{th} coordinate over all labels in $\hat{D}_N(M_0)$.

Proof:

(a) By Lemma 3.2, place p_i is unbounded iff a vector whose i^{th} coordinate is ω (and all other coordinates are zero) is coverable. By Theorem 3.2 this is equivalent to saying there exists a label whose i^{th} coordinate is ω.

(b) Suppose p_i is bounded, and the largest reachable number of tokens is b_i. Let M be a marking which achieves the bound, i.e. the i^{th} coordinate of M is equal to b_i. By Lemma 3.7 there exists a node α such that $L_\alpha \approx M$. By part (a) above, the i^{th} coordinate of L_α cannot be ω, and hence must equal b_i. If some node β had a label L_β whose i^{th} coordinate exceeded b_i, then by Theorem 3.2 some marking whose i^{th} coordinate exceeds b_i would be reachable, contradicting the fact that b_i is a bound on the number of tokens in p_i. Hence b_i must be the largest value of the i^{th} coordinate of all labels in $\hat{D}_N(M_0)$.

QED

From Lemma 3.6 and Theorems 3.2 and 3.3 we can conclude, without further proof:

Theorem 3.4:

(a) It is decidable whether a given submarking is coverable in a given Petri Net with a given initial marking.

(b) It is decidable whether a given place is bounded in a given Petri Net with a given initial marking.

The following corollary states some consequences of Theorem 3.4 which are easy to prove:

Corollary 3.1:

(a) Potential firability is decidable.

(b) t-deadness is decidable.

(c) It is decidable whether a given transition can fire arbitrarily many times (infinite firability).

(d) It is decidable whether a given place p_i will ever receive a token.

Proof:

(a) Potential firability of transition t at marking M is equivalent to the coverability of F(t) in $R_N(M)$; see the observation following Definition 2.9.

(b) t-deadness of M is the negation of (a).

(c) If we attach an extra output place p' to t to count the number of firings, we only have to check the boundedness of p'.

(d) This is equivalent to whether the marking whose i^{th} coordinate is 1 and all other coordinates are zero is coverable.

QED

CHAPTER 4

REACHABILITY PROBLEMS

4.1 Reachability of a Given Marking or Submarking

The decidability of the Reachability Problem is probably the most important open problem in the mathematical theory of Petri Nets and related formalisms. In the Introduction we saw how it relates to similar unsolved problems in other theories. In this chapter we exhibit a number of recursively equivalent formulations of the Reachability Problem.

Given a Petri Net $N = \langle \Pi, \Sigma, F, B, M_0 \rangle$ with places $\Pi = \{p_1 \ldots p_r\}$ and transitions $\Sigma = \{t_1 \ldots t_s\}$, these various formulations are:

The Reachability Problem (RP): Given $M \in \mathbb{N}^r$, is $M \in R_N(M_0)$?

The Submarking Reachability Problem (SRP): Given $P \subseteq \Pi$ and $M/P \in (\mathbb{N} \cup \{\omega\})^r$, does there exist an $M' \in R_N(M_0)$ such that $M/P \approx M'$?

The Zero Reachability Problem (ZRP): Is $0 \in R_N(M_0)$?

The Single-Place Zero Reachability Problem (SPZRP): Given a place $p \in \Pi$, does there exist an $M \in R_N(M_0)$ such that $M(p) = 0$?

Since RP and SPZRP are instances of SRP and ZRP is an instance of RP, it is sufficient to close the circle of reducibilities by showing that SRP is reducible to ZRP, and that ZRP is reducible to SPZRP.

Lemma 4.1:

SRP is reducible to ZRP.

Proof:

We are given a Petri Net N and a submarking M/P over a subset of the places $P \subseteq \Pi$.

Let us add a "run" place p_0 to N; p_0 contains one token and self-

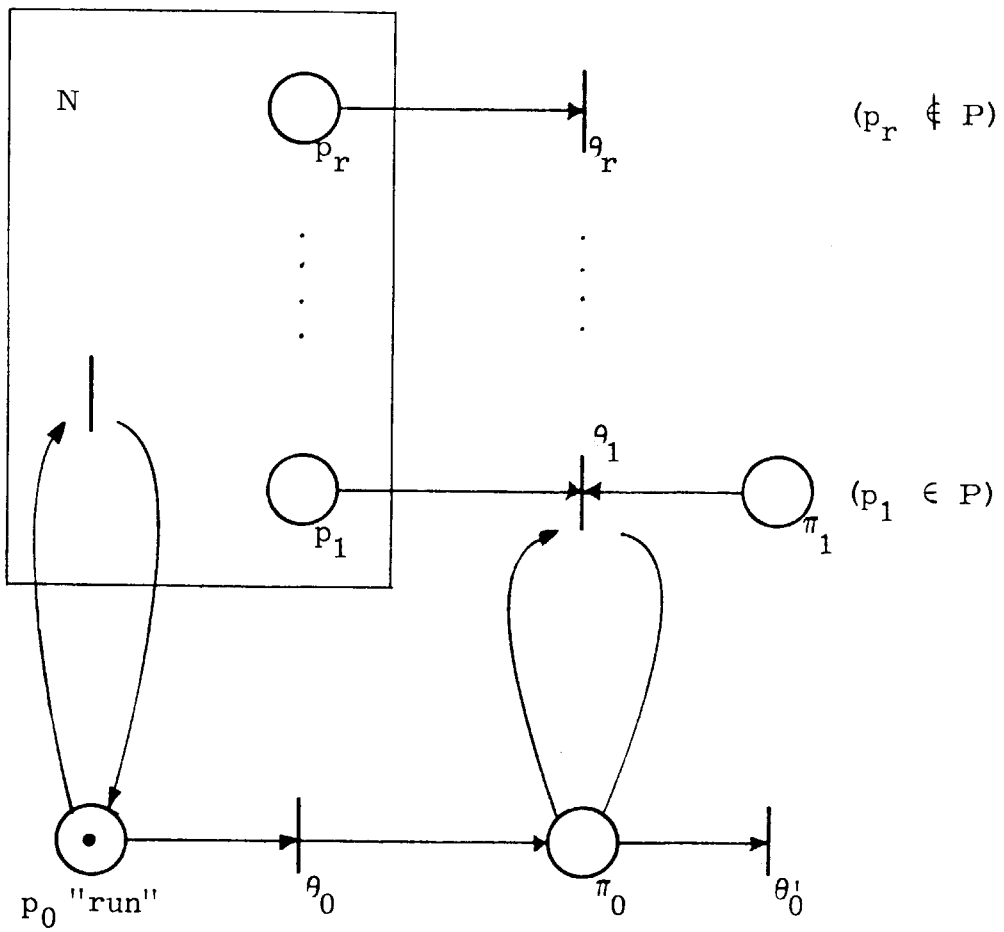

N
p_r
ϑ_r
$(p_r \notin P)$
ϑ_1
p_1
π_1
$(p_1 \in P)$
p_0 "run"
θ_0
π_0
θ_0'

Figure 4.1

loops on every transition of N. (See Figure 4.1.)

For every place $p_i \in \Pi$ we add a transition θ_i which receives a single arc from p_i. A transition named θ_0 transfers a token from p_0 to a new place π_0 which self-loops on every θ_i, $1 \leq i \leq r$, and a transition θ_0' removes a token from π_0.

For every place $p_i \in P$ we include a new place π_i, originally marked with $M_{/P}(i)$ tokens. Each place π_i sends a single arc to θ_i.

Now the only way the augmented Net can reach the zero marking is if all π_i places are emptied. This requires first reaching some marking M' in N, then firing θ_0 into π_0. At this point, we can empty all places in Π-P since the corresponding θ_i transitions are not further constrained. But for $p_i \in P$, θ_i can empty both p_i and π_i if and only if $M'(p_i) = M_{/P}(\pi_i)$; if either p_i or π_i contains more tokens, it cannot be emptied.

The last firing is that of θ_0', and the zero marking could have been reached if and only if $M' \approx M_{/P}$. Therefore, a test for ZRP of the augmented Net can decide SRP for M_P in N.

QED

Lemma 4.2:

ZRP is reducible to SPZRP.

Proof:

We want to check whether the zero marking is reachable in Petri Net N.

Let us add to N a new place π such that, at all times, π contains as many tokens as there are in all places of N, i.e. at every marking M:

$$M(\pi) = \sum_{i=1}^{r} M(p_i)$$

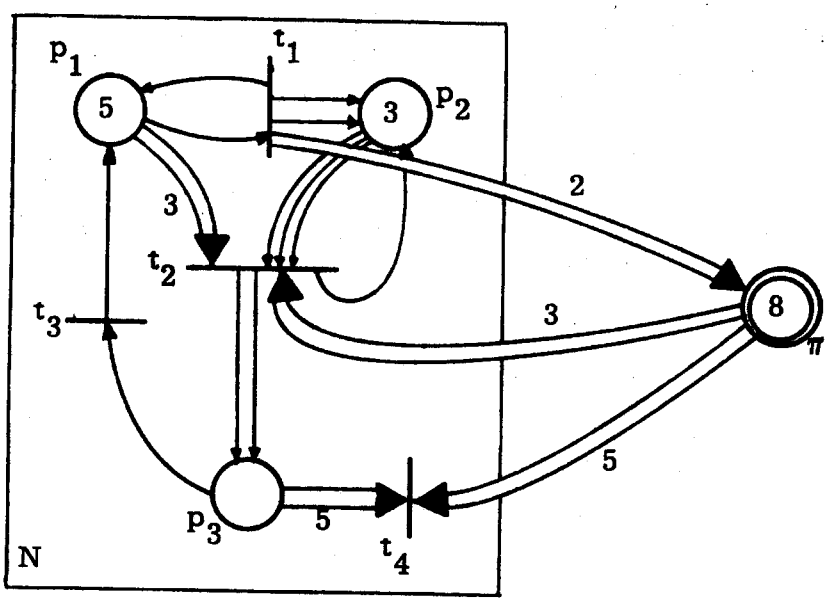


Figure 4.2

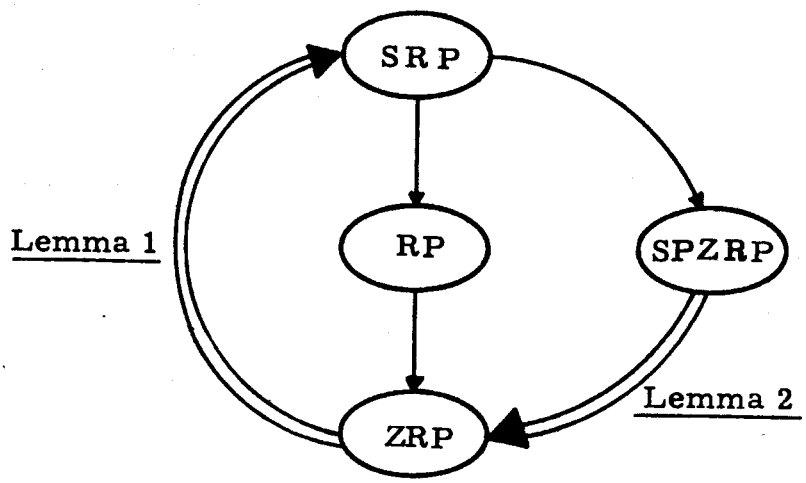


Figure 4.3

In particular, at the initial marking π contains $\sum_{i=1}^{r} M_0(p_i)$ tokens.

Let $\Delta_j = \sum_{i=1}^{r} (B(p_i, t_j) - F(p_i, t_j))$ be the change in the total number of tokens in N for one firing of transition t_j. We simply connect t_j to π by a bundle of thickness Δ_j such that:

$$\Delta_j \geq 0 \Rightarrow F(\pi, t_j) = 0 \;\&\; B(\pi, t_j) = \Delta_j$$

$$\Delta_j < 0 \Rightarrow F(\pi, t_j) = -\Delta_j \;\&\; B(\pi, t_j) = 0$$

Then the change to $M(\pi)$ is also Δ_j. Moreover, if t_j is firable at M in N, then it is also firable in N augmented by π, since $M(\pi)$ must exceed the sum $\sum_i F(p_i, t_j)$, which is greater than $F(\pi, t_j)$.

Now $M = 0$ iff $M(\pi) = 0$, so that a test for SPZRP of π in the augmented Net decides ZRP for N.

QED

Figure 4.2 shows an example of this construction.

From the obvious reducibilities and the two Lemmas we conclude:

Theorem 4.1:

RP, SRP, ZRP and SPZRP are all recursively equivalent to each other.

Figure 4.3 shows the circle of reducibilities. A thin arrow indicates the reducibility of a problem to a more general problem of which it is an instance.

4.2 Reachability of Some Marking in a Given Set of Markings

In some cases, such as in the investigation of Liveness in the next chapter, we would like to test whether at least one marking in a given set of markings is reachable. If the set is finite, this involves just a finite number of applications of RP, but if the set is infinite, we

have to use a different approach.

We have already encountered Reachability Problems of this kind. The SRP asks whether there exists a reachable marking in the set $\{M \in \mathbb{N}^r \mid M \approx V\}$ of all markings agreeing with the submarking V. The Coverability Problem is a decidable case of this kind, where we ask whether the set $\{M' \in \mathbb{N}^r \mid M' \geq M\}$ contains a reachable marking.

Such sets of markings to be tested for reachability can also be viewed as predicates, where P(M) is true of marking M iff M is a member of such a set. Thus, the predicate agrees-with-V holds for M iff $M \in \{M \in \mathbb{N}^r \mid M \approx V\}$.

Definition 4.1:

(a) A set $A \subseteq \mathbb{N}^r$ is said to be RP-solvable iff the problem of deciding, for a given Petri Net N with initial marking $M_0 \in \mathbb{N}^r$, whether there exists a reachable marking in the set A is recursively reducible to RP: $[?\ R_N(M_0) \cap A \neq \emptyset]$ is reducible to RP.

(b) A Predicate P(M) is said to be RP-solvable iff the problem of deciding, for a given Petri Net N with initial marking M_0, whether there exists a reachable marking which satisfies P is recursively reducible to RP: $[?\ \exists M \in R_N(M_0)\colon P(M)]$ is reducible to RP.

(c) This problem is called the General Reachability Problem for the Petri Net N and the Predicate P, or the set A.

The General Reachability Problem (GRP) is thus reducible to the RP by definition. The question of interest is now to exhibit a large class of RP-solvable sets and predicates.

Many sets of markings which will be of interest in later chapters can be directly proved to be RP-solvable, by showing a suitable construction, usually very similar to the construction of Figure 4.1. Examples are the set of markings covered by a given submarking (used in the proof of Theorem 5.1), or the set of markings not exceeding a given marking (used in the proof of Theorem 5.3).

But we shall use a more general approach and show that, among others, all semilinear sets (the two examples above are semilinear) are RP-solvable.

Lemma 4.3:

Every Reachability Set is RP-solvable.

Proof:

Let $R_N(M_0) \subseteq \mathbb{N}^r$ be the Reachability Set of a given Petri Net N with initial marking M_0. We have to show that for every other Petri Net of r places, say N' with initial marking M'_0, we can decide whether $R_{N'}(M'_0) \cap R_N(M_0) \neq \emptyset$ if we can decide RP or, in this proof, ZRP.

Given copies of the two nets N and N' with their respective initial markings, we construct a new net N'' as shown in Figure 4.4 (compare Figure 4.1!): Each component, N and N', has its "run" place, p_0 respectively p'_0. There is an extra place π which receives a token from transition θ_0; this transition removes both "run" tokens. The set of transitions θ_i, $1 \leq i \leq r$, matches the markings reached in N and N' token by token; it self-loops on place π. Finally, θ'_0 removes the token from π. It is easy to see that this new net N'' can reach the zero marking iff some marking can be reached in both N and N', so

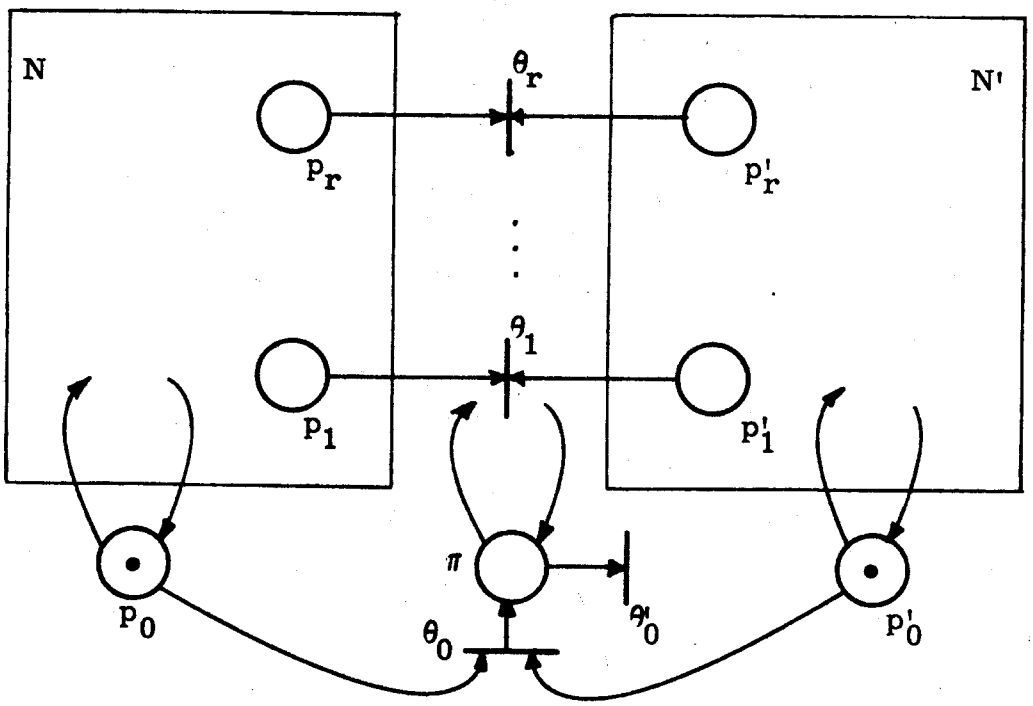


Figure 4.4

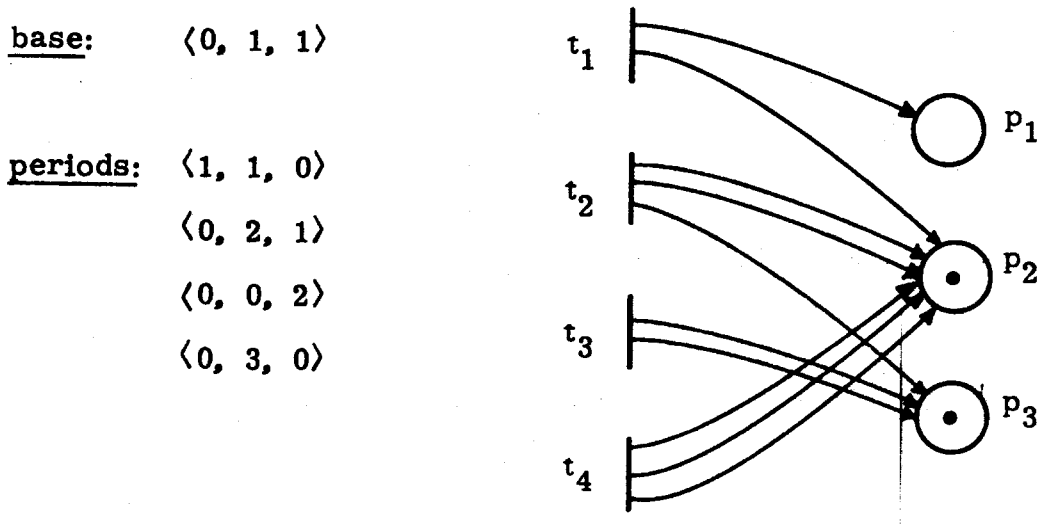


Figure 4.5

that the transitions θ_i can let the marking in N exactly cancel the marking in N'.

QED

This Lemma involves the Common Marking Problem: Does there exist a marking common to two Reachability Sets?

Corollary 4.1:

The Common Marking Problem is recursively equivalent to the Reachability Problem.

Proof:

Lemma 4.3 shows reducibility in one direction. For the other direction, let one net be a net without transitions. Its Reachability Set is then a singleton set, consisting only of the initial marking. Then RP is an instance of the Common Marking Problem.

QED

Lemma 4.4:

Every Linear Set in $\mathbb{N}^r$ is a Reachability Set.

Proof:

Recall that a Linear Set $A \subseteq \mathbb{N}^r$ can be defined by a vector $V_0 \in \mathbb{N}^r$ (the base) and a non-negative r x s matrix B (whose s column vectors are the periods) by:

$$A = \{V \in \mathbb{N}^r \mid \exists X \in \mathbb{N}^s : V = V_0 + B \cdot X\}$$

This also precisely defines the Reachability Set of a Petri Net $N = \langle \{p_1 \ldots p_r\}, \{t_1, \ldots, t_s\}, F, B, V_0 \} \rangle$ where F is identically zero (every transition has zero input places) and each transition t_j corresponds to a period, viz. the j^{th} column of matrix B.

Figure 4.5 shows an example.

QED

Lemma 4.5:

The finite union of RP-solvable sets (of same dimension) is an RP-solvable set.

Proof:

Let $A_1, \ldots, A_n$ be a finite collection of RP-solvable sets (of same dimension), and let $A = \bigcup_{i=1}^{n} A_i$ be their union. Then the GRP for a given Petri Net N and the set A is decided in the affirmative iff for some i, $1 \leq i \leq n$, the GRP for the net N and the set A_i is decided in the affirmative. If A contains a reachable marking $M \in R_N$, then A_i must contain that marking.

QED

Recall that a semilinear set is the finite union of linear sets. Hence:

Theorem 4.2:

Every semilinear set is RP-solvable.

This theorem is especially important because semilinear sets are closed under union, intersection and complementation (Theorem 2.9). Thus, if we define a semilinear predicate over $\mathbb{N}^r$ as a predicate whose Truth domain is a semilinear set, then every proposition involving semilinear predicates of the same argument is a semilinear predicate of that argument, and thus RP-solvable.

The following corollary lists a number of semilinear sets:

Corollary 4.2:

The following sets are RP-solvable:

(a) Given matrices A (t × r), B (t × s), C (t × 1) over $\mathbb{Z}$:

$$\{V \in \mathbb{N}^r \mid \exists X \in \mathbb{N}^s : A \cdot V + B \cdot X = C\}$$

(solutions to linear diophantine equations with dummy variables)

(b) Given vectors $V_1, \ldots V_n \in \Omega^r$:

$$\{V \in \mathbb{N}^r \mid \exists i,\ 1 \leq i \leq n;\ V \leq V_i\}$$

(c) Given a vector $W \in \mathbb{N}^r$:

$$\{V \in \mathbb{N}^r \mid V \neq W\}$$

Proof:

(a) The solution space to a set of linear diophantine equations with dummy variables is semilinear (Theorem 2.10). See, for example, Ginsburg and Spanier [14] or Van Leeuwen [63].

(b) This is a finite union of instances of (a), where A = B = I, the identity matrix, and C = V_i.

(c) This is the complement of an instance of (a), where A = -B = I and C = W.

QED

As an exercise, the reader may wish to prove RP-solvability of these three sets directly, by adding the appropriate mechanisms to the construction of Figure 4.1. These constructions are much simpler than trying to find a semilinear representation of the sets and then using Lemmas 4.3, 4.4 and 4.5.

Remark:

Semilinear Sets correspond exactly to Predicates expressible in Presburger arithmetic (Ginsburg and Spanier [14]).

CHAPTER 5

LIVENESS AND PERSISTENCE

5.1 Liveness

The decision problems discussed in this section are:

The Liveness Problem (LP): Given a Petri Net N with an initial marking M_0, is N live at M_0, i.e. is every transition live at M_0?

The Sub-Liveness Problem (SLP): Given a Petri Net, an initial marking M_0, and a transition t of the net, is t live at M_0 in N?

Let us recall that a transition is live at M_0 iff no t-dead marking is reachable, where a marking M is said to be t-dead iff no firing sequence starting at M can ever fire t, or alternatively, if t is not potentially firable at M (see Definitions 2.9, 2.10 and 2.11).

Thus SLP appears to be an instance of the General Reachability Problem applied to the set of t-dead markings, if we can show that this set is RP-solvable.

Let D_t be the set of t-dead markings of a given Petri Net:

$$D_t = \{M \in \mathbb{N}^r \mid t \text{ is not potentially firable at } M\}$$

The most important property of this set is its monotonicity (Definition 2.27):

Lemma 5.1

The set D_t of t-dead markings of a given Petri Net is monotone:

$$(M' \leq M \;\&\; M \in D_t) \Rightarrow M' \in D_t$$

Proof:

Suppose M' is not t-dead, i.e. there exists a firing sequence starting at M' which fires t. By the containment property (Theorem 2.1), this firing sequence is also firable at the larger marking $M \geq M'$. But this contradicts the assumption of t-deadness of M.

QED

From Theorem 2.11(a) we conclude that D_t, being monotone, must be semilinear. And if the chain-completion D_t^c (see Definition 2.29) is effectively recursive, i.e. if, given a Petri Net, we can decide membership in D_t^c, then D_t is effectively semilinear.

From Theorem 2.7 we get the following characterization of the chain-completion of D_t:

$$D_t^c = \{V \in \Omega^r \mid \forall M \in \mathbb{N}^r: M \approx V \Rightarrow M \in D_t\}$$

If we compare this characterization with the definition of a t-dead submarking (Definition 2.21) we conclude that:

$$D_t^c = \{V \in \Omega^r \mid V \text{ is t-dead}\}$$

Thus, the chain-completion of the set of t-dead markings is simply the set of t-dead submarkings.† All that remains to be proved is:

Lemma 5.2:

It is decidable whether a given submarking V is t-dead, for a

† In general, Theorem 2.7 implies that if A is a set of markings having a certain property $\mathcal{P}$, then its chain completion is the set of submarkings having the property $\mathcal{P}'$ which is the strong extension of property $\mathcal{P}$. We encountered a similar situation in Chapter 3, for the property of coverability.

given transition t in a given Petri Net N.

Proof:

Let P be the support P(V) of submarking V, i.e. the set of places on which it is defined (finite coordinates of V). Then V is t-dead in N iff V is t-dead as a marking of the subnet N_P Indeed, for any firing sequence starting at V in the subnet N_P we can find a marking $M \approx V$ at which the same firing sequence is firable (Theorem 2.2(b)) in the net N. Thus (V not t-dead in N_P) $\Rightarrow$ (M not t-dead in N) $\Rightarrow$ (V not t-dead in N by definition). And if no firing sequence involving t is possible from V in the subnet, then a fortiori no such firing sequence is possible in N at any $M \approx V$.

But now Corollary 3.1 says that the t-deadness of V in N_P is decidable. Hence the t-deadness of V in N is decidable.

QED

Now we can assert:

Theorem 5.1:

Liveness (both LP and SLP) is recursively reducible to Reachability.

Proof:

LP is a finite number of instances of SLP, one per transition.

Since the set of t-dead markings D_t is monotone (Lemma 5.1) and its chain-completion, the set of t-dead submarkings D_t^c, is effectively recursive (Lemma 5.2), D_t is effectively semilinear, by Theorem 2.11(b), and hence RP-solvable, by Theorem 4.2. This means that

the question of deciding whether some t-dead marking $M \in D_t$ is reachable, i.e. the SLP, is recursively reducible to the Reachability Problem (Definition 4.1).

QED

We should point out, however, that the reliance on the semilinearity of D_t may be considered overkill. The characterization of D_t given by Theorem 2.8, on which the claim of semilinearity is based, is in terms of the finite set $\widehat{D_t^c}$ of maximal elements of D_t^c. We may call this the set of maximal t-dead submarkings:

$$\widehat{D_t^c} = \{V_1, \ldots, V_k \mid 1 \leq i \leq k\text{: } V_i \text{ is a maximal element of } D_t^c\}$$

Then we have: $D_t = \{M \in \mathbb{N}^r \mid M \leq V_1 \text{ or } \ldots \text{ or } M \leq V_k\}$. Now a simple modification of the construction in Figure 4.1 can be used to reduce reachability of some marking $M \leq V_i$ to reachability of zero, and thus reduce SLP to k instances of ZRP applied to this construction, once for each maximal t-dead submarking V_i, $1 \leq i \leq k$. We leave the details as an exercise for the reader.

Now we shall prove that the converse reducibility also holds.

Theorem 5.2:

(a) Reachability is recursively reducible to Liveness.

(b) Reachability and Liveness are recursively equivalent.

Proof:

(a) We shall reduce the Single-Place Zero-Reachability Problem (SPZRP) to the LP. This is sufficient in view of the equivalence

of RP and SPZRP, from Theorem 4.1. Let N be a Petri Net in which we wish to test whether a given place p_i can ever become empty, for a given initial marking.

As shown in Figure 5.1, we construct a new net $\tilde{N}$ by adding to a copy of N the following:

- a "run" place p_0 which self-loops on every transition of N.
- a transition θ_0 which may remove the token initially present in p_0.
- a transition θ_1 which transfers a token from the test place p_i to a new place π.
- a transition θ_2 which self-loops on π and deposits tokens on all places of the net, including p_0 and p_i.

The operation of $\tilde{N}$ is as follows. As long as neither θ_0 nor θ_1 has fired, it behaves exactly like N. If, at any time, we fire θ_0 before having fired θ_1, then the whole net $\tilde{N}$ is frozen dead unless p_i contains at least one token, which may fire θ_1.

If, at any time whatsoever, we fire θ_1, we place a token on π which cannot disappear. Now θ_2 is permanently firable, and can generate enough tokens to fire any arbitrary firing sequence. It follows that any killing sequence for $\tilde{N}$ must end at a marking where p_i is unmarked. Conversely, if such a marking is reachable by a firing sequence σ, then $\sigma\theta_0$ is a killing sequence. Thus $\tilde{N}$ is live iff place p_i cannot become unmarked in N.

(b) This follows from (a) and from Theorem 5.1.

QED

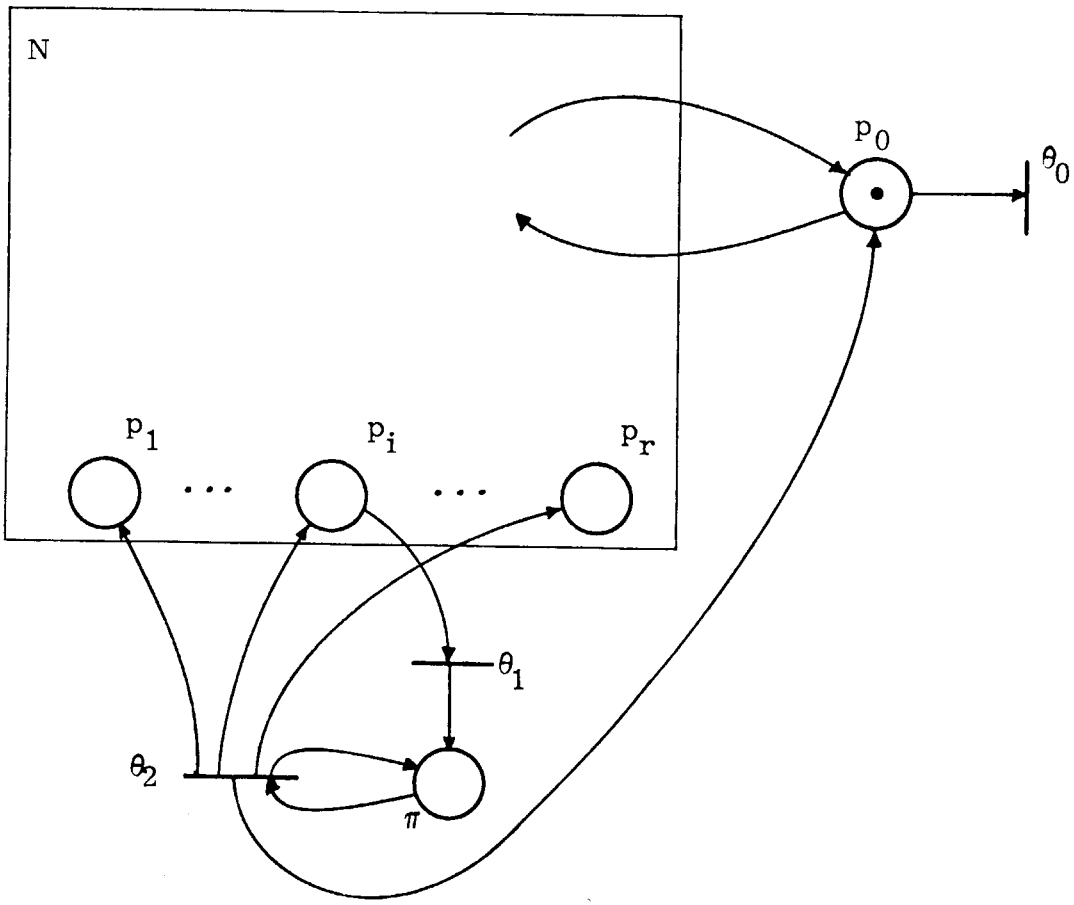


Figure 5.1

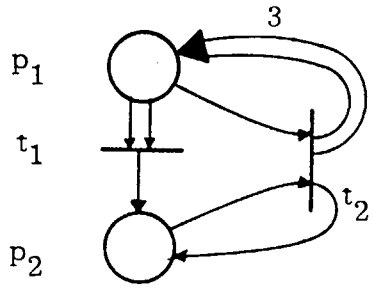


Figure 5.2

Corollary 5.1

The LP and the SLP are recursively reducible to each other.

Proof:

The LP is a finite number of instances of SLP, one for each transition. On the other hand, SLP is reducible to RP by Theorem 5.1, which is in turn reducible to LP by Theorem 5.2(a). This is why Theorem 5.2(b) simply states equivalence between Liveness (LP and SLP) and Reachability (RP, SRP, ZRP, SPZRP $\cdots$).

QED

In Hack [20], we give a direct proof of the reducibility of SLP to LP. Essentially, we show that in a Petri Net, any transition can be replaced by a construction in which every transition is live by construction, and such that this modified Net behaves exactly like the original Net. The trick is that some specific patterns of firings of the new transitions have an effect on the rest of the Net identical to the firing of the old transition, whereas other patterns have a zero effect on the rest of the Net. Then we test the liveness of a particular transition t by replacing all other transitions by such guaranteed live constructions. The resulting Net will be live iff the remaining original transition is live, and thus we test the liveness of this transition by testing the liveness of the whole new Net. The construction increases the size of the Net by a small linear factor (~ 6).

An interesting corollary of this is that any non-live Petri Net can be simulated in this way by a live Petri Net.

Historical Note:

As early as 1970 (R. C. Holt, [29]), it has been conjectured that Liveness was reducible to Reachability. Keller investigated the problem in his 1972 report [34]. He observed the decidability of potential firability (which he called "pseudo-liveness"), as well as the (reverse) monotonicity of the set of markings at which transitions are potentially firable, and he guessed (correctly) that this property would be useful in reducing liveness to reachability.

Our breakthrough (in 1973) was the realization that the possibly infinite set of t-dead markings (at which t is not potentially firable) could be described by a finite number of t-dead submarkings, thus reducing the SLP to a finite number of instances of the SRP (Hack, [20, 21]). It was from that proof that we subsequently abstracted the properties of monotone sets and their chain-completions described in section 2.6. The separation of these lattice-theoretic aspects from the Petri Net aspects of the proof, and the introduction of the General Reachability Problem, considerably simplified the proof.

The following example illustrates the use of t-dead submarkings. When we say that a submarking V is t-dead, we essentially say that the potential firability of transition t depends only on the marking of a certain subset of the places, namely the support of V. If this submarking is too small, then t will never be firable regardless of how large the marking of the other places is.

In the net of Figure 5.2, if p_1 is blank, no amount of tokens will make t_2 potentially firable; if p_2 is blank, it must receive a token via a firing

of t_1, to fire t_2, and therefore we can see that the only t_2-dead markings are $\langle 1, 0\rangle$, $\langle 2, 0\rangle$, and all markings of the form $\langle 0, x\rangle$, where $x \in \mathbb{N}$. But these markings $\langle 0, x\rangle$ are precisely all markings which agree with the submarking $p_1 = 0$, which we write as $\langle 0, \omega\rangle$, and two markings $\langle 1, 0\rangle$ and $\langle 2, 0\rangle$. As it turns out, neither of the two markings $\langle 1, 0\rangle$ and $\langle 2, 0\rangle$ is reachable, since if t_1 does not fire, there will always be more than 4 tokens in p_1, and after t_1 fires, p_2 will always contain at least one token. The submarking $\langle 0, \omega\rangle$ is also not reachable since no firing of t_1 or t_2 changes the parity of the marking in p_1. Since $M_0(p_1)$ is odd, we cannot reach a marking with zero tokens in p_1. The conclusion is that t_2 is live at $M_0 = \langle 5, 0\rangle$.

5.2 Persistence

As in the case of Liveness, there are essentially two decision problems to consider:

The Persistence Problem (PP): Is a given Petri Net with a given initial marking persistent?

The Sub-Persistence Problem (SPP): Is a given transition t persistent in a given Petri Net at a given initial marking?

And since a Net is persistent iff every transition is persistent, it is clear that the PP is just a finite number of instances of the SPP, one for each transition.

But in contrast to the previous section, we have not been able to reduce the SPP to the PP. This is because persistent Nets have special properties which restrict their generality in a significant way, whereas live Nets can "simulate" arbitrary Nets as indicated at the end of the previous section. In particular, Keller [35] has shown that Liveness is decidable for persistent Nets, and we have some evidence that the

Reachability Sets of persistent Nets are effectively semilinear*, and that persistence of a Petri Net is in fact decidable.

In this section we shall show that the SPP is recursively equivalent to the RP. We do in fact conjecture that the RP is decidable (see Chapter 11), but our conjecture for the decidability of the PP is totally independent of the RP, and is, in our opinion, also closer to being settled.

Let us recall that a transition is persistent in $R_N(M_0)$ iff:

$$\forall t' \neq t;\ \forall M \in R_N(M_0): \left[M \geq F(t) \ \ \& \ \ M \geq F(t') \Rightarrow M \geq F(t) + F(t')\right]$$

This can be rewritten as:

$$t \ \underline{\text{not persistent in}}\ R_N(M_0) \Leftrightarrow \exists M \in R_N(M_0) \cap A$$

where

$$A = \bigcup_{t' \neq t} (\{M \mid M \geq F(t)\} \cap \{M \mid M \geq F(t')\} \cap \{M \mid M \not\geq F(t) + F(t')\})$$

In other words, A is a semilinear set (see corollary 4.2 and Theorem 2.9) and RP-solvable, by Theorem 4.3. It follows that t-persistence is reducible to the General Reachability Problem:

Theorem 5.3:

Persistence (both PP and SPP) is recursively reducible to Reachability.

---

* The semilinearity of persistent Reachability Sets has recently been proved by Landweber and Robertson in "Properties of Conflict-Free and Persistent Petri Nets", TR #264, Computer Science Dept., U. of Wisconsin, Dec. 1975.

It should be observed that the related problem of whether a given transition can ever disable another transition can similarly be reduced to the GRP.

Now we shall show that the reverse reducibility also holds for the SPP, i. e. persistence of a given transition.

Theorem 5. 4:

(a) Reachability is recursively reducible to the Persistence of a given transition (SPP).

(b) The SPP is recursively equivalent to the RP.

Proof:

(a) We shall reduce the SPZRP to the SPP. Let N be a Petri Net (with its initial marking) in which we wish to test whether a given place, say p_1, can ever become unmarked. The construction required is quite trivial: We simply add a transition θ_0 which self-loops on the place to be tested for zero, i. e. p_1. If p_1 is initially unmarked, the SPZRP is trivially affirmed. Otherwise, θ_0 is enabled as long as p_1 is marked, and can only be disabled if some other transition eventually removes the last token from p_1. Then θ_0 is persistent iff p_1 cannot become unmarked.

(b) This follows from (a) and Theorem 5. 3.

QED

CHAPTER 6

UNDECIDABILITY AND WEAK COMPUTATION

6.1 The First Undecidability Proofs for Vector Addition Systems and Petri Nets

When Vector Addition Systems were first developed, it was believed that all Reachability Sets would be semilinear. Because of the connection between semilinear sets and Presburger Arithmetic, a decidable first-order theory, most questions about Vector Addition Systems and Petri Nets would then be decidable, including the Reachability Problem (still open) and the Inclusion Problem (in fact undecidable). But in 1967 M. Rabin [56] showed that this is not the case: he exhibited a non-semilinear Reachability Set, and showed that the problem of deciding whether one Reachability Set is a subset of another Reachability Set (the Inclusion Problem) was undecidable, by reducing the unsolvable problem of finding the roots of exponential diophantine equations to it. In 1970 the corresponding problem for diophantine polynomial equations (Hilbert's Tenth Problem) was shown to be undecidable, and Rabin presented a new proof of his Theorem in a talk at MIT in 1972. Rabin never published his proof, but an account of his 1972 talk can be found in Baker [4]. We presented a Petri Net version of this proof in Hack [20] and, on the occasion of publishing our proof of the undecidability of the Equality Problem for Reachability Sets (Hack [23]), we broke Rabin's proof down into several relatively independent steps, each of which may be interesting in its own right. This is also our approach in this and the following chapter.

6.2 Diophantine Polynomials and Hilbert's Tenth Problem

Hilbert's Tenth Problem can be stated as follows:

> Given a polynomial of several variables $P(x_1 \ldots x_n)$ with integer coefficients, does it have an integral root, i.e. does there exist a vector $\langle x_1 \ldots x_n \rangle \in \mathbb{Z}^n$ such that $P(x_1 \ldots x_n) = 0$?

It is one of 23 mathematical problems that D. Hilbert [26] proposed to mathematicians at a congress in 1900. Many of these were subsequently solved or proved undecidable, and the Tenth, despite its very simple formulation, was one of the toughest. In the U.S.A., Davis, Putnam and Robinson [10] showed that the corresponding problem for exponential polynomials (with variables allowed as exponents) was undecidable, and that if the integral roots of ordinary polynomials could grow like an exponential function of the coefficients Hilbert's Tenth Problem would also be undecidable. In the USSR, number theorists had been aware of such properties of the integral roots of polynomials quite early, but only in 1970 did Yu. Matijas'evič [40] bring the two lines of inquiry together and thus demonstrated the undecidability of Hilbert's Tenth Problem.

For our purposes, we prefer to restrict our attention to the non-negative integers.

Definition 6.1:

> A diophantine polynomial $P(x_1 \ldots x_n)$ is a polynomial of several variables with non-negative integer coefficients.

Definition 6.2:

> The graph of a diophantine polynomial $P(x_1 \ldots x_n)$ is the set:

$$G(P) = \{\langle x_1, \ldots, x_n, y\rangle \in \mathbb{N}^{n+1} \mid y \leq P(x_1 \ldots x_n)\}$$

The version of Hilbert's Tenth Problem we shall use in our undecidability proofs is what we call the Polynomial Graph Inclusion Problem (PGIP):

Given two diophantine polynomials P and Q with the same number of variables, do we have $G(P) \subseteq G(Q)$?

Theorem 6.1:

The Polynomial Graph Inclusion Problem is recursively undecidable.

Proof:

We shall reduce the undecidable Hilbert's Tenth Problem to the Polynomial Graph Inclusion Problem.

(a) We can restrict the arguments of the polynomials to the non-negative integers. Indeed, $P(x_1, \ldots, x_n) = 0$ has a solution in $\mathbb{Z}$ if and only if one of the 2^n polynomials obtained by replacing some variables by their negative has a solution in $\mathbb{N}$.

(b) Any root of $P(x_1, \ldots, x_n)$ is also a root of $P^2(x_1, \ldots, x_n)$, and vice versa. Hence we can restrict our attention to polynomials whose range is in $\mathbb{N}$.

(c) By separating the positive and the negative coefficients of a polynomial whose range is non-negative, we get two polynomials $Q_1(x_1, \ldots, x_n)$ and $Q_2(x_1, \ldots, x_n)$, each with non-negative integer coefficients, such that:

$$\forall x_1, \ldots, x_n \in \mathbb{N}: \; Q_1(x_1, \ldots, x_n) \geq Q_2(x_1, \ldots, x_n)$$

There exists an integral root to the original polynomial if and only if

$$\exists x_1, \ldots, x_n \in \mathbb{N}: \; Q_1(x_1, \ldots, x_n) = Q_2(x_1, \ldots, x_n).$$

Now let us consider the following two polynomial graphs:

$$G(Q_1) = \{\langle x_1, \ldots, x_n, y\rangle \in \mathbb{N}^{n+1} \mid y \leq Q_1(x_1, \ldots, x_n)\}$$

$$G(Q_2 + 1) = \{\langle x_1, \ldots, x_n, y\rangle \in \mathbb{N}^{n+1} \mid y \leq 1 + Q_2(x_1, \ldots, x_n)\}$$

From this it follows that:

$$G(Q_2 + 1) \subseteq G(Q_1) \Leftrightarrow [\forall x_1, \ldots, x_n, y \in \mathbb{N}:$$
$$(y \leq Q_2(x_1, \ldots, x_n) + 1 \Rightarrow y \leq Q_1(x_1, \ldots, x_n))]$$
$$\Leftrightarrow \nexists x_1, \ldots, x_n, y \in \mathbb{N}:$$
$$Q_1(x_1, \ldots, x_n) < y \leq 1 + Q_2(x_1, \ldots, x_n)$$

Combining this with the fact that Q_2 never exceeds Q_1, this implies:

$$G(Q_2 + 1) \subseteq G(Q_1) \Leftrightarrow \nexists x_1, \ldots, x_n, y \in \mathbb{N}:$$
$$y = 1 + Q_1(x_1, \ldots, x_n) = 1 + Q_2(x_1, \ldots, x_n)$$

In other words, Hilbert's Tenth Problem is decided in the negative if and only if the corresponding PGIP is decided in the affirmative, thus proving the undecidability of the PGIP.

QED

Remark:

The Polynomial Graph Equality Problem (PGEP) is clearly decidable, because two polynomial graphs are equal iff the two diophantine polynomials take the same value for every argument, which is possible if and only if the two polynomials are in fact the

same polynomial. We have thus a striking example of a family of sets where equality is decidable, but inclusion is not.

It is also not difficult to prove that Hilbert's Tenth Problem is not only reducible to the PGIP, but is in fact recursively equivalent to it.

In the next section we shall show that Petri Net Reachability Sets can express polynomial graphs. Actual undecidability proofs will be presented in Chapters 7 and 10.

6.3 Weak Computation by Petri Nets

In order to relate Hilbert's Tenth Problem to Petri Nets, we must show how Petri Nets can compute polynomials, in some sense. Usually, an automaton used to compute a function is given its arguments in some form, and started in some "initial" state. If and when the automaton halts in some "final" state, we can recover the computed value, for example by reading the contents of a certain register. Such an automaton is usually thought to be deterministic, or at least functional in the sense that all halting computations produce the same result. But the non-determinism associated with the set of possible firing sequences in a Petri Net is essential to the power of Petri Nets. In fact, if we only consider Nets whose firing sequences are monogenic ("deterministic" Petri Net, where at every reachable marking only one transition is firable), then all the problems mentioned so far are decidable (the reachability sets will be ultimately periodic or finite).

So, in order to get any non-trivial functions, we have to modify our idea of a computation. Following Rabin, we shall say that a non-deterministic automaton weakly computes a function $f(x_1 \ldots x_n)$ iff the maximum output value over all computations starting with the argument

$x_1, \ldots, x_n$ is $f(x_1, \ldots, x_n)$.

This definition makes sense only if the range of output values over all computations starting with a given argument is finite. There are thus two ways in which a weakly computed function may be undefined for a given argument: If there are no computations, i.e. no "final" state is reachable, or if there are computations which produce arbitrarily large output values for a given argument.

In this chapter we shall make the further assumption that every reachable state is a "final" state, so that every execution sequence (including the empty one) is a computation sequence, and every prefix of a computation sequence is also a computation sequence. We may call this the prefix interpretation.

There are several ways in which a Weak Computer can be represented in a Petri Net. The coding of the inputs is usually straightforward: A certain number of places, say $p_1 \ldots p_n$, are designated as "input" places of the net, and the initial marking is predetermined in the remaining places $p_{n+1} \ldots p_r$. The initial marking of the input places is the argument $\langle x_1, \ldots, x_n \rangle$. Every firing sequence starting from the initial marking is considered a computation.

The output of a Petri Net Weak Computer can be defined in several useful ways. In Rabin's proof (as translated into Petri Nets) and in Hack [20], the output was defined as the largest marking reached in a designated "output place". In Hack [23] it was found more convenient to use a distinguished "count" transition whose largest number of firings was defined as output. Now we wish to use the length of the longest firing sequence as output, in effect declaring every transition to be a "count" transition. The main reason is that this definition permits the

same construction to be used in proofs about Reachability Sets (Chapter 7) and in proofs about Petri Net Languages (Chapter 10). Since every transition firing counts, there is no "invisible scratchwork" in such a Weak Computer.

The class of functions weakly computable by Petri Nets may depend on the output convention. It is easy to see that the "output place" and the single "count transition" conventions are equivalent, and that every function weakly computable in the firing sequence length sense is also weakly computable in the "output place" sense (just add a new place which gets one token from every transition firing). It is not known whether every function weakly computable in the "output place" sense is also weakly computable in the "firing sequence length" sense. Because of this, we shall call a Weak Computer in the "firing sequence length" sense a λ-free Weak Computer. This terminology is borrowed from Petri Net Language theory, where a λ-transition is an "invisible" or "internal" transition whose firings do not explicitly show up in the output of the net.

We shall thus define a Petri Net Weak Computer in the λ-free prefix interpretation. Because of the containment property (Theorem 2.1) of Petri Nets, any computation with a given argument can also be carried out with any larger argument. This means that only non-decreasing functions (in every variable) can be weakly computed by a Petri Net under this interpretation.

Note:

In the remainder of this thesis, we shall interpret Petri Net Weak Computer as λ-free prefix Petri Net Weak Computer.

Definition 6.3:

A Petri Net Weak Computer (in the λ-free prefix interpretation) for a (non-decreasing) function $f = \mathbb{N}^n \to \mathbb{N}$ of n variables $x_1 \ldots x_n$ is a Petri Net with $r \geq n$ places and the following properties:

(a) The initial marking M_0 agrees with a fixed submarking $M_0/\{p_{n+1}, \ldots, p_r\}$ of the non-input places, and encodes the argument in the input places by $M_0/\{p_1, \ldots, p_n\} = \langle x_1, \ldots, x_n \rangle$.

(b) For every initial marking as described in (a), there exists a longest firing sequence of length $f(x_1, \ldots, x_n)$.

Note that there may also exist firing sequences of length shorter than $f(x_1, \ldots, x_n)$ which nevertheless cannot be continued.

Now we are going to show that diophantine polynomials are weakly computable by Petri Nets in the sense of Definition 6.3 (and hence also by the less restrictive earlier definitions of Petri Net weak computability). A polynomial $P(x_1 \ldots x_n)$ is a finite sum of monomials:

$$P(x_1 \ldots x_n) = \sum_{j=1}^{k} (M_j(x_1 \ldots x_n))$$

where each monomial is of the form:

$$M_j(x_1 \ldots x_n) = \alpha_j \cdot \prod_{i=1}^{n} (x_i^{\beta_{i,j}})$$

The α_j are positive integer coefficients and the $\beta_{i,j}$ are non-negative integer exponents. We shall first show how to compute monomials, and then how to add them together.

The basic "circuit element" will be the elementary multiplier,

illustrated in Figure 6.1.

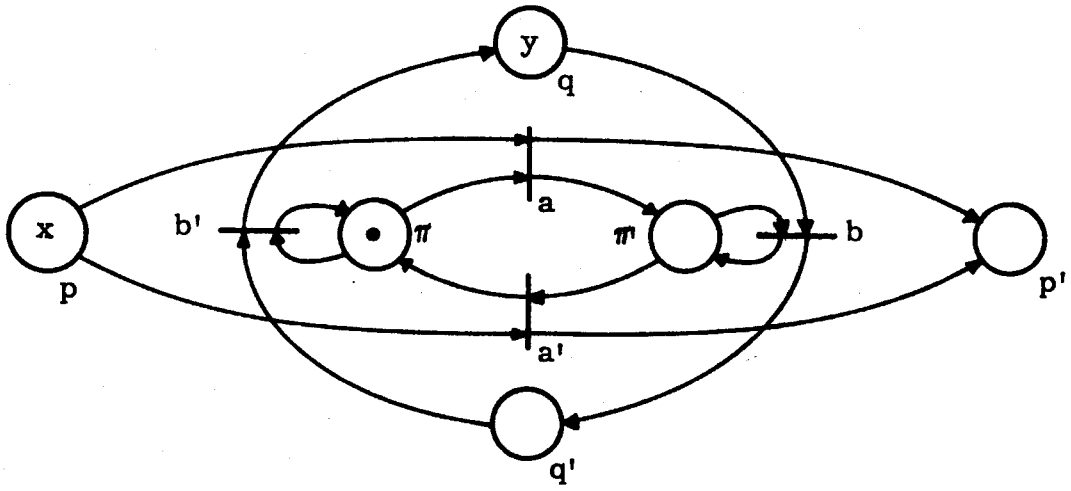


Figure 6.1

This net consists of two control places π and π', exactly one of which may contain a token initially in π. Two transitions a and a' transfer the token between π and π' and each time transfer one token from place p, which initially contains x tokens, to place p' (initially unmarked). Two transitions b and b', which self-loop on π' and π respectively, shuttle tokens between places q and q'; originally, q contains y tokens. It is easy to see that a and a' can together fire only x times, and between a and a', or a' and a, either b or b' can fire at most y times; the longest firing sequence achieves these upper bounds and fires a total of x times in {a, a'} and a total of $x \cdot y$ times in {b, b'} for a maximal firing sequence length of $x \cdot (y + 1)$; this leaves x tokens in place p'.

As used in the construction which follows, places p and q may be initially unmarked, but will receive up to x and y tokens respectively.

The maximal firing sequence is then achieved by waiting until all tokens have arrived; if firing starts before, it can only diminish the achievable sequence length, never increase it. Since we are only interested in the longest firing sequence, it will not be necessary to impose a certain sequencing on the various elementary multipliers, because the described sequencing will be maximal.

Lemma 6.1:

For each $i \in \mathbb{N}^+$, there exists a Petri Net S_i with the following properties:

(a) It is a λ-free Weak Computer (Definition 6.3) for the polynomial $(x_1 + 1)(x_2 + 1) \cdots (x_i + 1) - 1$, with input places $p_1 \cdots p_i$.

(b) It also has i "output" places $p'_1 \cdots p'_i$, initially unmarked, into which the tokens from the corresponding "input" places $p_1 \cdots p_i$ are transferred during the computation, i.e. each time a token is removed from p_j, $1 \leq j \leq i$, a token is deposited in p'_j.

Proof:

We first note that such a net has the property that after a maximal firing sequence, the argument initially in $p_1 \cdots p_i$ is now in $p'_1 \cdots p'_i$.

The proof is by induction.

basis:

The net S_1 consists of places p_1, p'_1 and transition a_1 which simply transfers tokens from p_1 to p'_1 (Figure 6.2a). For an initial marking of x_1 tokens in p_1 (and zero in p'_1) the longest firing sequence is

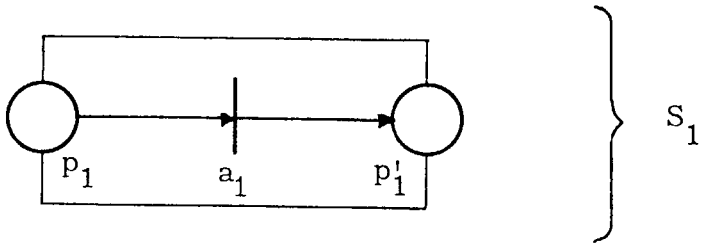


Figure 6.2a

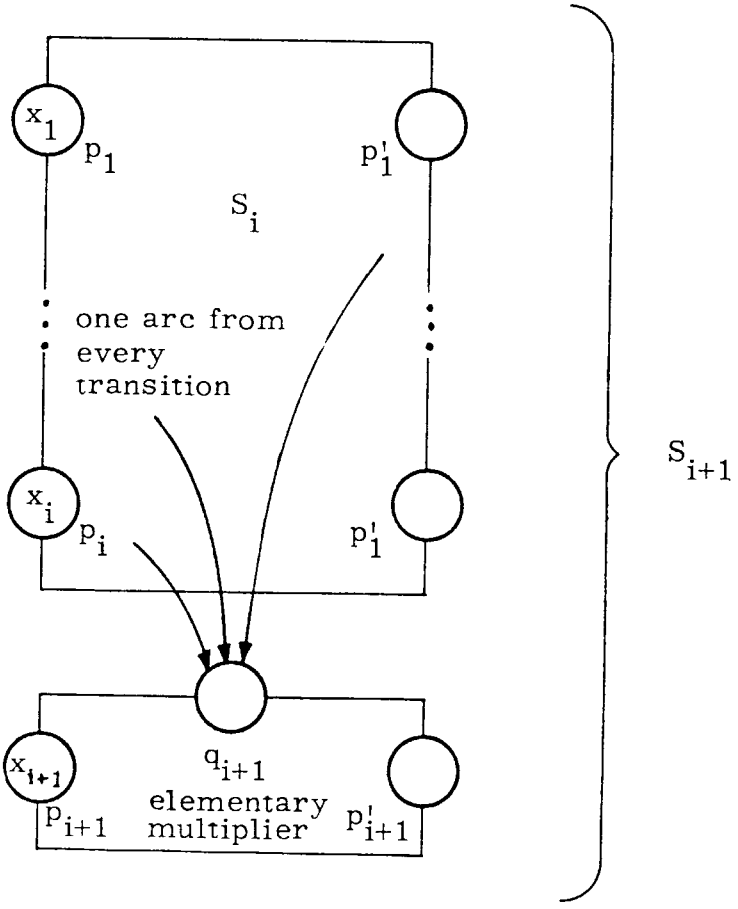


Figure 6.2b

clearly of length $(x_1 + 1) - 1$.

Inductive Step:

We are given the net S_i. We construct S_{i+1} by adding the "elementary multiplier" of Figure 6.1 with places and transitions indexed $i + 1$. Then we let every transition of S_i (i.e. transition a_1 and, for all j, $2 \leq j \leq i$, transitions a_j, a'_j, b_j and b'_j) deposit one token in place q_{i+1} (Figure 6.2b).

It is easy to see that the longest firing sequence is obtained by first firing a maximal firing sequence in S_i. This puts the largest number of tokens y into q_{i+1}, and accounts for the first y firings, where $y = (x_1 + 1)(x_2 + 1) \cdots (x_i + 1) - 1$. It also copies $x_1 \ldots x_i$ into places $p'_1 \ldots p'_i$. Then the "elementary multiplier" fires its maximal sequence, of length $x_{i+1}(y + 1)$, and transfers x_{i+1} to p'_{i+1}. The total length is thus $x_{i+1}(y + 1) + y = (x_{i+1} + 1)(y + 1) - 1 = (x_1 + 1)(x_2 + 1) \cdots (x_i + 1)(x_{i+1} + 1) - 1$.

QED

Lemma 6.2:

For every diophantine monomial, there exists a Petri Net Weak Computer for it which also copies its argument into "output" places, as in Lemma 6.1.

Proof:

Such a Weak Computer for monomial $\alpha \cdot x_1^{\beta_1} \cdot x_2^{\beta_2} \cdots x_n^{\beta_n}$ is obtained from Petri Net S_i, $i = 1 + \beta_1 + \beta_2 + \ldots + \beta_n$ by simple modifications as illustrated for the example $3 \cdot x^2 \cdot y$, where the net S_4 (Figure 6.3a) is transformed as shown in Figure 6.3b.

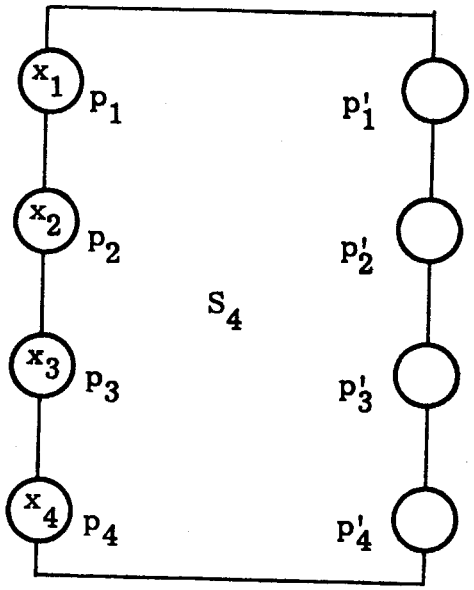


Figure 6.3a

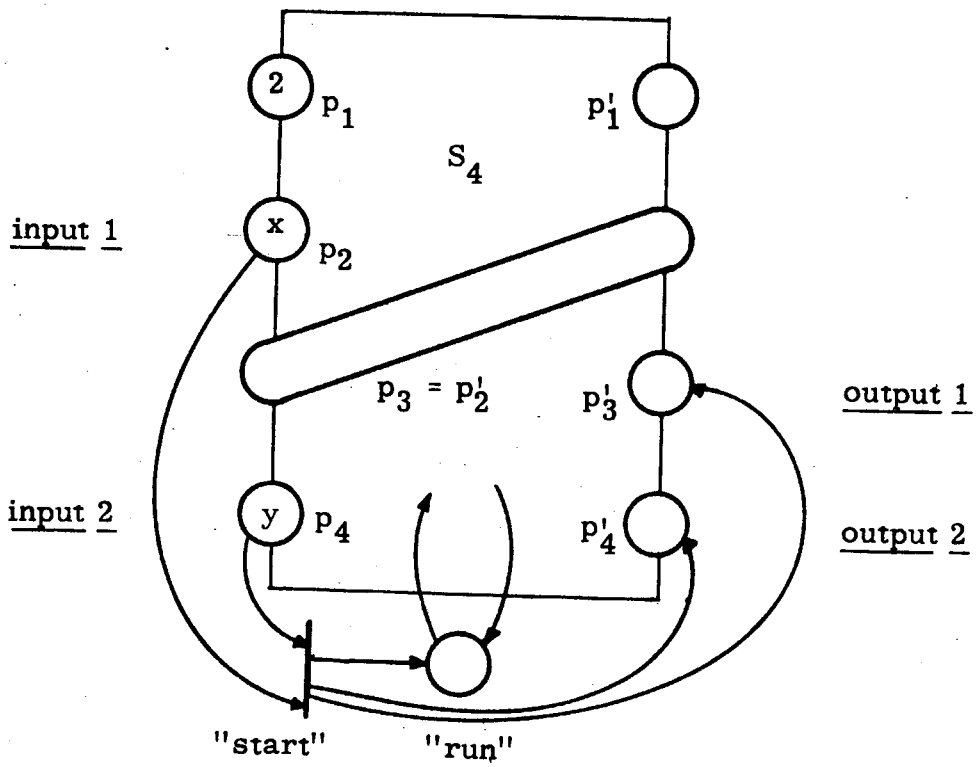


Figure 6.3b

(a) Place p_1 is marked with α-1, in this case 2 tokens.

(b) Repeated multiplication of one variable (exponentiation) is achieved by identifying the "output" place of one level (in the inductive construction of S_i) with the "input" place of the next level. In this case, p_2 is the x input; p_3 and p_2' are the same place, initially unmarked, used for multiplying again by x, and p_3' is now the output associated with x.

(c) We add a "run" place which self-loops on every transition in S_i, and a "start" transition which puts a token into the "run" place and removes a token from every p_j place used as an input for a monomial variable (in this case, p_2 for x and p_4 for y). The "start" transition also adds one token to every monomial "output" place (in this case, p_3' for x and p_4' for y), to restore the correct argument at the end of a maximal sequence.

This construction functions as follows. Recall from the proof of Lemma 6.1 that the maximal firing sequence is obtained by first consuming tokens from p_1, then from p_2, etc. The length of the maximal firing sequence of S_4 is thus $(x_1 + 1)(x_2 + 1)(x_3 + 1)(x_4 + 1) - 1$, where x_j is the marking of p_j prior to the firing of level-j transitions in S_4. Thus $x_1 = 2$. Since nothing can fire until "start" has fired (which, incidentally, implies $x \cdot y \neq 0$), we have $x_2 = x-1$ and $x_4 = y-1$. The value of x_3 is also $x_3 = x_2 = x-1$ because, in the maximal firing sequence, level 3 starts firing only after level 2 has transferred all tokens from p_2 to p_2', which is the same place as p_3. By counting the "start" firing, we get a maximum total of $(2+1)(x-1+1)(x-1+1)(y-1+1) - 1 + 1$, or $3 \cdot x^2 \cdot y$, as desired. At the end of such a maximal sequence, the argument $\langle x, y \rangle$ has been copied into the "output" places p_3' and p_4'.

QED

Lemma 6.3:

Every diophantine polynomial can be weakly computed by a Petri Net for all positive arguments in the sense of Definition 6.3.

Proof:

We construct a λ-free Petri Net Weak Computer for the polynomial by concatenating the Petri Nets corresponding to its monomials, in some summing order, as shown in Figure 6.4 for the example $3 \cdot x^2 \cdot y + 2 \cdot y \cdot z + x \cdot z + 2$. We identify the output places of one monomial computer with the corresponding input places of the next monomial computer. We also let the "start" transition of each monomial computer (except the first) remove the token from the preceding "run" place. This enforces the summing sequence and makes the operation easier to follow, although it is not essential. It will be useful in later applications, however. Finally, we allow for some extra firings to account for the constant term. The maximum firing sequence requires that each monomial Net be maximally fired; this makes a full copy of the argument available for the next monomial Net, if the argument was positive.

QED

The reason we restrict the argument to be positive is that, for a given summing order, certain zero arguments can prevent the transmission of some positive variables to non-zero monomials later in the sum. This happens in the example above if x (or y) is zero and y (or x) and z are positive. For $\langle x, y, z \rangle = \langle 0, 1, 1 \rangle$ there should be a firing sequence of length 4 when, in fact, only the constant can fire (length 2). But this can

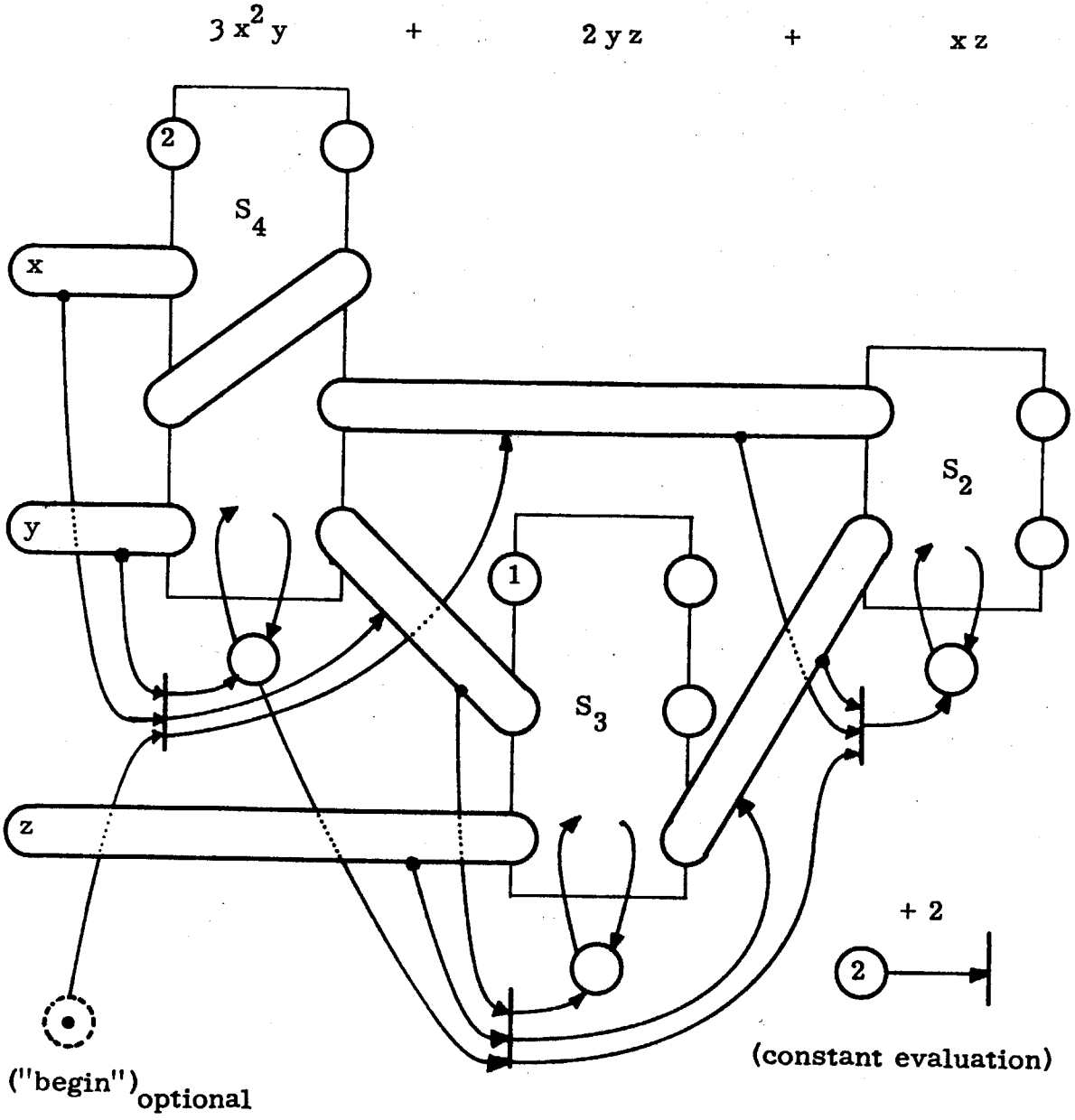

3 x² y + 2 y z + x z
2
S4
x
y
z
1
S3
S2
+ 2
2
("begin")optional
(constant evaluation)

Figure 6.4

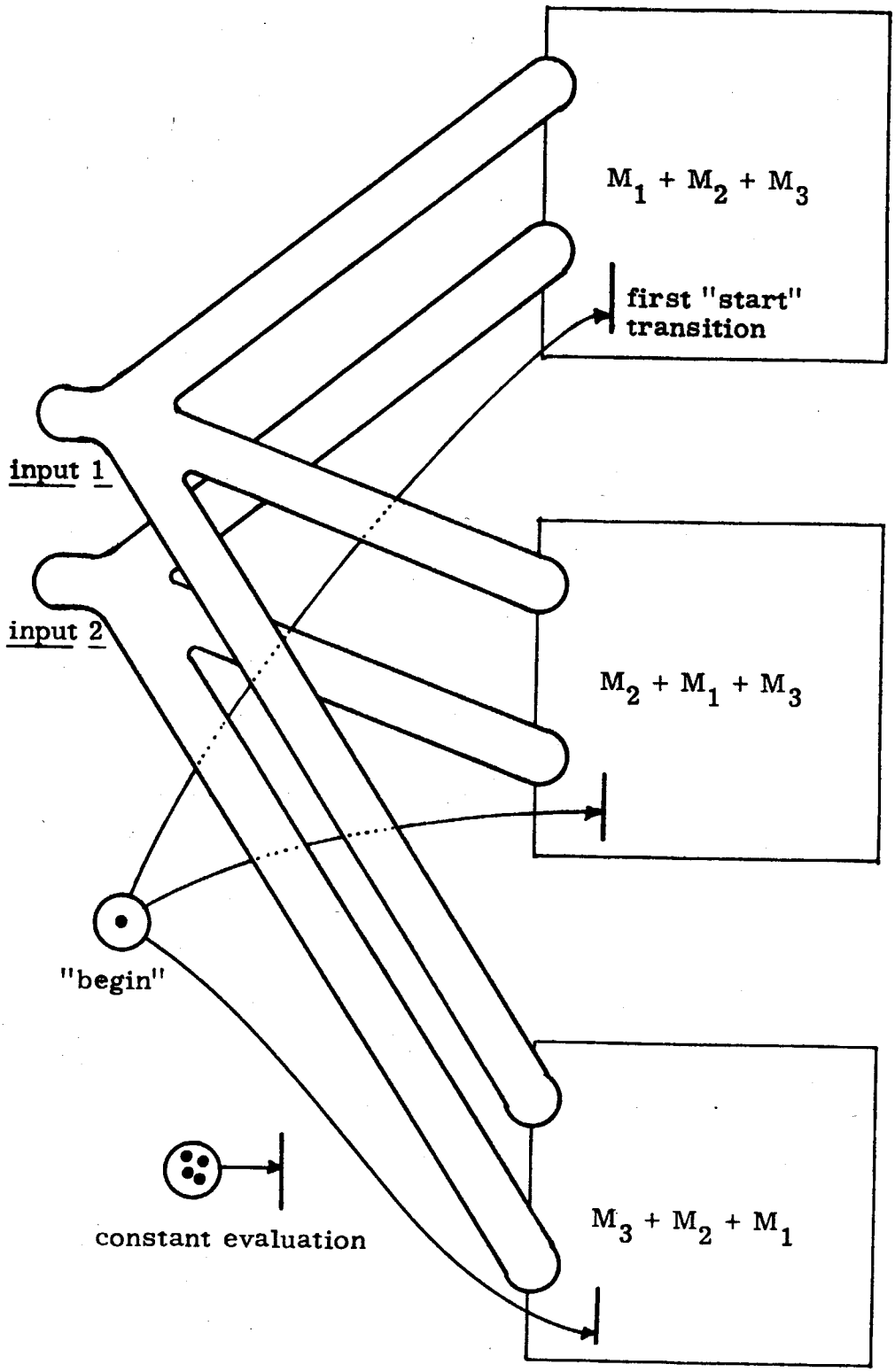

$M_1 + M_2 + M_3$
first "start" transition
input 1
input 2
$M_2 + M_1 + M_3$
"begin"
constant evaluation
$M_3 + M_2 + M_1$

Figure 6.5

be avoided, as we show in:

Theorem 6.2:

Every diophantine polynomial can be computed by a Petri Net Weak Computer (for all non-negative arguments).

Proof:

For every subset of zero-valued variables there is at least one way of summing the monomials of a polynomial by the construction used in the proof of Lemma 6.3.*) So we simply permit several possible orderings of the monomials. For every such ordering, we construct a Petri Net as described for Lemma 6.3, for the non-constant monomials. We add a "begin" place which initially contains one token, and which is an input place to the first "start" transition of every component fixed-order-summation net. All input places are shared, i.e. we identify input places corresponding to the same variable; these are the input places of the new net. The "begin" place enforces that exactly one summation order takes place. Finally, there is a transition for evaluating the constant term (see Figure 6.5.).

Now, for every non-negative argument, there is at least one component whose maximal firing sequence combined with that of the constant evaluation is of length $P(x_1 \ldots x_n)$, and no component has a longer maximal sequence. Since only one component can fire, the construction achieves the desired objective.

QED

*) For example, by adding those monomials which would be zero last.

In the next chapter we shall see how such a Weak Computer can be used to generate polynomial graphs as the projection of Reachability Sets, and in Chapter 10 it will be used to encode polynomial graphs as Petri Net Languages. In both cases, this forms the basis of the various undecidability proofs.

Remark:

Many different constructions are possible for weakly computing polynomials. All are more or less awkward if they have to be fully general. For a given polynomial it is often possible to "customize" the construction and end up with a smaller and more elegant Petri Net. It should be pointed out that the complexity of the construction presented here is due to the more restricted "firing sequence length" interpretation of weak computation by Petri Nets. Even though it is orders of magnitude more efficient than the construction proposed in Hack [24] for Petri Net languages, it is still of "size" $K \cdot N^2$ in terms of the "size" of the polynomial, whereas a construction using the "output place" interpretation of weak computation would be of "size" $K \cdot N$, for a reasonable definition of the notion of the "size" of nets and polynomials, such as total number of arcs in a net and sum of all exponents in a polynomial.

CHAPTER 7

INCLUSION AND EQUALITY PROBLEMS FOR REACHABILITY SETS

7.1 The Decidability Problems

In this chapter we investigate the problem of comparing the Reachability Sets of two Petri Nets A and B, each with a given initial marking. If A and B have the same number of places, and if these places are indexed 1 ... r in both nets, we can compare their Reachability Sets R(A) and R(B) directly. Given such A and B:

The Inclusion Problem (IP) is the question of whether $R(A) \subseteq R(B)$.

The Equality Problem (EP) is the question of whether $R(A) = R(B)$.

Sometimes we are only interested in comparing the Reachability Sets restricted to a certain subnet in each Petri Net. In this case we must have two subnets of the same number of places, as well as a bijection between these two subsets of places, in order to be able to compare submarkings; the nets themselves need not have the same number of places. Without loss of generality, we shall assume that the subnets consist of the first n places of two given Petri Nets A and B. Now we compare the projections of the Reachability Sets on the first n coordinates:

Definition 7.1:

The projection on the first n coordinates of a set $W \subseteq \mathbb{N}^r$, where $r \geq n$, is the smallest set $P_n(W) \subseteq \mathbb{N}^n$ such that $W \subseteq P_n(W) \times \mathbb{N}^{r-n}$.

Thus each vector in $P_n(W)$ consists of the first n coordinates of some vector in W. The $\times$ in the definition represents the cartesian product.

We also use $\times$ to denote the "concatenation" of two vectors: if $V \in \mathbb{N}^n$ and $V' \in \mathbb{N}^{r-n}$, then $V \times V'$ denotes the vector in $\mathbb{N}^r$ which is the element of the singleton set $\{V\} \times \{V'\} \subseteq \mathbb{N}^r$. Thus, every vector in W is the "concatenation" of a vector in the projection $P_n(W)$ and some arbitrary vector of r-n coordinates.

Let A and B be two Petri Nets with their initial marking, such that they both have at least n places (but not necessarily the same number):

The Subspace Inclusion Problem (SIP) is the question of whether $P_n(R(A)) \subseteq P_n(R(B))$.

The Subspace Equality Problem (SEP) is the question of whether $P_n(R(A)) = P_n(R(B))$.

In the next sections we shall show that these four problems (IP, EP, SIP and SEP) are all undecidable, because the PGIP, which is undecidable by Theorem 6.1, can be reduced to them. Figure 7.1 shows the various reducibilities; thin arcs are the trivial reducibilities of a special case to a general case.

7.2 The Subspace Inclusion Problem (SIP)

Now we shall use the fact that Petri Nets can weakly compute polynomials.

Lemma 7.1:

Given a polynomial $Q(x_1, \ldots, x_n)$ with non-negative integer coefficients, there exists a Petri Net A such that the projection of its reachability set on the first n + 1 coordinates is the graph of Q: $P_{n+1}(R(A)) = G(Q)$.

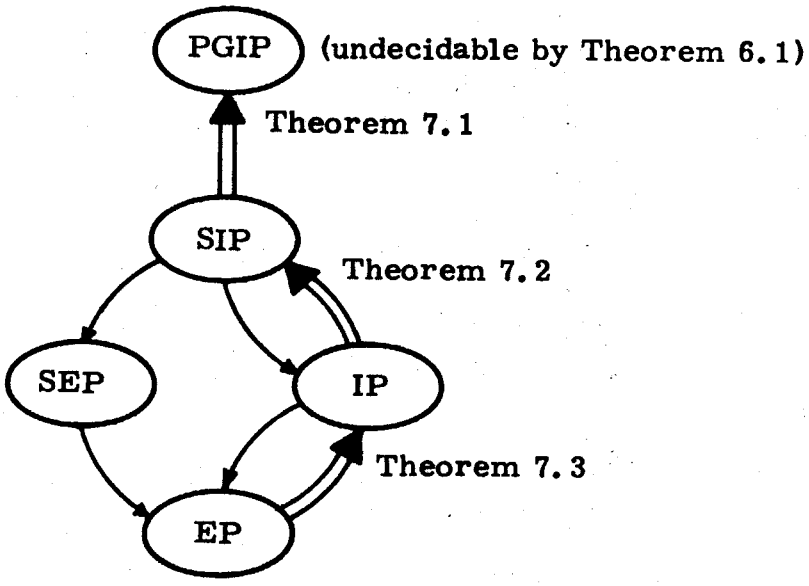


Figure 7.1

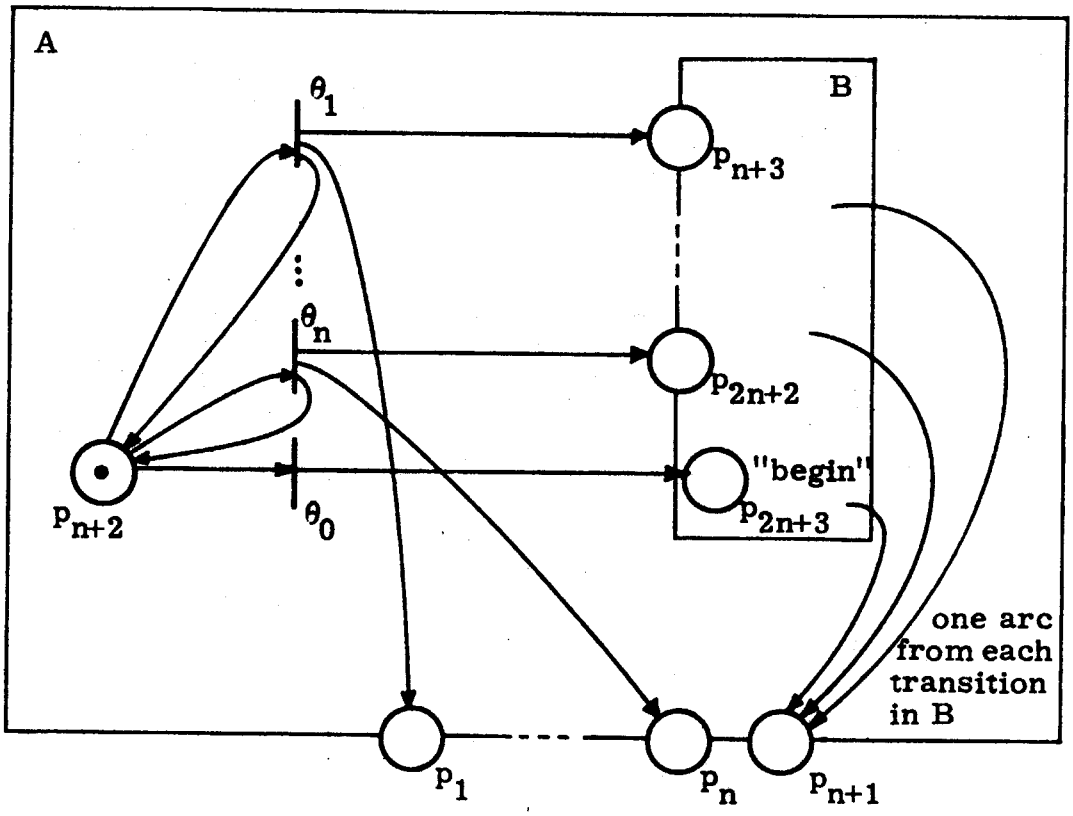


Figure 7.2

Proof:

Let B be a Petri Net Computer for polynomial Q, as described in Chapter 6, with a "begin" place as used in Theorem 6.2. If B has r places, let us index them from n+3 through r+n+2 such that $p_{n+3} \cdots p_{2n+2}$ are the "input" places for variables $x_1 \ldots x_n$, and that p_{2n+3} is the "begin" place. The initial marking of B's places is as constructed in Chapter 6, except that the "begin" place and all "input" places are initially unmarked. Thus no transition in B can fire until the "begin" place receives a token.

To construct Petri Net A, we take this copy of B and add n+2 places $p_1 \cdots p_{n+2}$ and n+1 transitions $\theta_0 \ldots \theta_n$ (see Figure 7.2). Place p_{n+2} is initially marked with one token; places $p_1 \ldots p_{n+1}$ are initially unmarked. Transition θ_0 transfers a token from p_{n+2} to the "begin" place of B, p_{2n+3}. Each transition θ_i, $1 \leq i \leq n$, selfloops on p_{n+2} and, at each firing, deposits a token into place p_i and into the i^{th} input place of B, p_{n+2+i}. Finally, place p_{n+1} receives one arc from every transition in B, and thus collects a number of tokens equal to the length of a firing sequence in B.

But before any transition in B can fire, θ_0 must fire, and before that only the θ_i, $1 \leq i \leq n$, could fire. Suppose each θ_i, $1 \leq i \leq n$, fires x_i times before θ_0 fires. Then places $p_1 \ldots p_n$ are marked with the argument $\langle x_1 \ldots x_n \rangle$ with which B starts to compute, and generates anywhere between zero and $Q(x_1 \ldots x_n)$ tokens in p_{n+1}. Thus: $P_{n+1}(R(A)) = G(Q)$.

QED

From this follows:

Theorem 7.1:

The PGIP is recursively reducible to the Subspace Inclusion Problem (SIP).

Proof:

Given two polynomials with non-negative integer coefficients we construct two Petri Nets whose projected Reachability Sets are the graphs of the respective polynomials as indicated by Lemma 7.1. Then a test for the SIP will also decide the corresponding PGIP.

QED

Corollary 7.1:

The SIP is undecidable.

Proof:

This follows from the undecidability of the PGIP (Theorem 6.1) and the reducibility of the PGIP to the SIP (Theorem 7.1).

QED

Remark:

We could easily prove now that the SEP is also undecidable. Indeed, projected Reachability Sets are closed under union: If A and B are two Petri Nets, each with at least n places, then there exists a Petri Net C such that $P_n(R(C)) = P_n(R(A)) \cup P_n(R(B))$. Such a net C can be constructed by adding a "run" place to A and another "run" place to B, both initially unmarked, a new "begin" place initially marked with one token, and two transitions which transfer the "begin" token to one or the other "run" place. Finally, we identify the n first

places of A and B and let them be the n first places of the new net C. Now we can use the fact that $P_n(R(A)) \subseteq P_n(R(B)) \Rightarrow P_n(R(A)) \cup P_n(R(B)) = P_n(R(B))$ to reduce the SIP to the SEP, thus proving the undecidability of the SEP.

The undecidability of the SEP will of course follow directly from our proof of the undecidability of the EP in section 7.4.

7.3 The Inclusion Problem (IP)

Now we shall show how we can modify Petri Nets of r places to "forget" the marking in r-n "uninteresting" places and thus reduce the SIP to a comparison of complete Reachability Sets, the IP.

Theorem 7.2:

The SIP is recursively reducible to the IP.

Proof:

Suppose we are given two Petri Nets of r and r' places, respectively, and we wish to test, for the two projections on the first r coordinates of the respective Reachability Sets, whether one is a subset of the other.

First, we note that we can always add $|r - r'|$ places to the smaller net (without renumbering the original places) to get two nets with the same number of places, say r. If we don't connect these new places to any transitions, we will not change the Reachability Set as far as the old places are concerned, and thus the problem is reduced to the following:

Given two Petri Nets A and B of r, $r \geq n$, places each, is

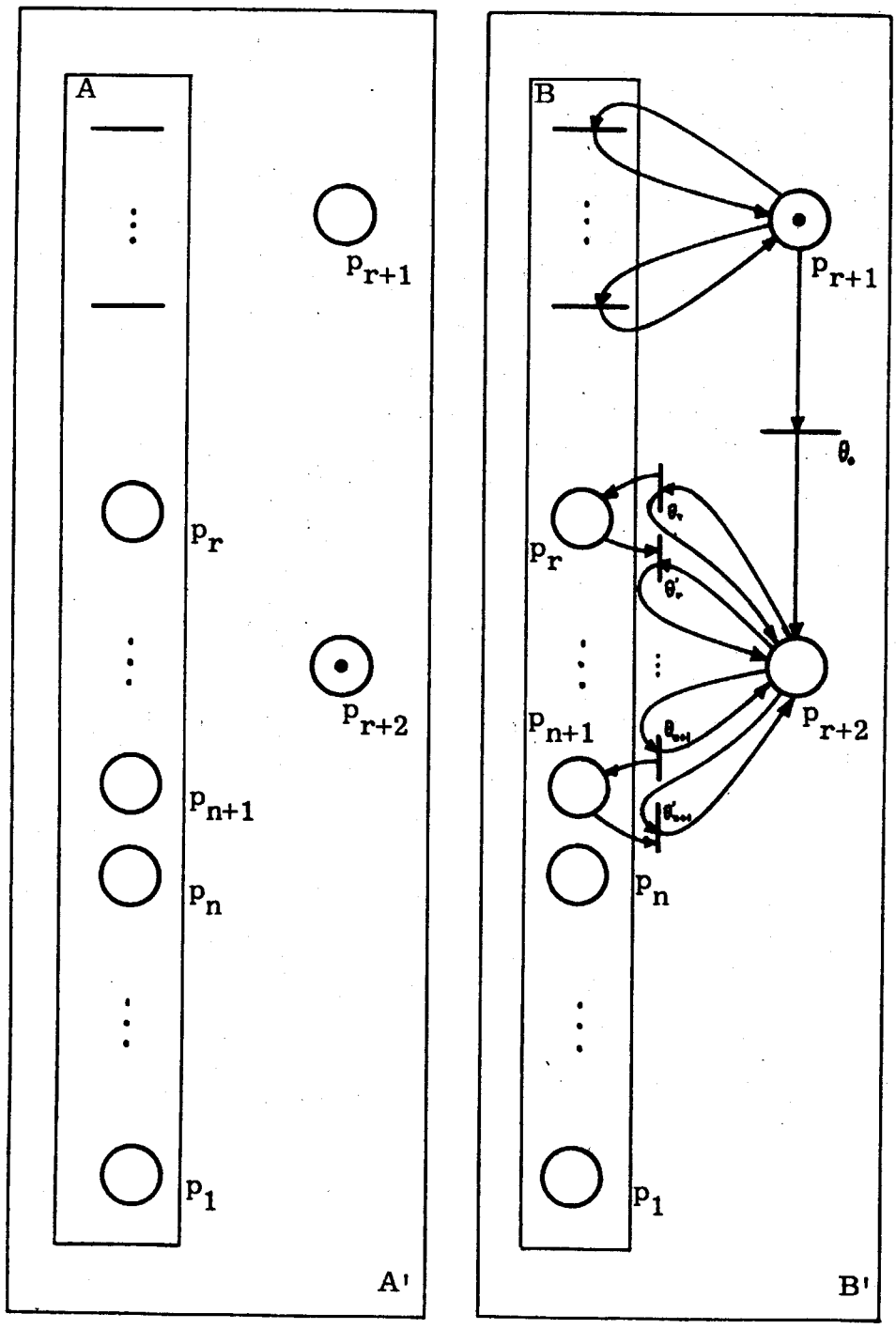


Figure 7.3

$$P_n(R(A)) \subseteq P_n(R(B))\,?$$

We shall modify both nets by adding two new places p_{r+1} and p_{r+2} to each net, and we shall modify the Reachability Sets in such a way as to make the inclusion depend only on the first n coordinates.

Specifically, the modifications are shown in Figure 7.3. Petri Net A' differs from A only in the two additional places, which are permanently marked $\langle 0, 1\rangle$. Therefore we have:

$$R(A') = R(A) \times \{\langle 0, 1\rangle\}$$

Petri Net B' is obtained from B by similarly adding two new places p_{r+1} and p_{r+2}, which are initially marked $\langle 1, 0\rangle$. But B' also contains several new transitions: a transition θ_0 which carries a token from p_{r+1} to p_{r+2}, and, for each "uninteresting" place p_i, $n+1 \le i \le r$, two transitions θ_i and θ_i'. θ_i removes a token from p_i and θ_i' deposits a token in p_i, and both θ_i and θ_i' self-loop on p_{r+2}. Finally, place p_{r+1} self-loops on every transition of B. Thus p_{r+1} plays the role of a "run" place for the "old" transitions and p_{r+2} plays the role of a "run" place for the "new" transitions.

As long as θ_0 has not transferred the token from p_{r+1} to p_{r+2}, B' behaves like B. But after θ_0 has fired, the "old" transitions are frozen. Since no other transitions involve the "interesting" places $p_1 \cdots p_n$, the marking of these places will not change anymore. But the "new" transitions θ_i, θ_i' (for $n+1 \le i \le r$) can now be used to generate any arbitrary marking in the "uninteresting" places $p_{n+1} \cdots p_r$, thus effectively "erasing" the information contained in these places. It follows that:

$$R(B') = R(B) \times \{\langle 1, 0\rangle\} \cup P_n(R(B)) \times \mathbb{N}^{r-n} \times \{\langle 0, 1\rangle\}$$

Recall that $R(A') = R(A) \times \{\langle 0, 1\rangle\}$.

Thus:

$$R(A') \subseteq R(B') \Leftrightarrow R(A) \subseteq P_n(R(B)) \times \mathbb{N}^{r-n}$$

Since, by definition (Definition 7.1), $P_n(R(A))$ is the smallest set such that $R(A) \subseteq P_n(R(A)) \times \mathbb{N}^{r-n}$, this is equivalent to $P_n(R(A)) \subseteq P_n(R(B))$. Hence:

$$R(A') \subseteq R(B') \Leftrightarrow P_n(R(A)) \subseteq P_n(R(B))$$

Since we constructed an instance of the IP from the proposed SIP, we conclude that the SIP is reducible to the IP.

QED

Corollary 7.2:

The IP is undecidable.

This is Rabin's result, which he first obtained for Vector Addition Systems. As mentioned in Chapter 6, our proof is largely based on his original proof (1967) as modified in 1972.

7.4 The Equality Problem (EP)

The first mention of Rabin's Theorem, in Karp and Miller [33], was unfortunately misleading: Rabin was quoted as having shown the undecidability of the Equality Problem (for Vector Addition Systems). When we found out (at Rabin's talk at MIT in 1972 [56]) that the Equality Problem was still open, we became interested in this and other decidability questions. But it was not until October 1974 that the search was successful.

The difficulty lies in the fact that Reachability Sets are not known to be closed under union, as opposed to projected Reachability Sets, as mentioned in section 7.2. We got around this difficulty by controlling the

non-projected coordinates in such a way as to make the equality of the Reachability Sets depend only on the projected Reachability Sets.

Theorem 7.3:

The Inclusion Problem is reducible to the Equality Problem.

Proof:

Suppose we are given two r-place Petri nets A and B. We wish to test whether $R(A) \subseteq R(B)$. We shall construct from A and B two Petri Nets D and E such that:

$$R(A) \subseteq R(B) \Leftrightarrow R(D) = R(E)$$

Both nets D and E will be constructed from a common net C which, in a sense, encodes the union $R(A) \cup R(B)$, and we shall use the fact that:

$$R(A) \subseteq R(B) \Leftrightarrow R(B) = R(A) \cup R(B)$$

Petri Net C is constructed as follows: First, we identify the places of A with the corresponding places of B. This produces the first r places of C. Then we add a "run" place p_{r+1} for the transition of A and a second "run" place p_{r+2} for the transitions of B. Places $p_1 \cdots p_{r+2}$ mentioned so far are initially unmarked. Finally we add a "start" place p_{r+3}, initially marked with one token, and two transitions θ_1 and θ_2 (see Figure 7.4).

Transition θ_1 transfers the "start" token from p_{r+3} to p_{r+1} and also deposits the initial marking of A into $p_1 \cdots p_r$. Similarly, transition θ_2 transfers the "start" token from p_{r+3} to p_{r+2} and deposits the initial marking of B into $p_1 \cdots p_r$. Thus, depending on whether θ_1 or θ_2 fires first, C will simulate either A or B, and we have:

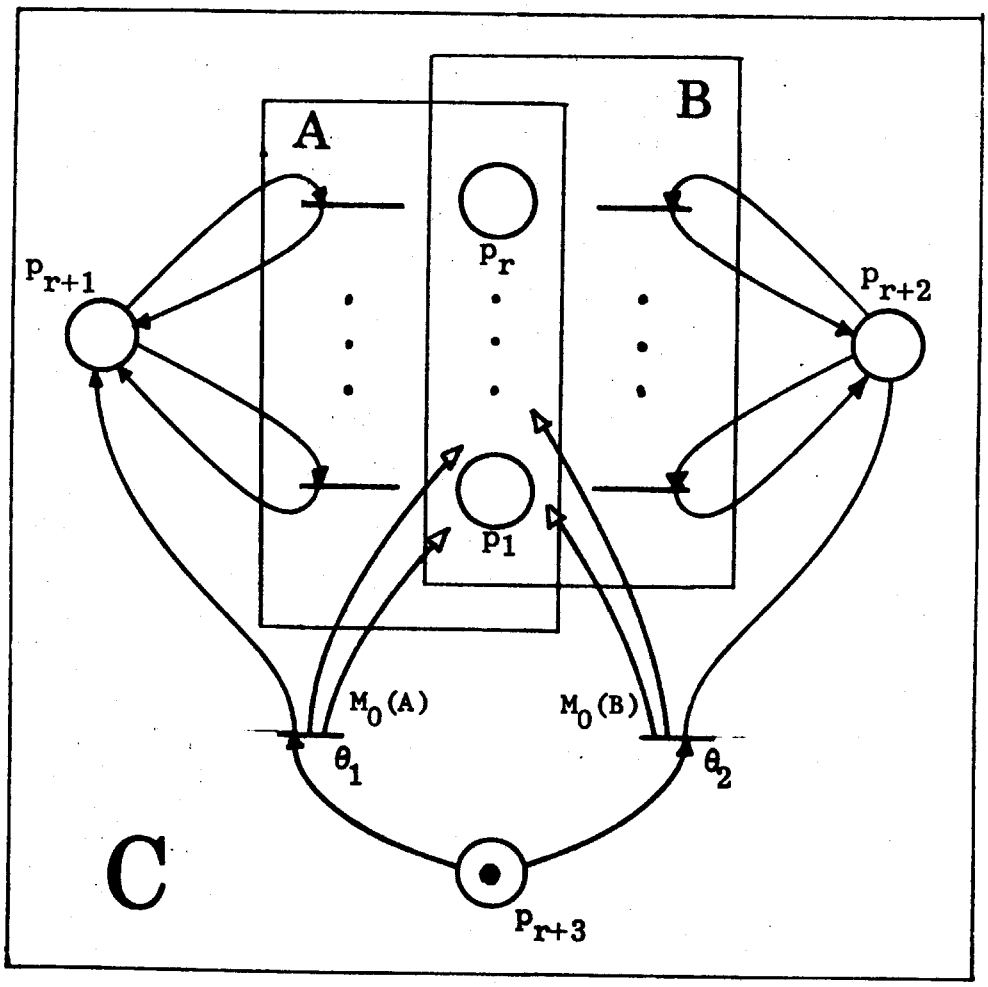


Figure 7.4

$$R(C) = \langle 0 \rangle^r \times \langle 0, 0, 1 \rangle \cup R(A) \times \langle 1, 0, 0 \rangle \cup R(B) \times \langle 0, 1, 0 \rangle$$

Now we can construct Petri Nets D and E as illustrated in Figure 7.5. D is obtained from C by adding transition θ_3, which removes the token from p_{r+2}. This can happen only if the first firing was θ_2 and C was in fact simulating B. A firing of θ_3 thus produces only new markings of the form $R(B) \times \langle 0, 0, 0 \rangle$, and we have:

$$R(D) = R(C) \cup R(B) \times \langle 0, 0, 0 \rangle$$

Petri Net E is obtained from D by also adding another transition, θ_4, which can remove a token from p_{r+1}. θ_4 can only fire if C was simulating A, and thus the only new markings are of the form $R(A) \times \langle 0, 0, 0 \rangle$. Hence:

$$R(E) = R(D) \cup R(A) \times \langle 0, 0, 0 \rangle$$
$$= R(C) \cup (R(A) \cup R(B)) \times \langle 0, 0, 0 \rangle$$

Since no marking in R(C) ends in $\langle 0, 0, 0 \rangle$, we conclude that:

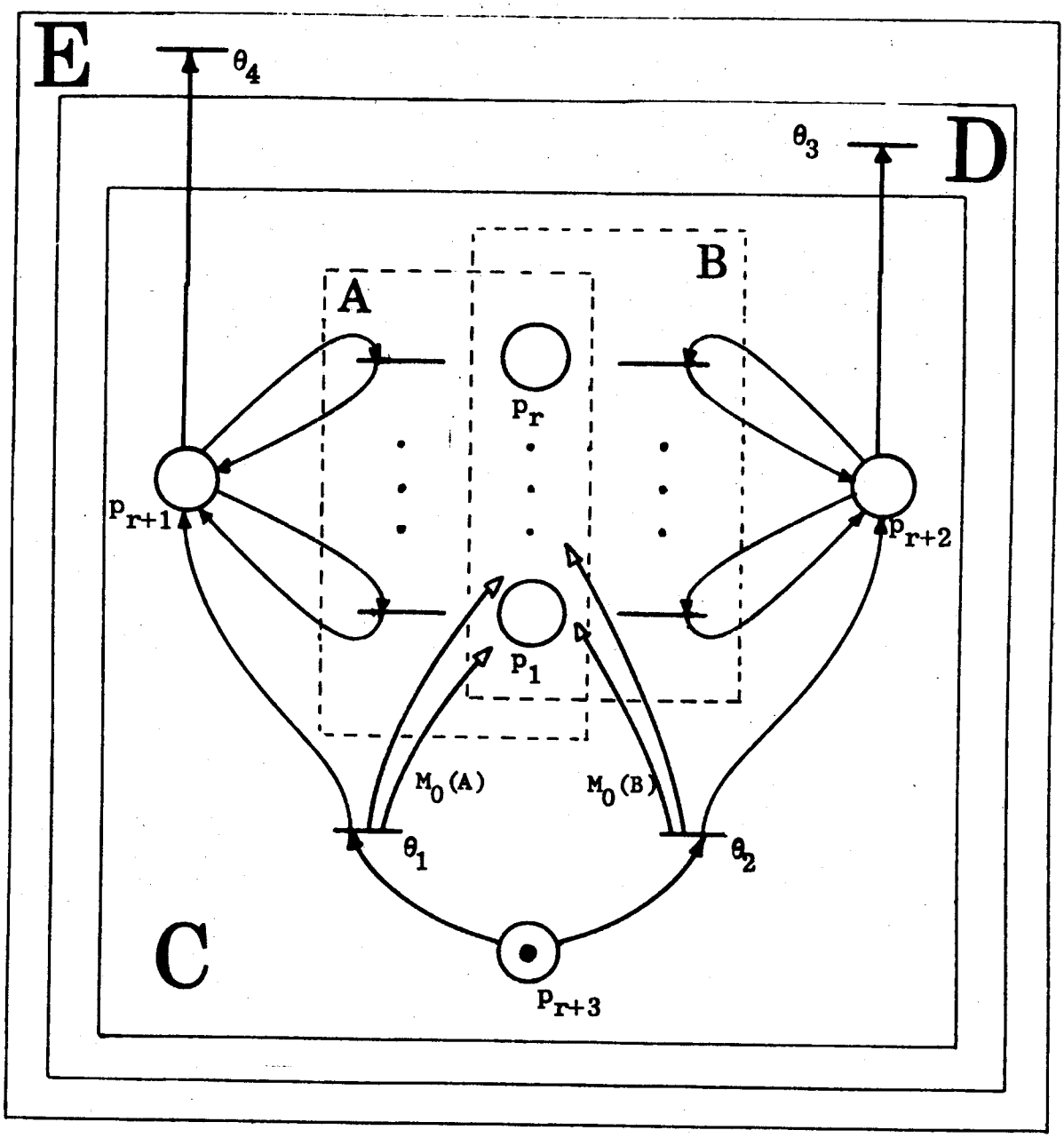
$$R(D) = R(E) \Leftrightarrow R(A) \subseteq R(B)$$

QED

The combined result of Theorems 7.1 ... 7.3 and the trivial reducibilities is:

Theorem 7.4:

The EP, IP, SIP and SEP are all recursively equivalent to each other, and are all undecidable.



$R(A) \subseteq R(B) \quad \Leftrightarrow \quad R(D) = R(E)$

Figure 7.5

In fact, we have proved a much stronger result, since the instance of the EP used in Theorem 7.3 is quite singular: The two Petri Nets whose Reachability Sets we compare differ only by the presence or absence of a single transition θ_4!

Thus we may state:

Theorem 7.5:

It is undecidable whether the removal of a particular transition in a Petri Net changes the Reachability Set or not.

We should point out, however, that this result is not as drastic as it might seem: even though the set of reachable markings may not change, its connectivity, as determined by which marking is reachable from which other marking by which firing sequence, is usually quite changed. But we shall see that a similar question for Petri Net Languages is also undecidable (Chapter 10).

CHAPTER 8

PETRI NET LANGUAGES: DEFINITIONS AND PROPERTIES

8.1 Labelled Petri Nets

Until now, we have mainly been interested in those properties that are directly related to the reachable markings of the net. In effect, for a given Petri Net N with an initial marking M_0, we have been studying the properties of the Reachability Set $R_N(M_0)$.

In many cases, however, it is the properties of the sets of firing sequences $S_N(M_0)$ or $T_N(M_0, M_f)$ that are of interest. For example, if the Petri Net describes an asynchronous system, the various event occurrences in the system are represented by transition firings, and we may be interested in which sequences are possible from a given initial state. This involves a study of the set of firing sequences $S_N(M_0)$. Sometimes we would like to know which sequences can lead from the initial state to a given final state, represented by a final marking M_f. In this case we must look at the set of terminal firing sequences $T_N(M_0, M_f)$.

In order to relate the various transitions to the events whose occurrence their firing represents, we attach labels to the transitions. If t is a transition, then its label $\Lambda(t)$ represents the event whose occurrence (in the system) is modelled by a firing of t (in the Petri Net). Now, if each transition received a distinct label, the labelling would add nothing new. The advantage of using a labelling function lies in the fact that we can model a single event by several transitions, and thus represent the case of an event which may occur under different circumstances, even if the corresponding markings are incomparable.

The labelling function also permits us to distinguish between "visible"

and "invisible" transitions, for example in the description of the input-output behaviour of a system, where "internal" events are to be ignored. Just as we use submarkings to distinguish between "interesting" and "uninteresting" places when we study Reachability Sets, we use the notion of λ-transitions to represent the "invisible" transitions. Their label is the "empty" label λ, which is another way of saying that they are unlabelled.

Definition 8.1:

A Labelled Petri Net $A = \langle N, \alpha, \Lambda \rangle$ over an alphabet α is a Petri Net $N = \langle \Pi, \Sigma, F, B, M_0 \rangle$ together with a labelling function $\Lambda: \Sigma \to \alpha$. If Λ is total, the labelled net is said to be λ-free; if Λ is partial, those transitions which have no label in α are called λ-transitions.

Definition 8.2:

The label sequence $\Lambda(\sigma)$ corresponding to a firing sequence $\sigma \in \Sigma^*$ is defined recursively as follows:

$\Lambda(\lambda) = \lambda$

$\Lambda(\sigma t) =$ if t is labelled then $\Lambda(\sigma) \cdot \Lambda(t)$
else $\Lambda(\sigma)$

Thus, λ-transitions in firing sequences transform as if their label was the empty string λ.

Now that the labelling function Λ has been defined for strings ($\Lambda: \Sigma^* \to \alpha^*$), we can extend it to sets of strings in the natural way. In particular, we use the following notation:

Let A be the labelled Petri Net $\langle N, \mathcal{A}, \Lambda \rangle$:

$$S_A(M_0) = \{x \in \mathcal{A}^* \mid \exists \sigma \in S_N(M_0): \; x = \Lambda(\sigma)\}$$

$$T_A(M_0, M_f) = \{x \in \mathcal{A}^* \mid \exists \sigma \in T_N(M_0, M_f): \; x = \Lambda(\sigma)\}$$

The set $S_A(M_0)$ is called the prefix language of the labelled Petri Net A (for initial marking M_0). The set $T_A(M_0, M_f)$ is called the terminal language of A (for initial and final markings M_0 and M_f).

Definition 8.3:

(a) $\mathcal{L}$ is the class of all prefix languages generated by λ-free labelled Petri Nets.

(b) $\mathcal{L}^\lambda$ is the class of all prefix languages generated by unrestricted labelled Petri Nets.

(c) $\mathcal{L}_0$ is the class of λ-free terminal languages generated by λ-free labelled Petri Nets.

(d) $\mathcal{L}_0^\lambda$ is the class of terminal languages generated by unrestricted labelled Petri Nets.

Remark:

$\mathcal{L}_0$-Languages (part (c) of the above definition) are required to be λ-free (i.e. they contain no words of length zero) to ensure the closure under union of the class $\mathcal{L}_0$. Thus $T_A(M_0, M_f)$ is in $\mathcal{L}_0$ only if $M_f \neq M_0$. The restriction is not as severe as it seems: For every language $T_A(M, M)$ (called a cyclic language) there exist A', M_0' and M_f' such that $T_A(M, M) = \{\lambda\} \cup T_{A'}(M_0', M_f')$ and $M_0' \neq M_f'$. For a further discussion of this point refer to Hack [24].

Figure 8.1 summarizes Definition 8.3. It is clear from the definition

	no λ-transitions	λ-transitions allowed
all firing sequences	$\mathcal{L}$	$\mathcal{L}^{\lambda}$
only terminal firing sequences	$\mathcal{L}_0$	$\mathcal{L}_0^{\lambda}$

Figure 8.1

that

$$\mathcal{L} \subseteq \mathcal{L}^{\lambda}$$
$$\mathcal{L}_0 \subseteq \mathcal{L}_0^{\lambda}$$

8.2 Standard Form

For many proofs and constructions it is useful to impose certain constraints on the Petri Nets used to generate a given language. We are of course mainly interested in constraints which do not restrict the class of languages that can be generated. If a certain set of constraints is particularly useful, it makes sense to define a Standard Form for language-generating Petri Nets:

Definition 8.4:

A Labelled Petri Net A is said to be in Standard Form iff it satisfies the following constraints:

(a) The initial marking M_0 is standard and consists of exactly one token in a designated "start" place, and zero tokens in all other places. Since M_0 is understood, we shall use S_A instead of $S_A(M_0)$ for the prefix language of A.

(b) For defining the terminal language of A, the final marking is standard, and is the zero marking: $M_f = 0$. We shall use T_A instead of $T_A(M_0, 0)$ if M_0 is the standard initial marking.

(c) No transition is firable at the zero marking, i.e. every transition has at least one input place.

The following Standard Form Theorem asserts that these constraints do not change the classes of Petri Net Languages that can be generated by

nets in Standard Form:

Theorem 8.1:

For every Labelled Petri Net A with initial and final markings M_0 and M_f, there exists a Labelled Petri Net B which is in standard form, and which generates the same language as A:

$$S_A(M_0) = S_B$$
$$T_A(M_0, M_f) = T_B \qquad \text{(assuming } M_f \neq M_0\text{)}$$

Proof:

Let $A = \langle N, \alpha, \lambda \rangle$ and $N = \langle \Pi, \Sigma, F, B, M_0 \rangle$, with $\Pi = \{p_1 \ldots p_r\}$ and $\Sigma = \{t_1 \ldots t_s\}$.

Let us also assume that every transition $t \in \Sigma$ has at least one input place in Π. This can always be guaranteed by including a "run" place which self-loops on every transition in Σ, and which contains one token at all markings, including M_0 and M_f. Such a "run" place does not change the firability or the result of a firing of any transition, and hence does not affect any firing sequences.

We shall transform N into a new net N' by adding a new place - the "start" place p_0 - and a number of transitions. The standard initial marking M_0' consists of one token in the "start" place p_0 and zero tokens in all other places (Π).

(a) To satisfy condition (a) of the Standard Form, we add, for each transition t_i which could fire in N at M_0 (i.e. for which $M_0 \geq F(t_i)$), a new transition t_i' whose only input is the "start" place p_0, and whose output places are such that the marking resulting from a firing of t_i' at the standard initial marking is the same as that

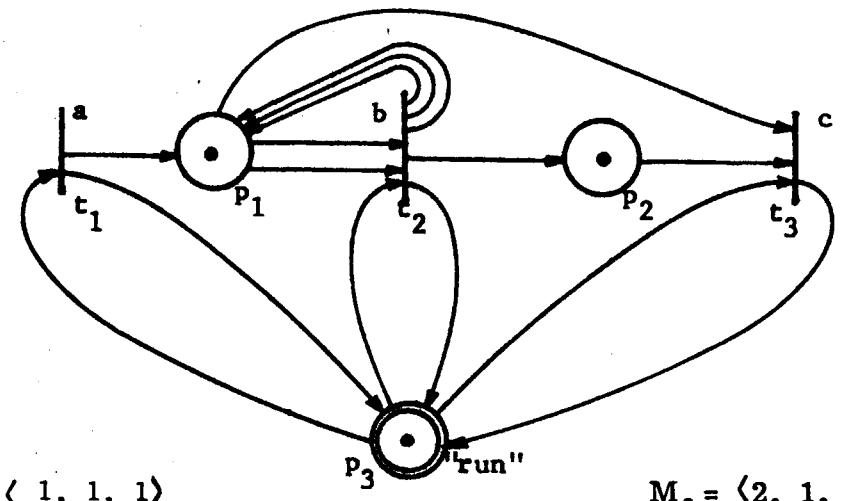
resulting from a firing fo t_i at M_0:

$$B(t_i') = M_0 - F(t_i) + B(t_i)$$

The label of t_i' is the same as that of t_i. It is now easy to see that every label sequence $\Lambda(\sigma)$ of A, corresponding to firing sequence $\sigma \in S_N(M_0)$, is also generated by the firing sequence $\sigma' \in S_{N'}(M_0')$ which differs from σ only in the first firing where some t_i has been replaced by t_i'. Conversely, every firing sequence of N' must start with a firing of some t_i', since all t_i-transitions are disabled at M_0': no place in Π is marked at M_0'.

(b) To satisfy condition (b) of the Standard Form, we add, for each transition t_i which could fire last in a terminal firing sequence of N (i.e. such that $M_f \geq B(t_i)$), a new transition t_i'', labelled like t_i, and such that $B(t_i'') = 0$ and $F(t_i'') = M_f - B(t_i) + F(t_i)$. This implies that t_i'' is firable only if t_i is firable (by construction, $F(t_i') \geq F(t_i)$), and that a firing of t_i'' reaches the zero marking iff a firing of t_i reaches M_f. Thus no new label sequences are obtained, and every terminal firing sequence σ of N can be replaced by a terminal firing sequence σ' of N' by priming the first firing of σ (replacing t_i by t_i') and by double-priming the last firing σ - provided the length of σ is at least 2. Since $M_f \neq M_0$ by assumption, the only remaining case is a terminal sequence of length one, i.e. transitions t_i such that $M_0[t_i\rangle M_f$. For such a t_i we add t_i''' labelled like t_i whose sole input place is the "start" place p_0, and which has no output places: $M_0'[t_i'''\rangle 0$.

(c) Since all new transitions have input places if all old transitions have input places (as assumed), condition (c) of the Standard Form is also satisfied.



$M_0 = \langle 1, 1, 1 \rangle$ $M_f = \langle 2, 1, 1 \rangle$

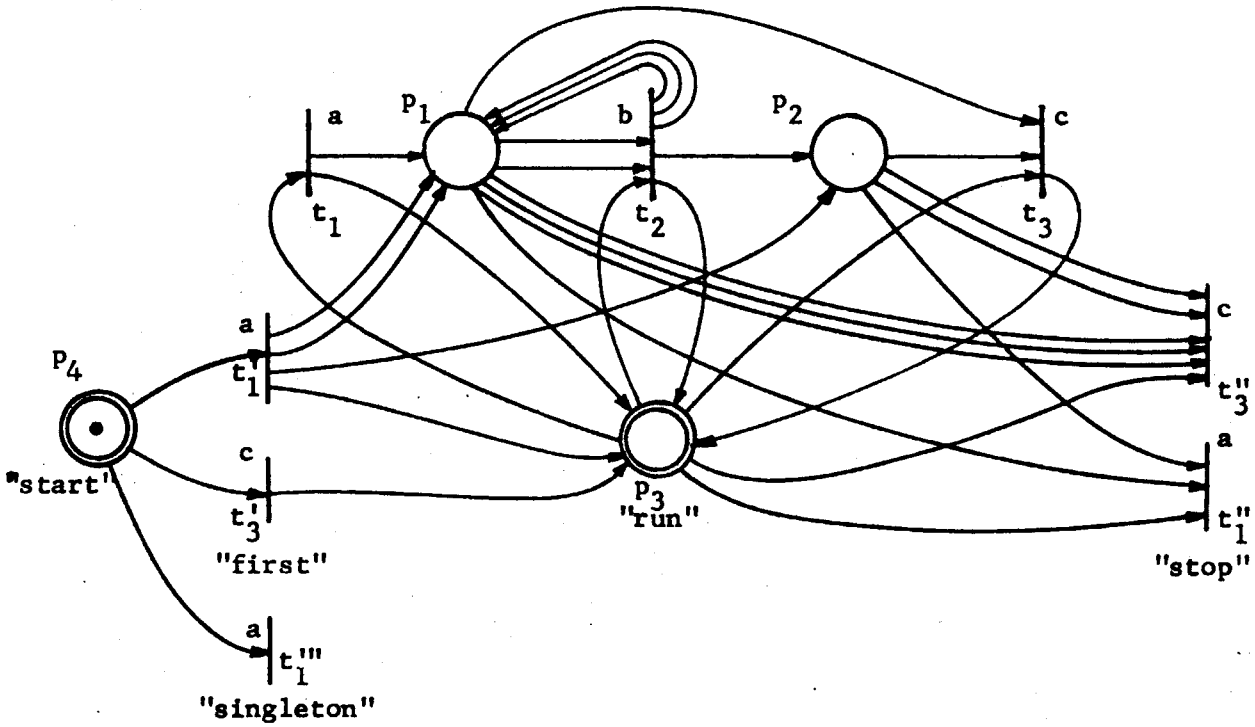


Figure 8.2

The new Labelled Petri Net B consists of the modified net N', and its labelling function is the extension of Λ which assigns to each new transition (t_i' or t_i'' or t_i''') the label of the old transition (t_i) to which it is due. B is in Standard Form by construction, and has the same prefix and terminal language as A.

QED

Figure 8.2 shows a Labelled Petri Net A and the corresponding net in Standard Form obtained by the construction above, B. For a more detailed discussion of Standard Form Labelled Petri Nets, refer to Hack [24].

8.3 The Relationship Between Prefix and Terminal Petri Net Languages

It is not difficult to add λ-transitions to a Labelled Petri Net such that the zero marking becomes reachable from every marking, without changing the prefix language generated by the net. For example, if we have a "run" place which self-loops on all "old" transitions of the net, we may add to every place a λ-transition which can remove any or all tokens, and a "clear" place which self-loops on all these new λ-transitions. The "run" token can be transferred by a λ-transition to the "clear" place and later absorbed by another λ-transition. Now the "old" transitions can be frozen after any firing sequence, after which the zero marking can be reached via λ-firings exclusively.

In fact, the sequencing control of the "run" and "clear" places is not needed. The new λ-transitions may be fired at any time to reduce the marking of the net. This does not change the set of label sequences that can be generated, because, by the containment property (Theorem 2.1),

any firing sequence possible at the smaller marking can also be fired at the larger marking.

We have just shown that $\mathcal{L}^{\lambda} \subseteq \mathcal{L}_0^{\lambda}$. But the same principle of being able to reach arbitrarily small markings by the same label sequences as in the original net can also be carried out without introducing new λ-transitions.

Theorem 8.2:

For every Labelled Petri Net A there exists a Labelled Petri Net B whose Prefix and Terminal Languages are equal, up to λ, to each other and to the Prefix Language of A, and such that B is λ-free if A is λ-free:

$$S_B(M_0) = S_A(M_0)$$

$$T_B(M_0, 0) = S_A(M_0) - \{\lambda\} = S_B(M_0) - \{\lambda\} \quad \text{(if A is } \lambda\text{-free)}$$

$$T_B(M_0, 0) = S_A(M_0) = S_B(M_0) \quad \text{(if } \lambda\text{-transitions are allowed)}$$

Proof:*)

The Labelled Petri Net B is obtained from A by adding new transitions. No new places are added, and the initial marking is unchanged. The terminal marking for B will be the zero marking.

Let $\Sigma = \{t_1 \ldots t_s\}$ be the set of transitions of A (the "old" transitions) and let Λ be the labelling function. Each $t_i \in \Sigma$ is replaced by the set of transitions $\{\theta_i^j \mid F(\theta_i^j) = F(t_i) \;\&\; B(\theta_i^j) \leq B(t_j) \;\&\; \Lambda(\theta_i^j) = \Lambda(t_i)\}$. Here, j is simply an extra index to distinguish between the various "new" transitions corresponding to a given "old" transition, and one θ_i^j, say θ_i^0, is an exact copy of t_i. The "new"

*) This proof is based on an idea of J. L. Peterson [25].

transitions are firable at the same markings as the "old" transitions, but they may "lose" any or all the tokens they would deposit. If we now consider an arbitrary firing sequence of the "new" transitions, say $M_0[\sigma'\rangle M'$, then the corresponding firing sequence σ of "old" transitions - obtained by replacing each θ_i^j-firing by a t_i-firing - is also firable and leads to a larger marking: $M_0[\sigma\rangle M \quad \& \quad M \geq M'$. Conversely, if we are given an "old" firing sequence σ such that $M_0[\sigma\rangle M$, we may replace it by a "new" firing sequence where, at each step, we choose the "smallest" θ_i^j capable of being followed by the rest of the firing sequence. The last firing will then reach the zero marking. Thus no new label sequences are added, and any non-empty firing sequence can be replaced by a zero-reaching firing sequence which generates the same label sequence.

QED

Corollary 8.1:

$$\mathcal{L}^{\lambda} \subseteq \mathcal{L}_0^{\lambda}$$

$$\{L-\{\lambda\} \mid L \in \mathcal{L}\} \subseteq \mathcal{L}_0$$

Figure 8.3 illustrates the construction of the proof of Theorem 8.2.

8.4 Closure of Petri Net Languages under Union and Intersection

The closure properties of Petri Net Languages are discussed in detail in Hack [24]. For the purpose of studying the Decidability Questions of Petri Net Languages (Chapters 9 and 10), we only need closure under Union and Intersection.

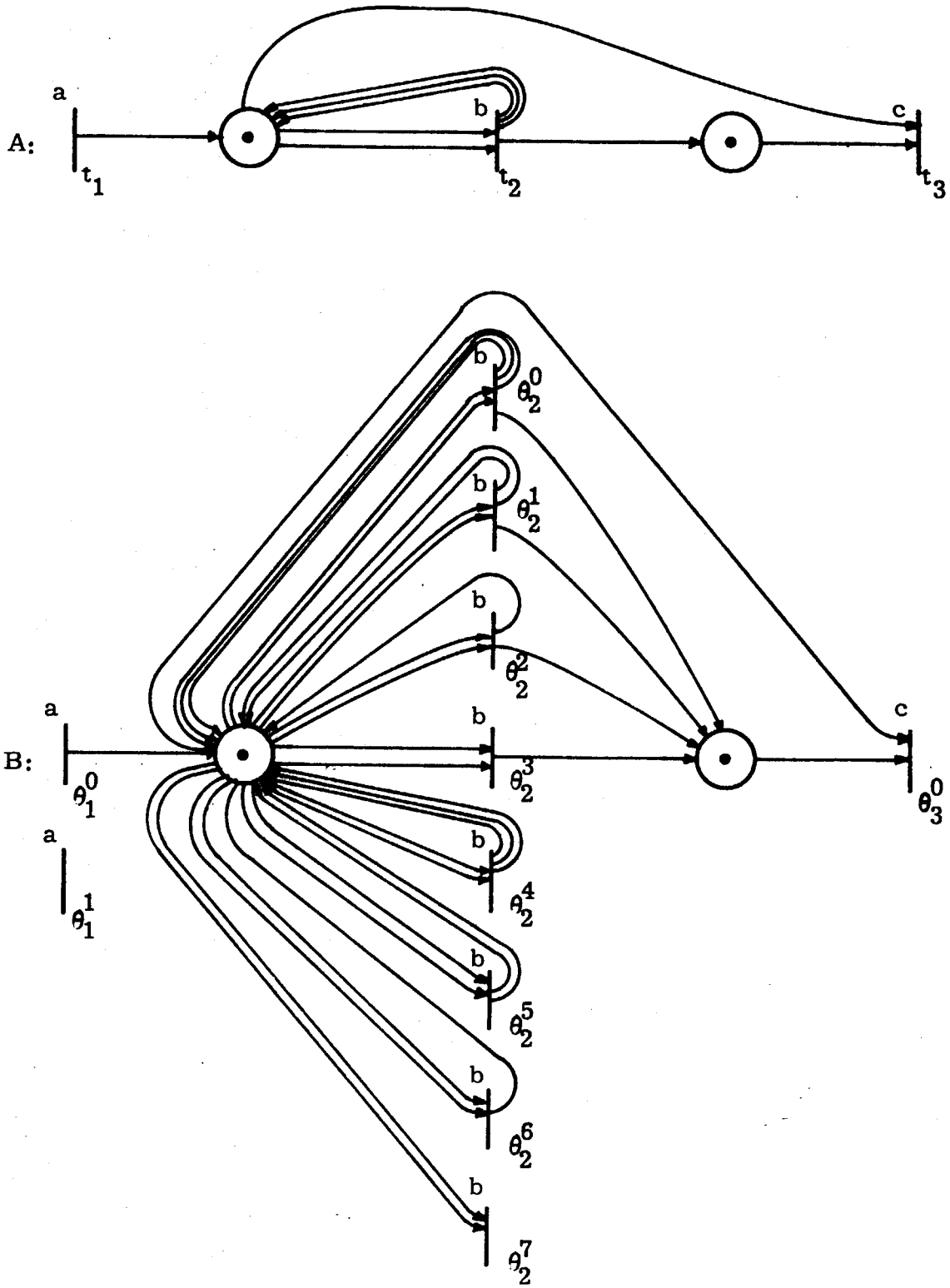

A:
a
b
c
t_1
t_2
t_3
B:
a
b
c
θ_1^0
θ_1^1
θ_2^0
θ_2^1
θ_2^2
θ_2^3
θ_2^4
θ_2^5
θ_2^6
θ_2^7
θ_3^0

Figure 8.3

Theorem 8.3

Given two Labelled Petri Nets A and B, there exists a Labelled Petri Net C which is λ-free if both A and B are λ-free, and whose language is the union of that of A and that of B:

$$S_C = S_A \cup S_B$$
$$T_C = T_A \cup T_B$$

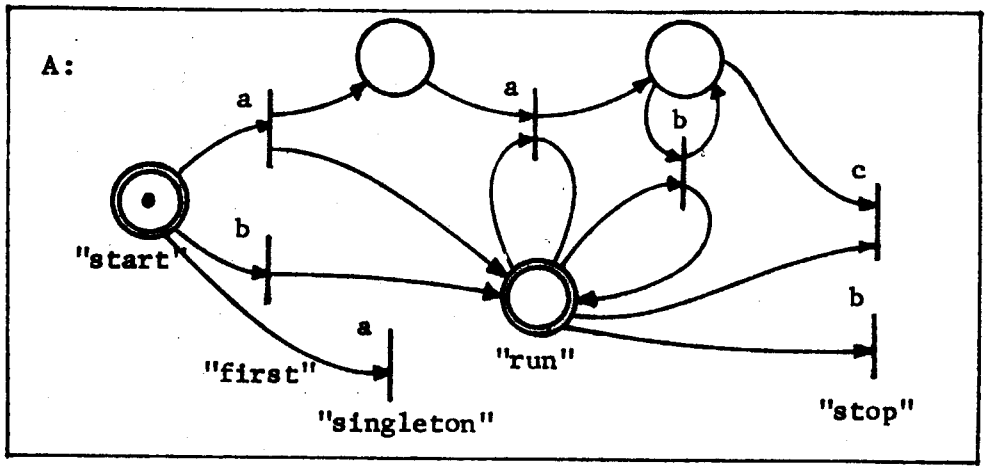
Proof:

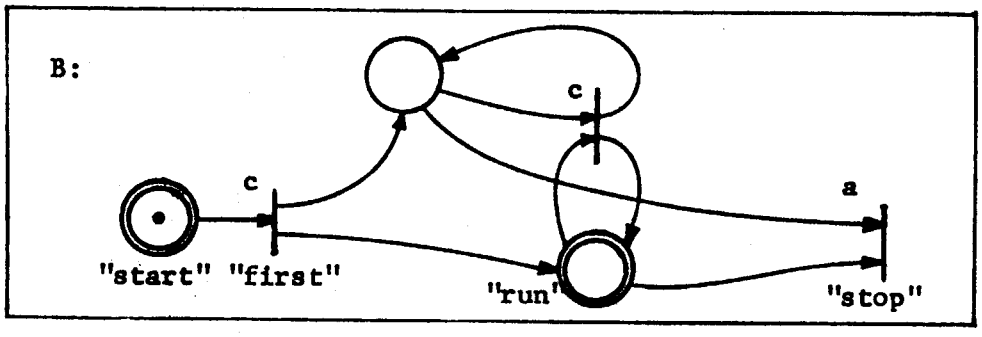
To establish the closure of $\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$, $\mathcal{L}_0^\lambda$ under union it is advantageous to use Labelled Petri Nets in Standard Form.

We recall that a net in Standard Form has a "start" place, which is the only place marked initially, and a standard final marking, the zero marking. Suppose we are given two nets A and B, generating S_A and S_B, respectively, as prefix label sequences ($\mathcal{L}$, $\mathcal{L}^\lambda$) or T_A and T_B as terminal label sequences ($\mathcal{L}_0$, $\mathcal{L}_0^\lambda$). We then construct a new net C by juxtaposing the two nets A and B, and by identifying the two "start" places; the resulting net has thus one "start" place and may have two "run" places. We note that if A and B are λ-free, then so is C. An example is shown in Figure 8.4.

The resulting net can easily be seen to satisfy the Standard Form conditions, and its label sequences are either those of A or those of B, depending on the first transition firing. The same applies to terminal sequences, since one portion of the net (corresponding to the language not simulated) retains its zero initial marking, and reaching the zero marking is thus the same as reaching the zero marking in the "active" portion of the net alone.

QED


A:
a
a
b
c
b
"start"
b
a
"first"
"run"
b
"singleton"
"stop"


B:
c
c
a
"start"
"first"
"run"
"stop"

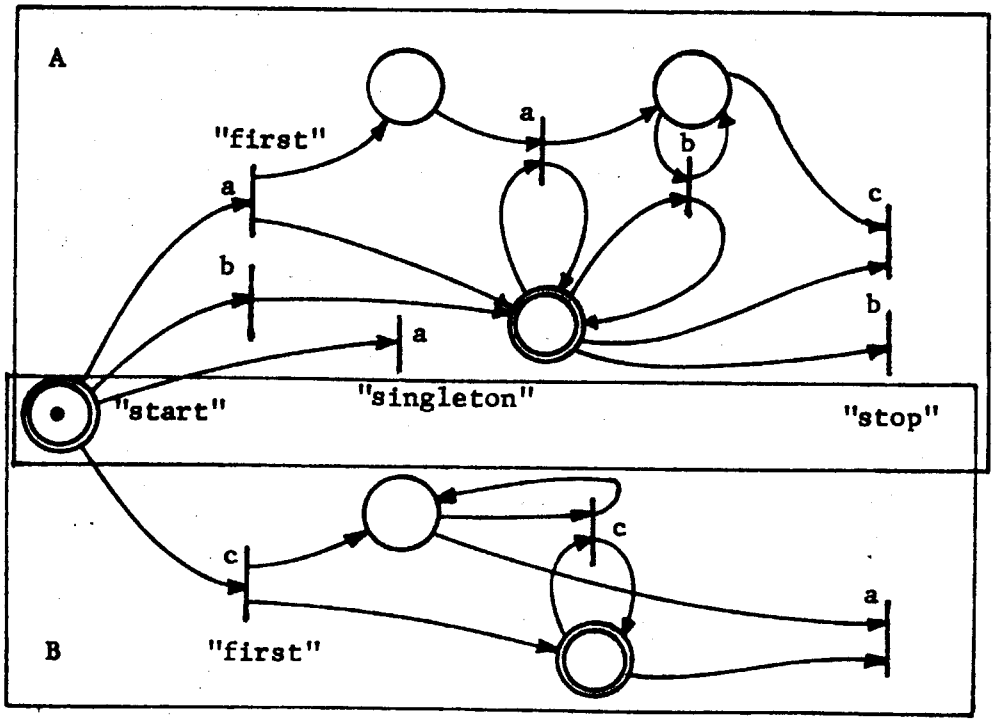

A
"first"
a
a
b
c
b
a
"start"
"singleton"
"stop"
c
c
a
B
"first"

Figure 8.4

Corollary 8.2:

The language families $\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$, $\mathcal{L}_0^\lambda$ are closed under union.

Theorem 8.4:

Given two Labelled Petri Nets A and B over the same alphabet, there exists a Labelled Petri Net C which is λ-free if both A and B are λ-free, and whose language is the intersection of that of A and that of B:

$$S_C = S_A \cap S_B$$
$$T_C = T_A \cap T_B$$

Proof:

Suppose we are given two Labelled Petri Nets A and B. Let us first consider the case of $\mathcal{L}^\lambda$-languages. We shall construct a Labelled Petri Net C such that its firing sequences correspond precisely to label sequences common to A and B. As a first step, we shall combine A and B in a way which forces them to generate the same strings. To do this, we juxtapose A and B (each with its initial marking). We add a new place π_0 and, for each symbol $a \in \alpha$ (the alphabet α is common to A and B), a new place π_a. Initially, π_0 has one token, all other π-places are blank.

As shown in Figure 8.5, we connect π_0 as an input to each labelled $t \in \Sigma_A$, and as an output to each labelled $t \in \Sigma_B$. For each symbol $a \in \alpha$, we connect π_a as an output to each a-labelled $t \in \Sigma_A$, and as an input to each a-labelled $t \in \Sigma_B$. λ-transitions in Σ_A or Σ_B are not connected to the π-places.

This arrangement enforces a strict alternation between labelled

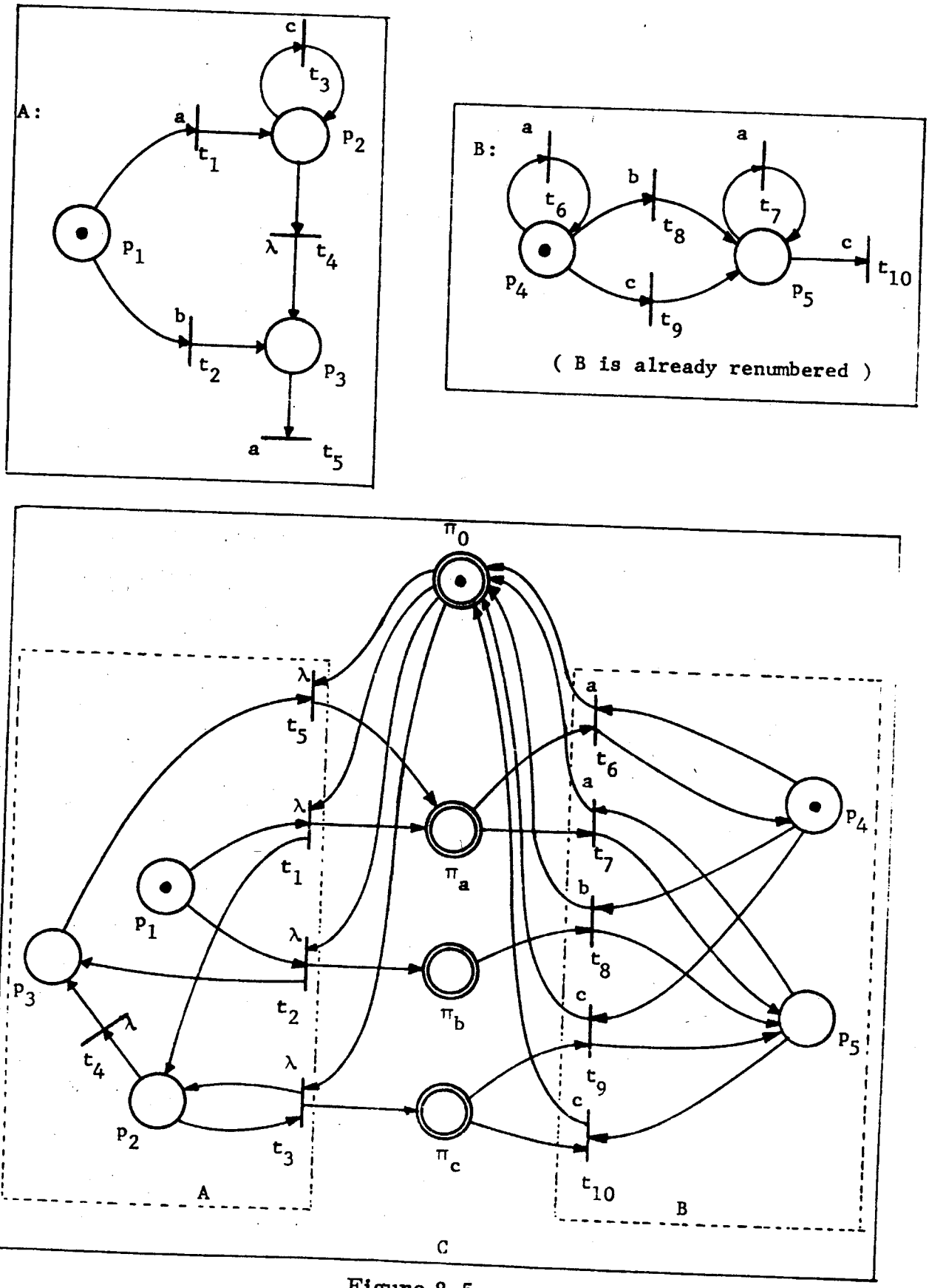

A:
c
t_3
a
t_1
p_2
p_1
λ
t_4
b
t_2
p_3
a
t_5
B:
a
t_6
b
t_8
a
t_7
c
t_{10}
p_4
c
t_9
p_5
(B is already renumbered)
π_0
λ
t_5
a
t_6
λ
t_1
π_a
a
t_7
p_4
p_1
b
t_8
λ
t_2
π_b
p_3
c
t_9
p_5
λ
t_4
λ
t_3
π_c
c
t_{10}
p_2
A
B
C

Figure 8.5

firings in A and B; λ-firings are not restricted. Each labelled firing in A is furthermore necessarily followed by a similarly labelled firing in B. In a sense, the π-places "remember" which symbol was last generated in A and enforce the repetition of this symbol in B before returning a token to π_0. As a result, the even-length label sequences of C are precisely those obtained by repeating twice each symbol from a label sequence that could be generated by both A and B. If we now remove the labels from all transitions in Σ_A, we will in effect erase the first symbol in each such repetition.

Our construction for the intersection of two $\mathcal{L}^\lambda$-languages consists thus of a Labelled Petri Net C, as described above, where all transitions in Σ_A have become λ-transitions. Then we have

$$S_C = S_A \cap S_B$$

In the case of two $\mathcal{L}_0^\lambda$-languages, both nets A and B are to reach a final marking. Let the final marking of the net C, constructed as above, be the juxtaposition of the two final markings, and one token in π_0 and zero tokens in the other π-places. Then it is clear that:

$$T_C = T_A \cap T_B$$

This proves the theorem for $\mathcal{L}^\lambda$ and $\mathcal{L}_0^\lambda$.

The situation is more complicated in the case of $\mathcal{L}_0$ and $\mathcal{L}$-languages. If the original nets A and B don't have λ-transitions, the net C resulting from the previous construction will have λ-transitions, namely all the Σ_A-transitions. However, each λ-firing will be immediately followed by a labelled firing. We will show how to combine these two firings into a single labelled firing.

Figure 8.6 shows the portion of the Labelled Petri Net C of Figure 8.5 that is connected to π_a.

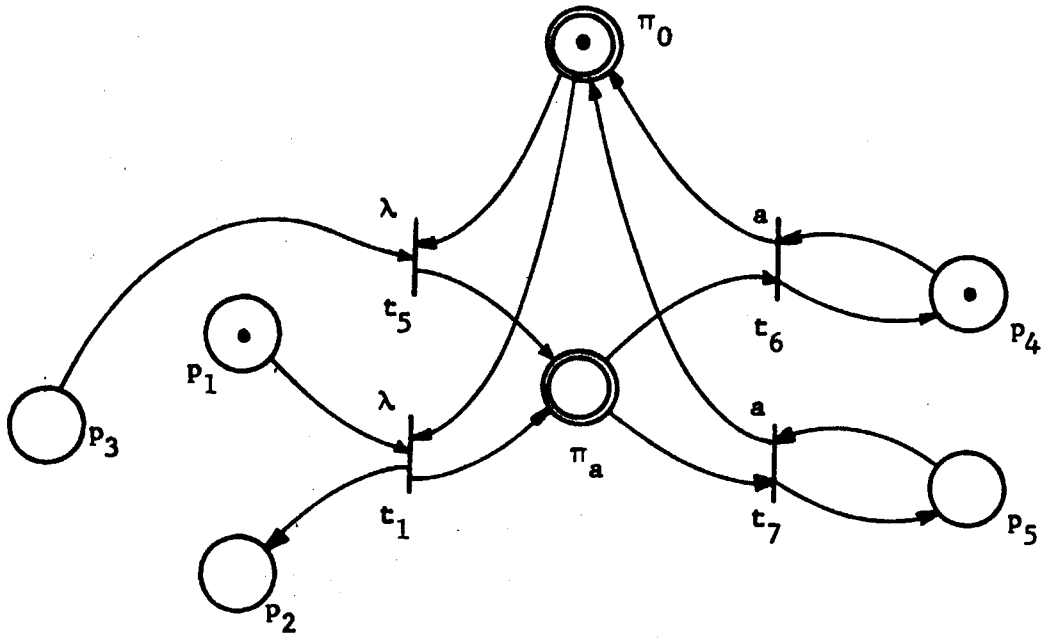


Figure 8.6

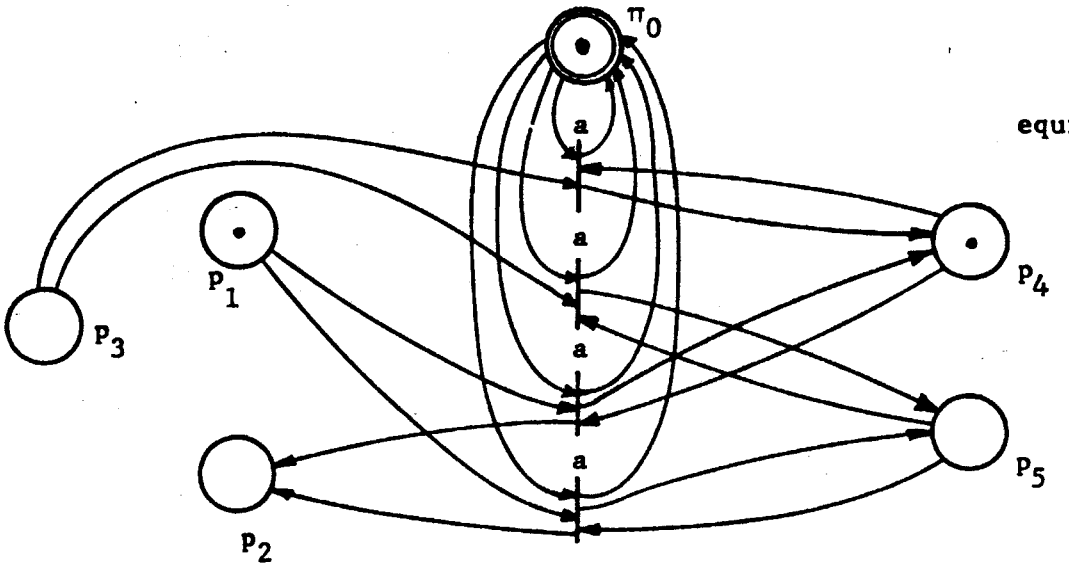


Figure 8.7

We see that any a-labelled firing (t_6 or t_7) is always preceded by a firing of t_5 or t_1. There are four (2 × 2) possible combinations: t_5t_6, t_5t_7, t_1t_6, t_1t_7, each generating the symbol a. Thus, we can eliminate the λ-transitions by replacing t_5, t_1, t_6, t_7 with four new a-labelled transitions which have the same effect as the combined firings t_5t_6, t_5t_7 ···; this eliminates place π_a.

This reduction can be applied to all other π-places, except π_0 which remains as a marked self-loop on all new (combined) transitions, like a "run" place.

Figure 8.7 shows the result of eliminating place π_a from the partial net of Figure 8.6.

This construction shows that, if both A and B are λ-free, we can transform C into a λ-free Labelled Petri Net whose $\mathcal{L}$ or $\mathcal{L}_0$-language is the intersection of the corresponding languages for A and B.

QED

From this we may conclude:

Corollary 8.3:

The families $\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$, $\mathcal{L}_0^\lambda$ are closed under intersection.

CHAPTER 9

PETRI NET LANGUAGES: MEMBERSHIP AND EMPTINESS PROBLEMS

9.1 Membership Problems

The membership problem is the question of deciding whether a given string can be generated by a given Labelled Petri Net. In the case of λ-free nets, the problem is trivial: Each label sequence can be generated by only a finite number of firing sequences, all easily obtained from the given label sequence. And it is clearly decidable whether a given firing sequence can be fired from the initial marking, and whether it reaches the final marking; just try to fire it! Thus:

Theorem 9.1:

The membership problem for $\mathcal{L}_0$-languages and for $\mathcal{L}$-languages is decidable.

In Hack [24] we show that $\mathcal{L}$- and $\mathcal{L}_0$-languages are effectively context-sensitive. This of course also implies the decidability of membership.

The case of $\mathcal{L}^\lambda$-languages is more interesting, because a given label sequence may correspond to infinitely many different firing sequences. But this case is also decidable.

Theorem 9.2:

The membership problem for $\mathcal{L}^\lambda$-languages is decidable.

Proof:

We shall reduce this problem to the coverability problem

(Chapter 3). Suppose we wish to decide whether a string, say "abac", is in the $\mathcal{L}^{\lambda}$-language of some labelled Petri Net A. Let us construct a Petri Net B which spells out the string "abac", as shown in Figure 9.1; it is a trivial Finite-State Machine. Place p_5 will receive a token if and only if the string "abac" is actually fired.

Now let us perform the intersection construction of Section 8.4 for the two nets A and B, as is indicated schematically in Figure 9.2.

Now the test place p_5 of B may eventually receive a token if and only if abac $\in S_A$. But it is decidable whether p_5 may ever get a token, by Corollary 3.1(d). Hence membership in S_A is decidable.

QED

The construction used in the preceding proof can also be used to test for membership in the $\mathcal{L}_0^{\lambda}$-language of a Labelled Petri Net such as A. But, in this case, the test string "abac" is in T_A only if it is possible to reach the final marking of A while getting a token into the test place p_5. In other words, we must test whether this combined final marking is reachable in the net of Figure 9.2: This is the Reachability Problem.

As it turns out, the Reachability Problem is also reducible to the membership problem for $\mathcal{L}_0^{\lambda}$-languages:

Theorem 9.3:

The membership problem for $\mathcal{L}_0^{\lambda}$-languages is recursively equivalent to the Reachability Problem.

Proof:

The reduction of the membership problem to the Reachability

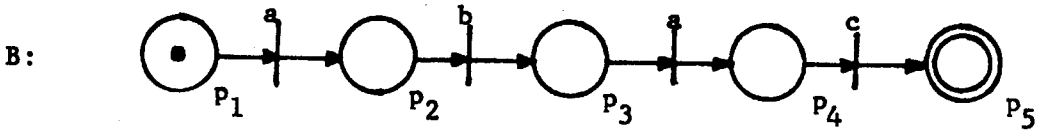


Figure 9.1

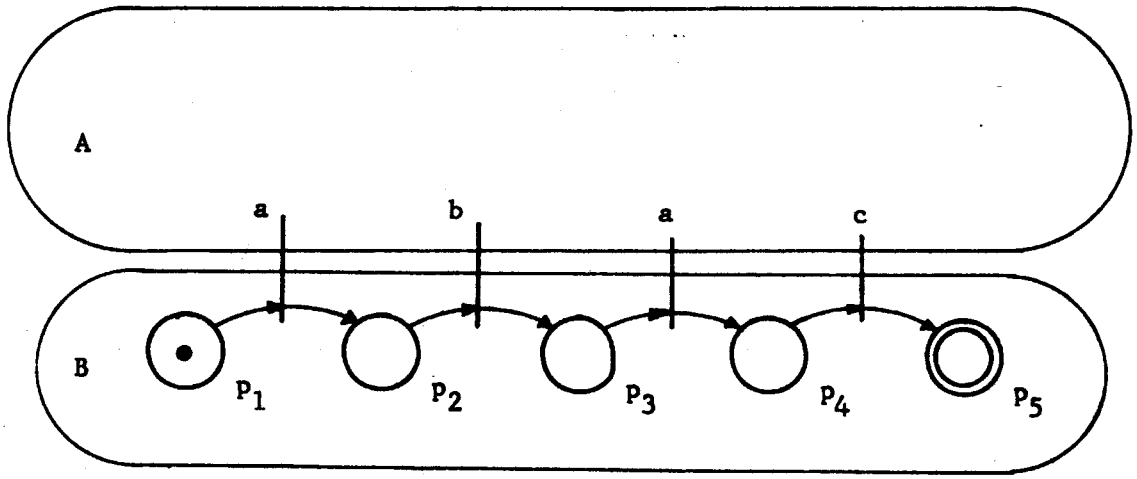


Figure 9.2

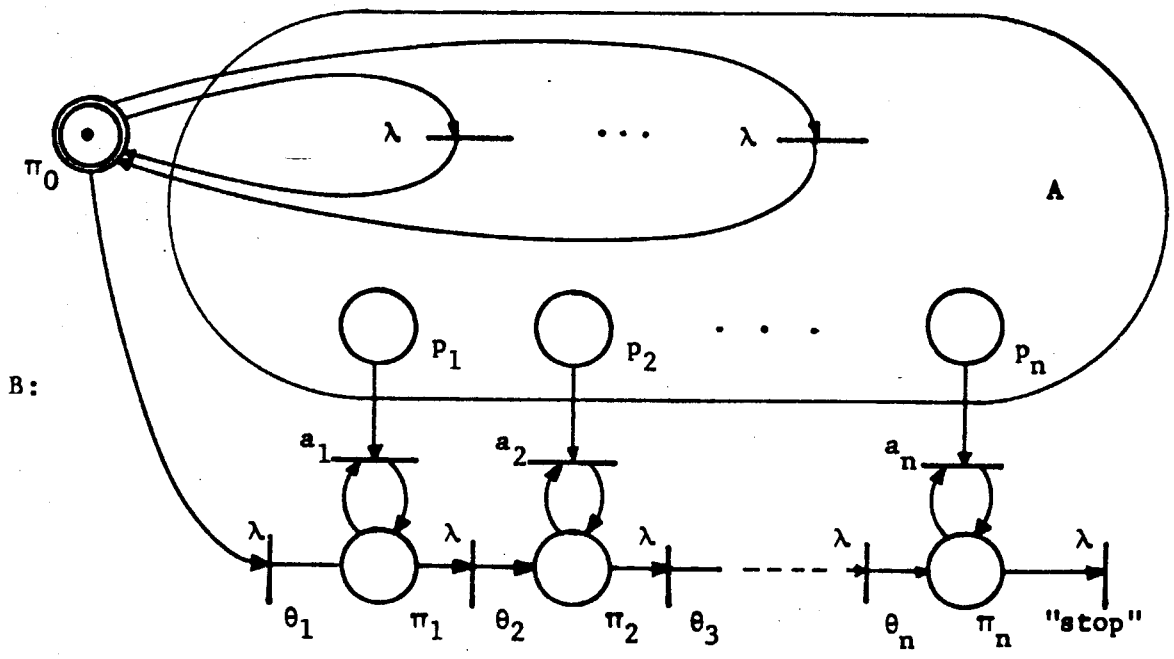


Figure 9.3

Problem was illustrated above by means of the same construction as in the previous proof.

To prove the reverse reducibility, we will show that $\mathcal{L}_0^\lambda$-languages can suitably encode Reachability Sets.

Let A be a GPN with places $p_1 \ldots p_n$ whose Reachability Set is to be encoded. Let B be the labelled GPN obtained by leaving all of A's transitions unlabelled (λ-transitions) and by adding a "run" place π_0 which self-loops on every transition in A, a set of n places $\pi_1 \ldots \pi_n$, a set of new λ-transitions $\theta_1 \ldots \theta_n$, a set of n labelled transitions with labels $a_1 \ldots a_n$, and a "stop" λ-transition. See Figure 9.3.

The initial marking M_0' consists of the initial marking M_0 of A for the old places $p_1 \ldots p_n$, one token in π_0 and zero tokens in $\pi_1 \ldots \pi_n$. The new λ-transitions θ_i transfer a token from π_{i-1} to π_i; "stop" removes a token from π_n. Each a_i-transition self-loops on π_i and removes one token from p_i.

While π_0 has its token, A fires as it did before being modified, and reaches some marking $M \in R_A(M_0)$ before θ_1 fires. Now the only way to reach the zero marking in the modified net B is to fire the firing sequence $\theta_1 a_1^{M(p_1)}\ \theta_2 a_2^{M(p_2)} \ldots \theta_n a^{M(p_n)}$ "stop". Therefore, the $\mathcal{L}_0^\lambda$-language of B encodes the reachability set of A as follows:

$$T_B(M_0', 0) = \left\{ a_1^{x_1}\, a_2^{x_2} \cdots a_n^{x_n} \;\middle|\; \langle x_1, \ldots, x_n \rangle \in R_A(M_0) \right\}$$

We may now use this encoding to test whether a marking is reachable in A: We test whether the corresponding string is in T_B.

QED

9.2 Emptiness Problems and Finiteness Problems

The Emptiness Problem asks whether the language generated by a

given Labelled Petri Net is the empty set, i.e. whether the net generates any strings at all.

This question is moot for prefix languages ($\mathcal{L}$ and $\mathcal{L}^{\lambda}$), since these always contain at least the empty string. And if we ask whether the prefix language contains strings other than λ, it is sufficient to ask whether it contains strings of length one, which is simply a finite number of instances of the decidable membership problem for prefix languages.

In the case of terminal languages ($\mathcal{L}_0$ and $\mathcal{L}_0^{\lambda}$), we ask whether the set of terminal strings $T_A(M_0, M_f)$ is empty for a given Labelled Petri Net $A = \langle N, \alpha, \Lambda \rangle$. But this is precisely the Reachability Problem for Petri Net N, because, regardless of the labelling Λ and the alphabet α, we have:

$$T_A(M_0, M_f) = \emptyset \Leftrightarrow T_N(M_0, M_f) = \emptyset \Leftrightarrow M_f \notin R_N(M_0)$$

Thus:

Theorem 9.4:

The emptiness problem for terminal Petri Net Languages ($\mathcal{L}_0$ and $\mathcal{L}_0^{\lambda}$) is recursively equivalent to the Reachability Problem.

Finally, let us mention the Finiteness Problem, where we ask whether a given Labelled Petri Net can generate infinitely many distinct label sequences.

For prefix languages we have:

Theorem 9.5:

Finiteness is decidable for prefix Petri Net Languages ($\mathcal{L}$ and $\mathcal{L}^{\lambda}$).

Proof:

Let A be a Labelled Petri Net. Then S_A is infinite iff it contains arbitrarily long label sequences. Let us add to the Petri Net a "count" place which receives a token from every labelled transition. This place is bounded iff the prefix language is finite. But boundedness is decidable (Theorem 3.4(b)).

QED

So far, not much is known about the finiteness of terminal languages. But we have:

Theorem 9.6:

The Reachability Problem is recursively reducible to the Finiteness Problem for terminal Petri Net Languages ($\mathcal{L}_0$ and $\mathcal{L}_0^\lambda$).

Proof:

In the light of Theorem 9.4 it is sufficient to reduce the Emptiness Problem for terminal languages to the corresponding Finiteness Problem.

Let A be a Labelled Petri Net in Standard Form. Add to it a labelled transition which self-loops on the "start" place. This does not affect the reachability of the final (zero) marking, but if a terminal label sequence exists, then arbitrarily long terminal sequences can be obtained by first firing the new transition arbitrarily often: The language of the modified net is infinite iff the language of the given net is non-empty.

QED

Summary of the results of this chapter (the abbreviations speak for themselves):

$\in \mathcal{L}$	decidable
$\in \mathcal{L}^{\lambda}$	decidable
$\in \mathcal{L}_0$	decidable
$\in \mathcal{L}_0^{\lambda}$	equivalent to RP
$\emptyset\ \mathcal{L}$	trivial
$\emptyset\ \mathcal{L}^{\lambda}$	trivial
$\emptyset\ \mathcal{L}_0$	equivalent to RP
$\emptyset\ \mathcal{L}_0^{\lambda}$	equivalent to RP
$\infty\ \mathcal{L}$	decidable
$\infty\ \mathcal{L}^{\lambda}$	decidable
$\infty\ \mathcal{L}_0$	RP reducible to it
$\infty\ \mathcal{L}_0^{\lambda}$	RP reducible to it

Note:

The results of Chapters 8 and 9 pertaining to the class $\mathcal{L}_0$ have been obtained independently by Peterson [52] in 1973.

CHAPTER 10

PETRI NET LANGUAGES: EQUIVALENCE AND INCLUSION PROBLEMS

10.1 Petri Net Languages can Encode Polynomial Graphs

We recall that the graph of a diophantine polynomial (non-negative integer coefficients) $P(x_1 \ldots x_n)$ is the set:

$$G(P) = \{\langle x_1, \ldots, x_n, y\rangle \in \mathbb{N}^{n+1} \mid y \leq P(x_1 \ldots x_n)\}$$

In Chapter 7 we showed that Petri Nets could encode polynomial graphs in terms of projected Reachability Sets. In this section we show how to encode polynomial graphs by means of $\mathcal{L}$-languages.

A natural way to encode sets of vectors over the integers into languages is to use the Parikh mapping:

Definition 10.1:

(a) The Parikh mapping for an alphabet $\alpha = \{a_1, \ldots, a_n\}$ is a function #: $\alpha^* \to \mathbb{N}^n$ such that # (w) is a vector whose i^{th} coordinate expresses the number of occurrences of symbol a_i in string w.

(b) The Parikh mapping is extended to languages in the natural manner:

$$L \subseteq \alpha^*: \ \#(L) = \{V \in \mathbb{N}^n \mid \exists w \in L: \ V = \#(w)\}$$

Now we shall prove that polynomial graphs can be encoded as the image under the Parikh mapping of an $\mathcal{L}$-type Petri Net Language. The coding is chosen such that there is exactly one language which encodes a given polynomial graph. Each vector in the polynomial graph corresponds to a set of strings, and the language is the disjoint union of these sets of strings.

Theorem 10.1:

For every diophantine polynomial P there exists a λ-free Labelled Petri Net A such that the $\mathcal{L}$-language of A encodes the graph of P via the Parikh mapping as follows:

S_A is the largest subset of the regular language $(a_1 + a_2 \ldots + a_n)^* (a_{n+1})^*$ such that

$$\#(S_A) = G(P)$$

Proof:

Let B be a Petri Net Weak Computer (λ-free and prefix) for the polynomial P, as described in Theorem 6.2. We construct the Labelled Petri Net A by adding transitions $\theta_1 \ldots \theta_n$, one for each "input" place of B. Each transition θ_i self-loops on the "begin" place and deposits tokens into the i^{th} "input" place, corresponding to variable x_i. All transitions of B (the "old" transitions) are labelled a_{n+1}, and each "new" transition θ_i is labelled a_i. Thus all transitions of A are labelled, and A is λ-free. (See Figure 10.1.)

The initial marking of A is the standard initial marking for B (as constructed in Chapter 6), with one token in the "begin" place and zero tokens in the n "input" places. One property of Petri Net B is that none of its transitions (the "old" transitions, labelled a_{n+1}) can fire until one of them has removed the token from the "begin" place, and that once this token has been removed, the "begin" place cannot become marked again. This means that all firings of the "new" transitions θ_i must precede all firings of the "old" transitions. Thus $S_A \subseteq \{(a_1 + a_2 + \cdots + a_n)^* (a_{n+1})^*\}$, and the only restriction is that the number of firings in B be no more than the value of the polynomial

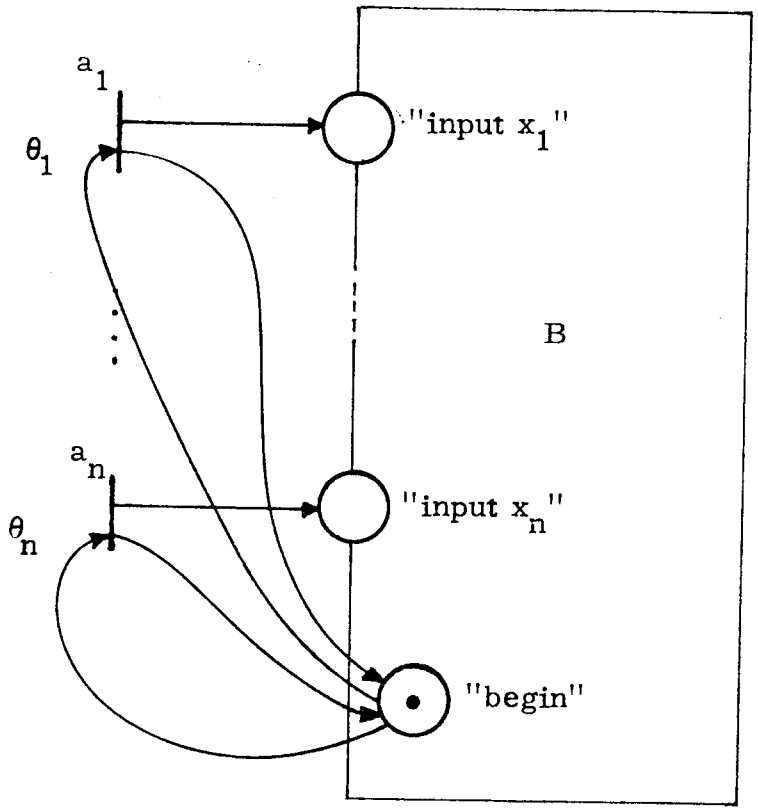

a 1
θ 1
"input x 1"
B
a n
θ n
"input x n"
"begin"
all "old" transitions are
labelled a n+1

Figure 10.1

P for the argument generated by the preceding θ-firings: S_A is indeed the largest language satisfying the sequencing requirement above such that $\#(S_A) = G(P)$

QED

Remark:

(a) The construction is not essentially different from that used in section 7.2 for projected Reachability Sets.

(b) Since every prefix language is also a terminal language (up to λ), we can also encode polynomial graphs as $\mathcal{L}_0$, $\mathcal{L}^\lambda$ or $\mathcal{L}_0^\lambda$-languages, except that this encoding leaves out the zero vector in the case of $\mathcal{L}_0$.

10.2 Undecidable Equivalence Problems

In this section we shall establish the undecidability of various Inclusion and Equivalence Problems by reducing the undecidable Polynomial Graph Inclusion Problem (PGIP) to them. The undecidability of the PGIP was established in Theorem 6.1.

Theorem 10.2:

The Equivalence and Inclusion Problems for Petri Net Languages ($\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$ and $\mathcal{L}_0^\lambda$) are undecidable.

Proof:

(a) The Inclusion Problem for $\mathcal{L}$-languages is undecidable: Let P and Q be two arbitrary diophantine polynomials, and ask whether $G(P) \subseteq G(Q)$ (The PGIP). Theorem 10.1 asserts the existence of two λ-free Labelled Petri Nets A and B such that:

$\#(S_A) = G(P)$

$\#(S_B) = G(Q)$

Both S_A and S_B are the largest subset of the regular language $(a_1 + \ldots + a_n)^* (a_{n+1})^*$ satisfying the polynomial constraints above.

Therefore $G(P) \subseteq G(Q) \Leftrightarrow S_A \subseteq S_B$, and the PGIP can be reduced to the Inclusion Problem for $\mathcal{L}$-languages (IP$\mathcal{L}$).

(b) By Theorem 8.2 there exist λ-free Labelled Petri Nets A' and B' whose terminal ($\mathcal{L}_0$-) language is the same as the $\mathcal{L}$-language of A and B, up to the empty string λ. Since the zero vector $\#(\lambda)$ is always in both G(P) and G(Q), we also have:

$$G(P) \subseteq G(Q) \Leftrightarrow T_{A'} \subseteq T_{B'}$$

Therefore the Inclusion Problem for $\mathcal{L}_0$-languages is also undecidable.

(c) Since $\mathcal{L} \subseteq \mathcal{L}^\lambda$ and $\mathcal{L}_0 \subseteq \mathcal{L}_0^\lambda$, the Inclusion Problem is undecidable for all Petri Net Languages ($\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$ and $\mathcal{L}_0^\lambda$-languages).

(d) Since all four families ($\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$ and $\mathcal{L}_0^\lambda$) are effectively closed under union (Theorem 8.3 and Corollary 8.2), the undecidability of inclusion implies the undecidability of equivalence for $\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$ and $\mathcal{L}_0^\lambda$.

QED

Now we shall investigate to what degree the language generated by a Petri Net depends on the structure of the net. We shall see that the generated language is quite sensitive to minor changes in the structure of the net. Indeed, it is undecidable in general whether such small changes in the net also induce a change in the language. This recalls a similar situation for Reachability Sets (Theorem 7.5).

Theorem 10.3:

It is undecidable whether the addition or removal of a given transition changes the language (prefix or terminal) of the net.

Proof:

Consider the Labelled Petri Net C of Figure 10.2. It contains two components A and B which are assumed to be in standard form, with respective "start" places p_1 and p_2. These places are connected to a new "start C" place p_3 by transitions t_1 and t_2, both labelled c, where c is a new symbol not in the alphabet of A or B. The initial marking of C consists of just one token in its "start" place, p_3.

We have:

$$S_C = \{\lambda\} \cup c \cdot (S_A \cup S_B)$$
$$T_C = c \cdot (T_A \cup T_B)$$

Let C' be obtained from C by removing t_2. Now B cannot be started, and we have:

$$S_{C'} = \{\lambda\} \cup c \cdot S_A$$
$$T_{C'} = c \cdot T_A$$

Hence:

$$S_{C'} = S_C \Leftrightarrow S_B \subseteq S_A$$
$$T_{C'} = T_C \Leftrightarrow T_B \subseteq T_A$$

In other words, the inclusion problem for the languages of A and B can be reduced to the equality problem for the languages of C and C', which differ only in the presence of transition t_2.

QED

Corollary 10.1:

It is undecidable whether any of the following changes affects the

C or C':

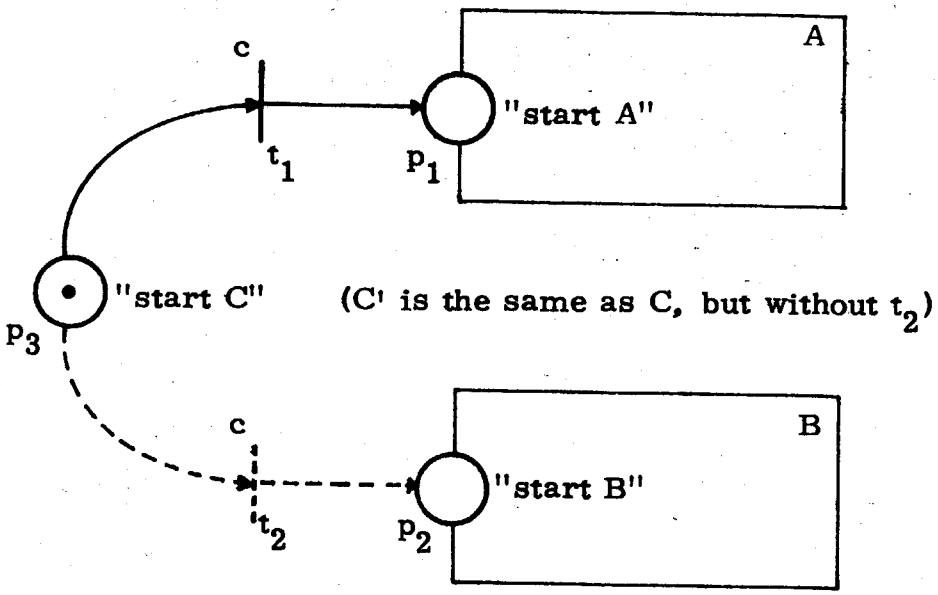


Figure 10.2

D:

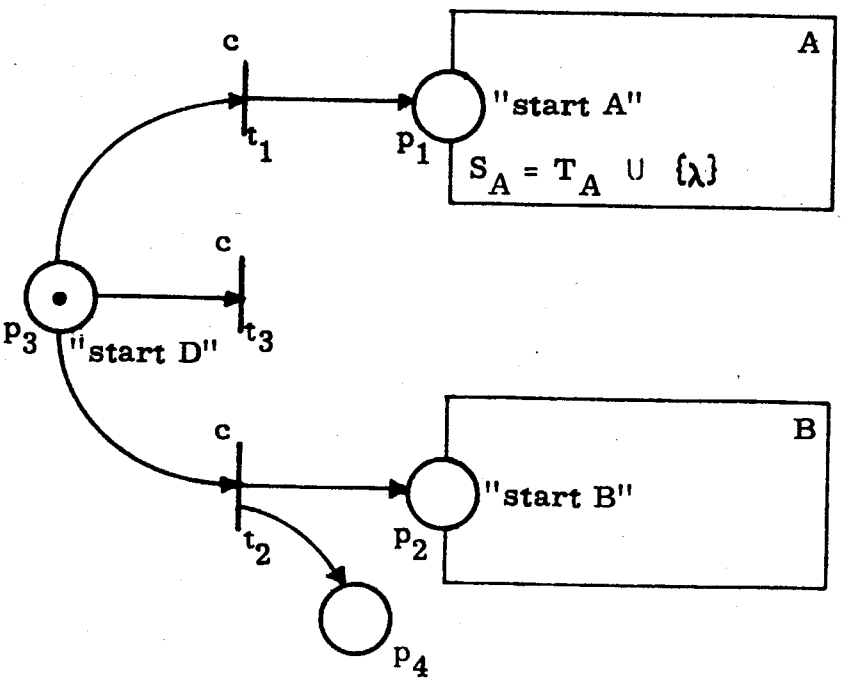


Figure 10.3

language generated by a Labelled Petri Net:

(a) changing the initial marking by one token

(b) (for $\mathcal{L}_0$, $\mathcal{L}_0^\lambda$): changing the final marking by one token

(c) removing a place

(d) removing or changing the size of an arc

(e) removing or changing a label on a transition

Proof:

All these cases can be transformed into the removal or addition of a transition, as in Theorem 10.3. We leave the details of the construction to the reader as an instructive exercise. (Cases (a) and (c) are discussed in Hack [24].)

Finally, we recall that every prefix language can be generated by a net whose prefix and terminal languages (up to λ in the case of $\mathcal{L}_0$) are the same. But in general, for a given Petri-Net, we cannot determine whether the prefix and the terminal language of the net are the same (up to λ in the case of $\mathcal{L}_0$):

Theorem 10.4:

It is undecidable whether every non-empty prefix label sequence of a Labelled Petri Net is also a terminal label sequence of the same Net.

Proof:

Consider the Labelled Petri Net D of Figure 10.3. It is obtained from C (in Figure 10.2) by adding an output place p_4 (initially

unmarked) to t_2 and a third transition t_3, also labelled c, which simply may remove the "start" token from p_3, the "start D" place.

Without loss of generality (Theorems 8.1 and 8.2) we choose A such that $S_A = T_A \cup \{\lambda\}$.

We have:

$$S_D = \{\lambda\} \cup c \cup c \cdot (S_A \cup S_B)$$

Since $\lambda \in S_A$, we can rewrite this as:

$$S_D - \{\lambda\} = c \cdot (S_A \cup S_B)$$

Let the final marking of D be the zero marking. Because of p_4, no terminal sequence can fire t_2. Thus:

$$T_D = c \cup c \cdot T_A = c \cdot (T_A \cup \{\lambda\})$$

Because of our choice of A this is also:

$$T_D = c \cdot S_A$$

In other words:

$$T_D = S_D - \{\lambda\} \Leftrightarrow S_B \subseteq S_A$$

and the undecidability follows from the undecidability of the inclusion problem for $\mathcal{L}$-languages.

QED

10.3 The Equivalence Problem for Sets of Firing Sequences

The sets of firing sequences $S_N(M_0)$ or terminal firing sequences $T_N(M_0, M_f)$ can of course be regarded as Petri Net Languages of type $\mathcal{L}$ and $\mathcal{L}_0$ respectively, by considering the Petri Net N to be a Labelled Petri Net where the alphabet is the set of transitions, and each transition is its own label. In Hack [24] we call such Labelled Petri Nets, where all transitions have distinct labels, Free-Labelled Petri Nets, and their languages, the Free Petri Net Languages, of type $\mathcal{L}^f$ (prefix) or $\mathcal{L}_0^f$ (terminal).

The Equivalence Problem for Sets of Firing Sequences is the problem of deciding, given two Petri Nets N and N' and a bijective correspondence between their transitions (for purposes of comparing firing sequences), whether $S_N(M_0) = S_{N'}(M'_0)$ or $T_N(M_0, M_f) = T_{N'}(M'_0, M'_f)$. In terms of Petri Net Languages, it is the Equivalence Problem for Free Petri Net Languages (of type $\mathcal{L}^f$ or $\mathcal{L}_0^f$). We also have the corresponding Inclusion Problems. We shall show that the Inclusion Problems for Sets of Firing Sequences are reducible to the Reachability Problem. If it turns out that this is decidable, then this will imply that Free Petri Net Languages are essentially less powerful than Petri Net Languages in general. (It is already known that some particular $\mathcal{L}_0$-languages are not Free.)

Theorem 10.5:

> The Inclusion and Equivalence Problems for the Sets of all Firing Sequences (for $\mathcal{L}^f$-languages) are reducible to the Reachability Problem.

Proof:

It is sufficient to reduce the Inclusion Problem to a problem equivalent to the Reachability Problem, such as the Sub-Liveness Problem (SLP) for a given transition.

Let two Petri Nets A and B be given, each with its initial marking, and let their sets of transitions be $\{t_1^A \ldots t_n^A\}$ and $\{t_1^B \ldots t_n^B\}$. For the bijective pairing $t_i^A \leftrightarrow t_i^B$, we ask whether $S_A \subseteq S_B$.

We connect the two nets together in a new net C, as shown in Figure 10.4. The construction is based on that for the intersection of two languages (Figure 8.5): there is a control place π_0 (initially

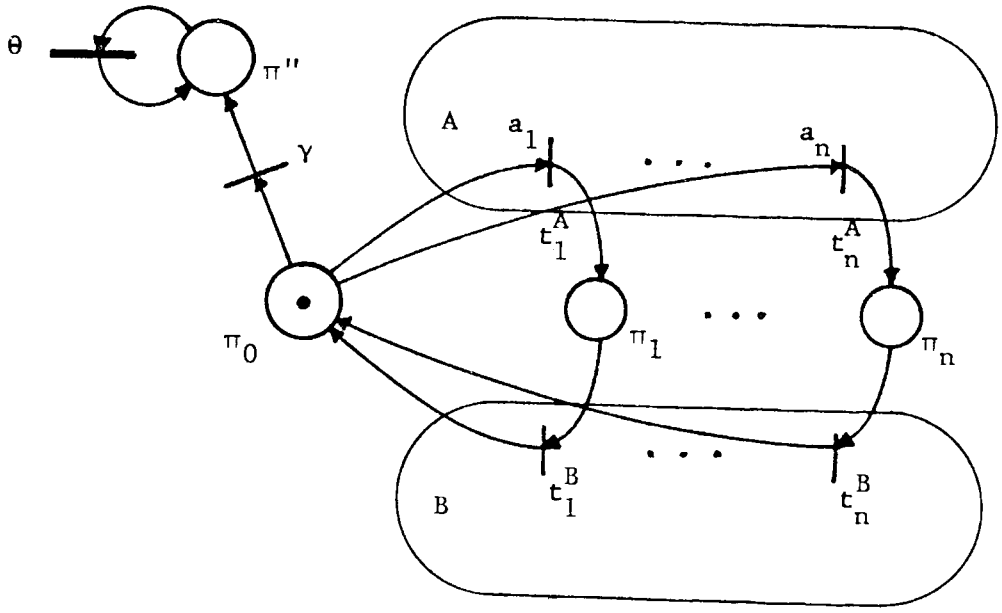


Figure 10.4

marked with one token) and "symbol-remembering" places π_i, one for each pair of corresponding transitions t_i^A, t_i^B. If all π_i, $1 \leq i \leq n$, are empty, then the firing sequence fired so far in A has been exactly echoed by B.

The token in π_0 can also be transferred to place π'' via transition γ, and permanently enable the test transition θ, which self-loops on π'.

It appears that the only markings of the new net C at which θ is not potentially firable are markings with a token in some π_i, $1 \leq i \leq n$, at which the corresponding transition t_i^B is not firable. Such a marking is reachable if and only if there exists a firing sequence σ in A, ending in t_i^A, which cannot be echoed completely by B: $\sigma \in S_A - S_B$, and $S_A \not\subseteq S_B$.

Thus θ is live iff $S_A \subseteq S_B$, and the inclusion problem for A and B can be reduced to the SLP for θ in C.

QED

The inclusion problem for terminal firing sequences ($\mathcal{L}_0^f$) will also be shown to be reducible to the Reachability Problem. But in this case, the RP is also reducible to the equivalence problem: the RP for $M_f \in R_N(M_0)$ is the equivalence problem $T_N(M_0, M_f) = \emptyset$, because it is trivial to find a Petri Net N' such that $T_{N'} = \emptyset$. We have:

Theorem 10.6:

The Inclusion and Equivalence Problems for Sets of Terminal Firing Sequences are recursively equivalent to the Reachability Problem.

Proof:

We just mentioned the reducibility of the RP to the inclusion and equivalence problem. To show the reducibility of the inclusion (and thus also equivalence) problem to the RP, we reduce it to the SLP for a given transition θ, as in the preceding proof.

Let two Petri Nets A and B be given, with their initial markings, and let their final markings be $M_f(A)$ and $M_f(B)$, respectively. We construct a new net C as shown in Figure 10.5. It contains the construction of Figure 10.4, plus the following:

- a place π' which records the presence of a token in some "symbol-remembering" place π_i, $1 \leq i \leq n$.
- for each component (A or B, indicated by subscript), a mechanism for testing whether the corresponding final marking $M_f(A)$ or $M_f(B)$ has been reached. This consists of a transition θ_A which removes exactly $M_f(A)$ from the places of A, a place π_A which gets a token from θ_A, and a set of transitions α, one per place of A, which can remove a token from π_A only if the corresponding place of A still contains a token. A place π'_A, initially with one token, prevents θ_A from firing more than once. If A has reached a marking $M(A) = M_f(A)$, then a firing of θ_A is possible and it disables all α-transitions.
- The final-marking detectors are interconnected as follows:

  θ_A removes a token from π'.

  The α-transitions return this token to π'.

  θ_B removes a token from π_A.

  The β-transitions return this token to π_A.
- Finally, a transition γ' removes a token from each of π', π'_A and

π_B and drops a token in π'', whereas transition γ'' transfers a token directly from π_B to π''.

This construction works as follows. We start by firing only transitions in A, echoed in B. If at any time we fire γ' instead of some t_i^B, we get a token stuck in π', and θ cannot cease to be firable: π_A is empty, and θ_A is disabled.

If we fire γ, we have previously completed and echoed some firing sequence $\sigma \in S_A \cap S_B$, reaching markings M(A) and M(B) in A respectively B.

The token now in π' may be stuck there if $M(A) \not\geq M_f(A)$, because then θ_A is not firable. Hence $\sigma \notin T_A$ and θ cannot cease to be firable. If, however, $M(A) \geq M_f(A)$, we fire θ_A. The token can escape from π_A if some α is firable ($M(A) > M_f(A)$) or if θ_B is firable ($M(B) \geq M_f(B)$). If neither is firable, then $M(A) = M_f(A)$ and $M(B) \not\geq M_f(B)$, i.e. $\sigma \in T_A - T_B$. The token is stuck in π_A and θ is not potentially firable: θ was not live at the initial marking.

If we did leave π_A by firing some α, the token returns to π'' and is now stuck there, because θ_A has already fired (π'_A is empty), and θ is permanently firable.

If we did leave π_A by firing θ_B, we have $M(B) \geq M_f(B)$. The token cannot get stuck in π_B because via γ'' it can return to π'' and get stuck there, with θ permanently firable. But if some β is firable (implying $M(B) > M_f(B)$, i.e. $\sigma \notin T_B$), the token may return to π_A. Since θ_B has already fired, the token is stuck in π_A unless some α is firable and returns the token to π'' where it must stay. Again, the token is stuck in π_A only if $M(A) = M_f(A)$, i.e. $\sigma \in T_A - T_B$.

This description exhausts all possible firing sequences, and is

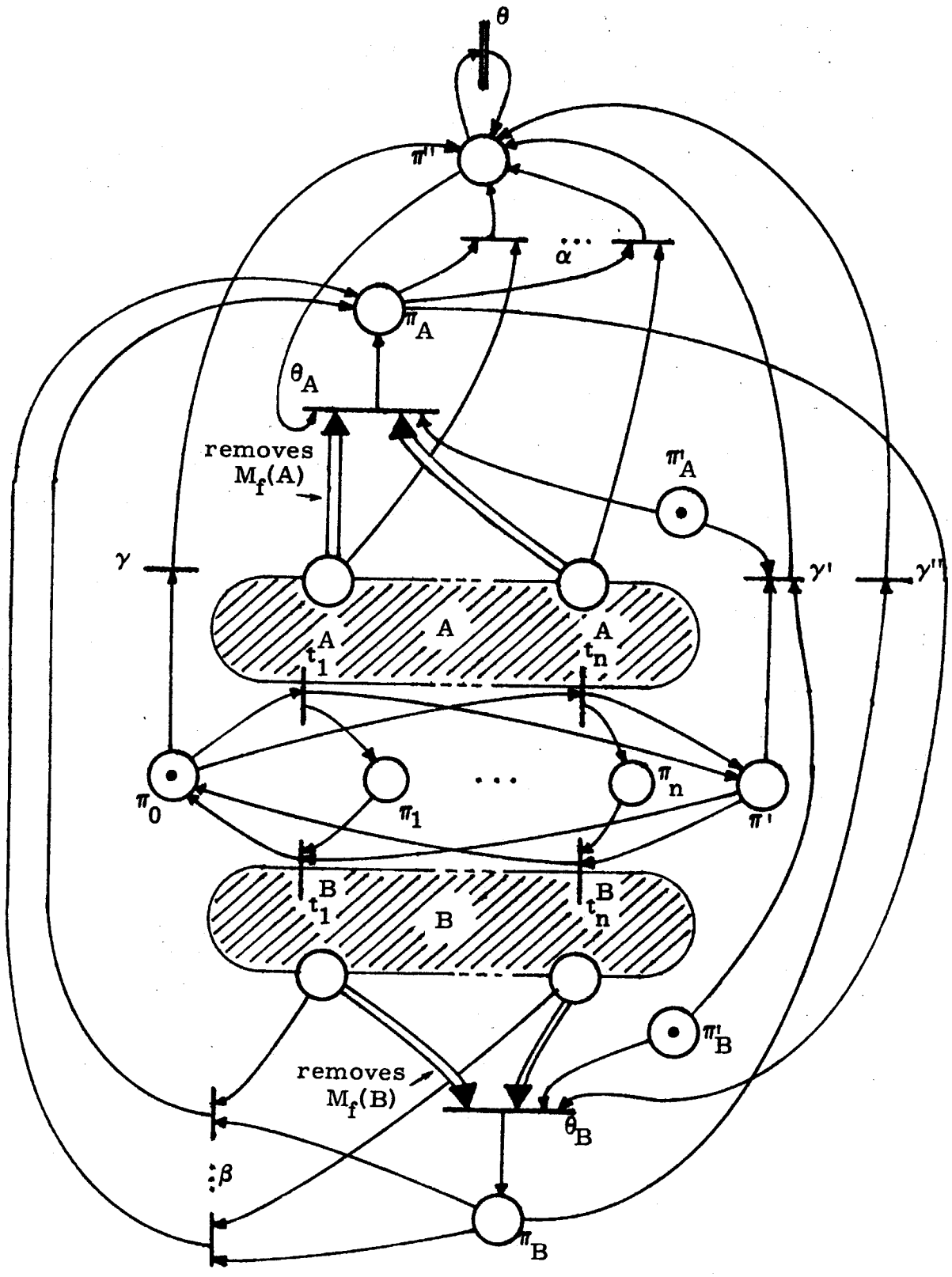

θ
π'
α
π_A
θ_A
removes
M_f(A)
π'_A
γ
γ'
γ''
A
A
A
t_1
t_n
π_0
π_1
π_n
π'
B
B
B
t_1
t_n
π'_B
removes
M_f(B)
θ_B
β
π_B

Figure 10.5

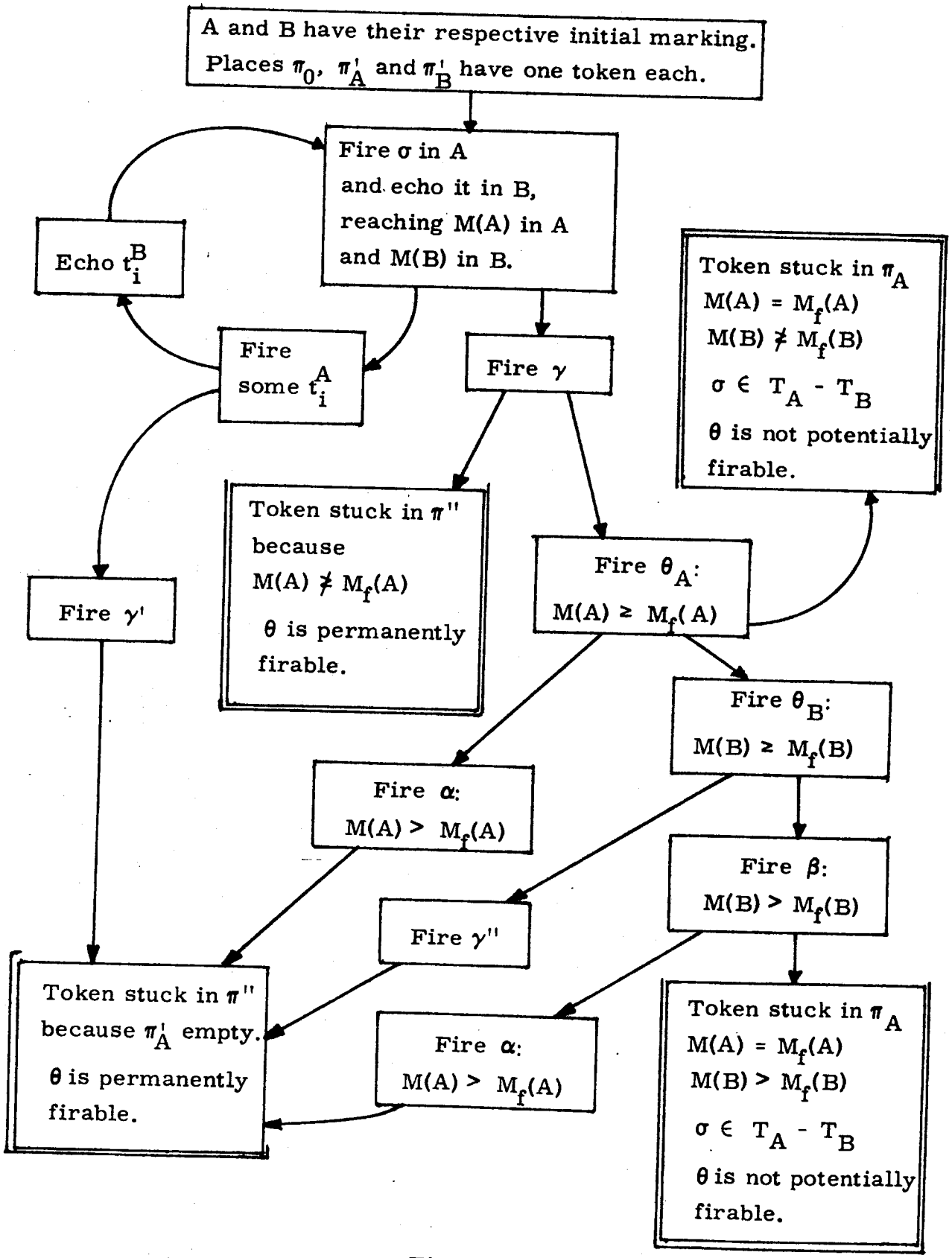

A and B have their respective initial marking.
Places π_0, π'_A and π'_B have one token each.
Fire σ in A
and echo it in B,
reaching M(A) in A
and M(B) in B.
Echo t_i^B
Fire
some t_i^A
Fire γ
Token stuck in π_A
$M(A) = M_f(A)$
$M(B) \neq M_f(B)$
$\sigma \in T_A - T_B$
θ is not potentially
firable.
Token stuck in π''
because
$M(A) \not\geq M_f(A)$
θ is permanently
firable.
Fire θ_A:
$M(A) \geq M_f(A)$
Fire γ'
Fire θ_B:
$M(B) \geq M_f(B)$
Fire α:
$M(A) > M_f(A)$
Fire β:
$M(B) > M_f(B)$
Fire γ''
Token stuck in π''
because π'_A empty.
θ is permanently
firable.
Fire α:
$M(A) > M_f(A)$
Token stuck in π_A
$M(A) = M_f(A)$
$M(B) > M_f(B)$
$\sigma \in T_A - T_B$
θ is not potentially
firable.

Figure 10.6

summarized in Figure 10.6. It appears that a θ-dead marking (where θ is not potentially firable) can be reached if and only if $\sigma \in T_A - T_B$: Transition ϑ is live in the new net iff $T_A \nsubseteq T_B$.

QED

Summary of the decidability results of this chapter:

$IP\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$, $\mathcal{L}_0^\lambda$)	undecidable
$EP\mathcal{L}$, $\mathcal{L}_0$, $\mathcal{L}^\lambda$, $\mathcal{L}_0^\lambda$)	
$IP\mathcal{L}^f$, $EP\mathcal{L}^f$ (firing sequences):	reducible to RP
$IP\mathcal{L}_0^f$, $EP\mathcal{L}_0^f$ (terminal firing sequences):	equivalent to RP

CHAPTER 11

CONCLUSION: OPEN QUESTIONS AND CONJECTURES

11.1 Is Reachability Decidable?

The decidability questions considered in this thesis fall into three classes: decidable problems, problems equivalent (or reducible) to the Reachability Problem, and undecidable problems. One might call these the three Petri Net "degrees of unsolvability". The decidability of the Reachability Problem is of course the major open problem in this area. Its resolution will not only settle most questions considered in this thesis, it will have repercussions in several fields outside of Petri Net theory, because of the connections mentioned in Chapter 1.

Problems equivalent to the Reachability Problem typically involve the existence of a firing sequence satisfying certain effectively testable conditions, such as reaching a given marking (RP) or some t-dead marking (LP). Now, we can enumerate firing sequences of increasing length and check whether they satisfy the required conditions. The question is: How long do we have to search before we may convince ourselves that no such firing sequence exists? In other words, is it possible to put a bound on the length of the shortest firing sequence satisfying the conditions, if such a sequence exists? We would expect such a bound to depend on the size of the Petri Net and of its initial marking.

It is not difficult to construct a sequence of Petri Nets N_i ($i = 1, 2, \ldots$) of size $k \cdot i$ (measured by the total number of arcs, i.e. the sum $\Sigma (F(p,t) + B(p,t))$ over all places and transitions) and with initial markings of x tokens, such that the shortest firing sequence reaching the zero marking is of length proportional to $x \cdot 2^i$. In fact, a recent

construction by Lipton [39] can be adapted to Petri Nets to generate a sequence of nets N_i such that the shortest zero-reaching sequence is of length proportional to $x \cdot 2^{2^i}$. This very rapid growth suggests that the Reachability Problem, if decidable, may still be quite complex. Indeed, a direct consequence of Lipton's result is that the complexity of the RP is at least "exponential-space-hard" [39].

In the preceding discussion, we have intentionally separated the size of the initial marking (x) from the size of the net ($k \cdot i$). This is because of the following important observation:

> Every Petri Net we have ever constructed, no matter how contrived, has the property that the length of the shortest zero-reaching sequence (or of the shortest killing sequence) is bounded by a linear function of the size of the initial marking.

The failure to find a counterexample has never proved anything, but it can provide a strong hint. There seems to be a pattern among the ways the various Petri Nets allow a killing sequence (a firing sequence which reaches some t-dead marking) of length proportional to the initial marking, and it is not unlike the pattern of firing sequences used to cover a marking of a given size, as in Chapter 3. A detailed analysis of the construction of coverability trees shows that, in a given Petri Net, there exists a constant K such that if a marking M (of size $|M|$) is coverable, a covering marking can be reached by a firing sequence of length less than $K \cdot |M|$. For a sequence of nets N_i of size proportional to i, the corresponding constant K_i may grow like 2^{2^i} (again using Lipton's constructions), and the best known upper bound appears to be Ackermann's function of i (cf. Hack [24]).

Our conjecture with regard to the Reachability Problem is then:

Conjecture:

The Reachability Problem is decidable, because for a Petri Net of size y with an initial marking of size x we can determine a constant K_y such that the zero marking is reachable iff it can be reached by a firing sequence of length less than $K_y \cdot x$.

11.2 Some Sufficient Conditions for the Undecidability of RP

Given the versatility of possible Petri Net constructions and the surprising complexity of some of them, it is not unreasonable to suspect the undecidability of the Reachability Problem. Some colleagues believe the problem to be undecidable, and in the course of this research the author's opinion has oscillated a few times between decidability and undecidability.

The undecidability results we have proved so far rely on a suitable encoding of a polynomial graph G(P). Suppose we could similarly encode the complement of a polynomial graph:

$$\overline{G}(P) = \mathbb{N}^{n+1} - G(P) = \{\langle x_1 \ldots x_n, y\rangle \in \mathbb{N}^{n+1} \mid y \geq 1 + P(x_1 \ldots x_n)\}$$

The PGIP can then be reformulated as the emptiness problem for the intersection of a polynomial graph and the complement of a polynomial graph:

$$G(P) \subseteq G(Q) \Leftrightarrow G(P) \cap \overline{G}(Q) = \emptyset$$

Since Petri Net Languages are closed under intersection (Corollary 8.3) and since their emptiness problem is reducible to RP, we can assert:

> A sufficient condition for the undecidability of RP is the possibility of encoding the complement of an arbitrary polynomial graph as an $\mathcal{L}_0$-language by the mapping used in Theorem 10.1 for

polynomial graphs.

A direct corollary of the preceding condition is:

> A sufficient condition for the undecidability of RP is the closure under complementation of the Petri Net Language family $\mathcal{L}_0^\lambda$ (or the inclusion in $\mathcal{L}_0^\lambda$ of the complementation closure of $\mathcal{L}$, $\mathcal{L}^\lambda$ or $\mathcal{L}_0$).

It is also possible to use the closure properties of the family $\mathcal{L}_0^\lambda$ (see Hack [24]) to show that the complement of an arbitrary polynomial graph can be encoded (as in Theorem 10.1) in $\mathcal{L}_0^\lambda$ iff the language $\{a^x b^y \mid y \geq x^2\}$ is an $\mathcal{L}_0^\lambda$-language.

> A sufficient condition for the undecidability of RP is the existence of the language $\{a^x b^y \mid y \geq x^2\}$ in $\mathcal{L}_0^\lambda$.

Finally, a Petri Net which generates the language $\{a^x b^y \mid y \geq x^2\}$ can be modified into a net where the length of the shortest zero-reaching sequence is proportional to the square of the size of the initial marking. Compare this with the conjecture of the previous section!

11.3 Decidability Questions for Restricted Classes of Petri Nets

Although we defined both Generalized Petri Nets (GPN) and Restricted Petri Nets (RPN) in Chapter 2, all theorems in this thesis are true for GPN's as well as for RPN's, and the only proofs that need to be (slightly) modified are those of Lemma 4.4 and Theorems 7.3, 8.1 and 10.6.

This is why we simply say "Petri Net" instead of "Ordinary Petri Net", GPN, or RPN.

The more commonly used "Ordinary Petri Nets" (section 2.2) have been subdivided into a number of classes in the literature, such as Simple Nets, Free-Choice Nets, Marked Graphs or State Machines. Definitions and further references can be found in Hack [18].

State Machines, and, in a more general sense, all bounded Petri Nets, behave like classical Finite-State Automata, and all problems considered in this thesis are decidable for this class of Petri Nets.

Marked Graphs are a subclass of the Persistent Petri Nets, and their mathematical properties have been extensively studied (Commoner [7]). Again, all problems are decidable, although languages generated by Marked Graphs have not been studied much (see, for example, Baker [3]).

The Liveness Problem is decidable for Free-Choice Nets, because liveness in these nets depends only on simple structural properties, by a Theorem of Commoner (see Hack [18]). On the other hand, all constructions for Reachability and Equivalence can be carried out using Free-Choice Nets, and thus have the same status as for GPN's.

Simple Nets include the Free-Choice Nets and have the same Reachability and Equivalence Problems as GPN's. Although there is a simple sufficient condition for liveness in Simple Nets, no useful necessary condition is known, and the Liveness Problem is unsettled.

We have already mentioned (section 5.2) that the Liveness Problem is decidable for Persistent Nets, but Reachability and Equivalence are unsettled.

Because of the remark following Theorem 5.2, the Reachability Problem and the Equivalence Problem for Live Nets are the same as for GPN's.

Finally, let us mention the interesting class of Symmetric Nets, where for each transition t there is a "reverse" transition t' such that $F(p, t) = B(p, t')$ and $B(p, t) = F(p, t')$. In Symmetric Nets every potentially firable transition is live, so liveness is decidable. Reachability is decidable because Symmetric Nets are closely related to

commutative semigroups (Cardoza [6]).

Let us also mention in a few words some further generalizations (as opposed to restrictions) of Petri Nets. There has been some controversy about the modelling power of Petri Nets: Can they - or can they not - represent "all" synchronization problems (Patil [51]; Parnas [47]; Habermann [16])? It is implicit in his paper [16] that Habermann could only be satisfied by a formalism which has the power of Turing Machines. A more reasonable approach is to check a number of classical synchronization problems. It then appears that all practical synchronization problems which Petri Nets fail to solve involve the notion of priority: certain things can happen only if no things of higher priority can happen. These problems can be solved if we modify the firing rule of Petri Nets to include zero-testing transitions or arcs, which are enabled only if their input place contains no tokens. The inability of Petri Nets to test for zero (for several reasonable definitions of "zero-testing") follows from the containment property (Theorem 2.1) of Petri Nets (Keller [34], Kosaraju [37]). The inclusion of zero-testing arcs has been proposed by Agerwala [2], among others. By comparing the resulting "Inhibitor Nets" with Minsky's Program Machines (cf. section 1.3), it appears that these nets have the full power of Turing Machines. We have shown (Hack [24]) that priority firing rules have exactly the same effect, and that "Inhibitor Nets" and "Priority Nets" can be simply transformed into each other. It is not difficult to see that for these "improved" Petri Nets most problems treated in this thesis, such as boundedness and reachability, are undecidable.

11.4 Conclusion

The subject of Decidability Questions for Petri Nets has by no means

been exhausted. There are a number of problems which seem to be more difficult than Reachability (i.e. reducibility is suspected in only one direction), for which we have not been able to prove their undecidability: Is the Reachability Set strongly connected, i.e. is every reachable marking also reachable from every other reachable marking? Does there exist a live initial marking? Does the Reachability Set contain some live marking? Is every marking which agrees with a given submarking reachable ("strong" submarking reachability; see the discussion following Definition 2.18)? These problems belong, for the time being, in a fourth Petri Net "degree of unsolvability", between RP and undecidability.

The author's original goal was to settle the decidability of Reachability, and to develop insights into the complexities and possibilities of Petri Nets as a mathematical model. The first goal proved to be too ambitious; we only found relative reducibilities, as well as a number of new undecidability results (the various equivalence problems). We leave it to the reader to assess the fulfillment of our second goal, and wish her or him a successful investigation of the remaining open problems.

APPENDIX

SETS OF VECTORS OVER THE AUGMENTED INTEGERS Ω

In this appendix we shall prove various results presented in section 2.6 concerning the properties of the complete lattice of vectors over the augmented integers Ω and its non-complete sublattice of vectors over the non-negative integers $\mathbb{N}$. Completeness in this sense means that every subset of Ω^r has a least upper bound (lub) with respect to the partial order $\leq$ for vectors. Let us first recall the relevant definitions (the numbering is as in Chapter 2):

Definition 2.22:

The augmented set of non-negative integers is the set $\Omega = \mathbb{N} \cup \{\omega\}$, where ω is an element which behaves like an integer larger than any given integer and is characterized by:

$$\forall n \in \mathbb{N}: \ \omega \neq n \ \& \ \omega \geq n \ \& \ \omega + n = \omega \ \& \ \omega - n = \omega \ \&$$
$$\omega + \omega = \omega - \omega = \omega$$

Definition 2.25:

A chain $C \subseteq \Omega^r$ is a subset which is totally ordered under $\leq$, i.e. $C = \{V_0, V_1, \ldots V_j, \ldots\}$ and $V_{j+1} > V_j$ (for all j if C is infinite, or up to $j = |C| - 2$ if C is finite).

Definition 2.26:

A subset $A \subseteq \Omega^r$ is chain-complete iff, for every chain $C \subseteq A$, its least upper bound is an element of A: lub $(C) \in A$.

Definition 2.27:

A subset $A \subseteq \Omega^r$ is monotone iff $\forall V \in A: \; V' \leq V \Rightarrow V' \in A$. *)

Definition 2.28:

For a set $A \subseteq \Omega^r$, its set of maximal elements $\hat{A}$ is the set:

$$\hat{A} = \{V \in A \mid \nexists V' \in A: \; V' > V\}$$

Definition 2.29:

For a set $A \subseteq \Omega^r$, its chain-completion A^c is the smallest chain-complete set containing A.

The following Theorem forms the basis of many finiteness proofs:

Theorem 2.4:

(a) Every infinite subset of Ω^r contains an infinite chain.

(b) Every set of mutually incomparable vectors in Ω^r is finite.

Proof:

(a) Every infinite sequence of integers or augmented integers contains an infinite nondecreasing (scattered) subsequence, because if there does not exist a strictly increasing subsequence, there must exist some number (or ω) which is repeated infinitely often, and whose repetition also forms an infinite nondecreasing subsequence.

If we now have an infinite subset of Ω^r, we may arrange it into an infinite non-repeating sequence (Ω^r is denumerable). From this sequence we can now extract an infinite subsequence non-

*) Such sets are also known as 'order ideals'.

decreasing in the first coordinate, from which we extract an infinite subsequence also nondecreasing in the second coordinate, and so on for all r coordinates. In this manner, we end up with a non-repeating infinite subsequence which is nondecreasing in each coordinate, and thus forms a chain.

(b) A direct consequence of (a) is that every infinite subset of $\mathbb{N}^r$ or Ω^r contains distinct comparable elements. A set of incomparable vectors must thus be finite.

QED

Corollary 2.1:

A set of maximal elements $\hat{A}$, as defined in Definition 2.28, is always finite.

Proof:

Maximal elements are incomparable.

QED

The proof of Theorem 2.5 requires a few Lemmas.

Lemma 2.1:

If $A \subseteq \Omega^r$ is a chain-complete set, then:

$$V \in A \Rightarrow (\exists V' \in \hat{A}:\ V \leq V')$$

Proof:

Given $V \in A$, let $B = \{V' \in \Omega^r \mid V' \geq V\} \cap A$. Let $C \subseteq B$ be a chain in B. Since $B \subseteq A$, C is a chain in A and, by chain-completeness of A, we have lub (C) $\in A$. On the other hand, we have $\forall V'' \in C$: lub (C) $\geq V'' \geq V$. Hence, lub (C) $\in B$ and B is chain-complete.

We also have $\hat{\hat{B}} \subseteq \hat{\hat{A}}$. Indeed, suppose that $V' \in \hat{\hat{B}}$ but $V' \notin \hat{\hat{A}}$, i.e., $\exists V'' \in A$: $V'' > V'$. Since $V'' > V' \geq V$, it follows that $V'' \in B$, implying $V' \notin \hat{\hat{B}}$.

Now, Zorn's Lemma assures that every chain-complete set contains a maximal element, which implies $\hat{\hat{B}} \neq \emptyset$: Thus:

$$V \in A \Rightarrow \exists V' \in \hat{\hat{B}} \subseteq \hat{\hat{A}}: V' \geq V$$

QED

Note:

This Lemma is actually a variant of Zorn's Lemma and is not restricted to $A \subseteq \Omega^r$. If $A \subseteq \Omega^r$ it can also be proved directly by at most r induction arguments constructing infinite chains in A which eventually lead to a maximal element.

For the following Lemma we need two functions f, g: $\mathbb{N} \times \Omega^r \to \Omega^r$. Given an integer b and a vector V, f(b, V) is the result of replacing in V those coordinates which are not less than b by ω, and g(b, V) is the result of replacing these same coordinates by b:

$$1 \leq i \leq r: \begin{cases} f(b, V)(i) = \underline{\text{if}}\ V(i) \geq b\ \underline{\text{then}}\ \omega\ \underline{\text{else}}\ V(i) \\ g(b, V)(i) = \underline{\text{if}}\ V(i) \geq b\ \underline{\text{then}}\ b\ \underline{\text{else}}\ V(i) \end{cases}$$

In other words, a vector V whose finite coordinates are less than b is characterized by f(b, V) = V, and if $B \subseteq \Omega^r$ is a set (necessarily finite) whose elements have no finite coordinates which reach or exceed b, we have:

b is a bound on the finite coordinates of vectors in B $\Leftrightarrow$ $\forall V \in B$: f(b, V) = V

We interpret "bound" in the exclusive sense: The bound strictly exceeds that which is bounded.

We shall show that in the case of the set of maximal elements $\hat{A}$ of a monotone set A, we can effectively find such a bound b by testing for membership in A.

The following numbered, easily verified properties of the functions f and g will be used ($\langle b \rangle^r$ is the vector in $\mathbb{N}^r$ all of whose coordinates are equal to b).

(1) $f(b, V) \geq V$

(2) $g(b, V) \leq V$

(3) $g(b, V) \leq \langle b \rangle^r$

(4) $f(g(b, V)) = f(b, V)$

(5) $V \leq V' \Rightarrow f(b, V) \leq f(b, V')$

Now we shall prove:

Lemma 2.2:

If $A \subseteq \Omega^r$ is monotone and chain-complete, and $\hat{A}$ is the set of maximal elements of A, then b is a bound (in the strict, exclusive sense) on the finite coordinates of maximal elements iff:

$$(*) \quad \forall V \leq \langle b \rangle^r: \; V \in A \Rightarrow f(b, V) \in A$$

Proof:

if part: Suppose $V \in \hat{A}$ and some finite coordinates reach or exceed b, i.e. $f(b, V) \neq V$. By (1), it follows that $f(b, V) > V$, and since V is maximal, this implies $f(b, V) \notin A$.

By (4), we also have $f(g(b, V)) \notin A$, and by (3) we have $g(b, V) \leq \langle b \rangle^r$. But then the contrapositive of hypothesis (*) implies that $g(b, V) \notin A$.

This, together with (2), contradicts the monotonicity we have assumed for A. (Note that chain-completeness is not required for this part.)

only if part: Since $\hat{A}$ is finite (Corollary 2.1), there exists a bound b such that:

$$V \in \hat{A} \Rightarrow f(b, V) = V$$

From Lemma 2.1 (which is where chain-completeness is needed) it follows that:

$$V \notin A \Rightarrow \exists V' \in \hat{A}: V' \geq V$$

By (5) this implies $f(b, V') \geq f(b, V)$, and, since $V' \in \hat{A}: V' \geq f(b, V)$. Then $f(b, V) \in A$ follows from the monotonicity of A.

QED

Now we are ready to prove:

Theorem 2.5:

If $A \subseteq \Omega^r$ is monotone and chain-complete, then its finite set of maximal elements $\hat{A}$ is uniformly reducible to A, and it characterizes A as follows:

$$A = \{V \in \Omega^r \mid \exists V' \in \hat{A}: V' \geq V\}$$

Proof:

Since A is monotone and chain-complete, we have $\hat{A} \subseteq A$ and:

$$\forall V \in \Omega^r: (\exists V' \in \hat{A}: V \leq V') \Rightarrow V \in A$$

Lemma 2.1 shows that the converse also holds:

$$\forall V \in \Omega^r: V \in A \Rightarrow (\exists V' \in \hat{A}: V \leq V').$$

This proves the characterization of A by $\hat{A}$.

To establish the uniform reducibility of $\hat{A}$ to A, we must show how

to generate exhaustively all vectors in $\hat{A}$. Since $\hat{A}$ is finite (Corollary 2.1), there exists a bound b on the finite coordinates of its elements. This bound can be found by testing larger and larger integers b for the property (*) of Lemma 2.2, which involves a finite and bounded number of membership tests in A at each step.

Once a bound b has been found, only a bounded number of vectors are candidates for membership in $\hat{A}$. For each candidate V, the following procedure tests whether $V \in \hat{A}$: Let U_i be the vector $U_i(j) = \underline{\text{if}}\ i = j\ \underline{\text{then}}\ 1\ \underline{\text{else}}\ 0$. Then

$$[V \in A \ \&\ (\forall i,\ 1 \leq i \leq r:\ V + U_i \neq V \ \Rightarrow\ V + U_i \notin A)] \ \Leftrightarrow\ [V \in \hat{A}].$$

This follows from the definition of $\hat{A}$ and the monotonicity of A.

QED

For the sake of completeness, it should be noted that the converse of Theorem 2.5 also holds, i.e. that:

$A = \{V \in \Omega^r \mid \exists V' \in \hat{A}:\ V' \geq V\} \ \Rightarrow$ A monotone and chain-complete.

This is thus a useful characterization of monotone and chain-complete sets of vectors over the augmented integers Ω.

We shall now study the chain-completions of monotone sets.

Lemma 2.3:

Let C be a chain in a monotone set $A \subseteq \Omega^r$, and let $V = \underline{\text{lub}}\ (C)$. Then we have:

$$\forall V' \in \mathbb{N}^r:\ V' \leq V \ \Rightarrow\ V' \in A$$

Proof:

Whether C is infinite or not, it must contain a vector V'' such that $V' \leq V'' \leq V$, because each coordinate in the chain must eventually

reach or exceed the corresponding coordinate of V', which is finite. But this vector V'' covers V' and, being in C, is an element of A. Since A is monotone, V' is also in A.

QED

Theorem 2.6:

The chain-completion of a monotone set $A \subseteq \Omega^r$ is monotone and consists exactly of the least upper bounds of all chains in A. (If $A \subseteq \mathbb{N}^r$, then $A^c - A$ consists exactly of the least upper bounds of all infinite chains in A.)

Proof:

(a) composition of A^c:

A^c certainly contains all the lub's of chains in A. These lub's include the elements of A, which are the lub's of one-element chains. It remains to be shown that nothing else is in A^c, i.e. that the set $A' = A \cup \{\text{all lub's of chains in } A\}$ is already chain-complete.

If A' is finite, there is no contest, so let us assume that $C = \{V_1, V_2, \ldots\} \subseteq A'$ is an infinite chain in A': $\forall j$: $V_{j+1} > V_j$. Some of these V_j's may have ω-coordinates.

Let us scan along the sequence $V_1, V_2, \ldots$ and replace each V_j by V_j' as follows:

$V_1' = \langle 0 \rangle^r$

$\forall j > 1;\ 1 \le i \le r$: $V_j'(i) = \underline{\text{if}}\ V_j(i) = \omega\ \underline{\text{then}}\ V_{j-1}'(i) + 1\ \underline{\text{else}}\ V_j(i)$

These vectors V_j' form a chain $C' \subseteq \mathbb{N}^r$, and it is clear that C and C' have the same lub: $V = \underline{\text{lub}}\,(C) = \underline{\text{lub}}\,(C')$. Each vector $V_j \in C$ is thus covered by $V = \underline{\text{lub}}\,(C)$. But then Lemma 2.3 implies $V_j' \in A'$, so that C' is also a chain in $A' \cap \mathbb{N}^r$. Now we observe that

$A' \cap \mathbb{N}^r = A \cap \mathbb{N}^r$, because if the lub of a chain in A has no ω-coordinates, it is the lub of a finite chain, and hence an element of A. Thus C' is a chain in A, and its lub is in A': We have proved $V \in A'$, and thus the chain-completeness of A'.

(b) monotonicity of A^c:

Let $V \in A^c$, and let $V' \leq V$. From (a) it follows that there exists a chain $C \subseteq A$ whose lub is V. If we scan the vectors in C in increasing sequence, each coordinate must eventually reach or exceed any finite coordinate of V'. Let $V'' \in C$ be a vector which covers V' in every finite coordinate of V', and let C'' be the chain of all vectors following V'' in C, so that lub (C'') = lub (C) = V. Each vector in C'' covers V' in the finite coordinates of V'. Now let C' be the chain obtained by replacing in each vector of C'' those coordinates which exceed V' by the corresponding coordinates of V'. The monotonicity of A (recall that $C'' \subseteq C \subseteq A$) implies that $C' \subseteq A$, and clearly $V' = \text{lub}\,(C')$. Hence $V' \in A^c$, and A^c is monotone.

QED

Corollary 2.2:

If $A \subseteq \mathbb{N}^r$ is monotone, then $A = A^c \cap \mathbb{N}^r$.

Proof:

This follows from Theorem 2.6 and the fact that any lub which is not in A is the lub of an infinite chain, and thus contains ω-coordinates.

QED

Let us now recall the definition of agreement between two vectors $V, V' \in \Omega^r$ (Definition 2.15), expressed in vector notation:

V agrees with V', written $V \approx V'$, iff the coordinates which are finite in both V and V' are equal in V and V':

$$V \approx V' \Leftrightarrow (\forall i,\ 1 \leq i \leq r:\ V(i) + V'(i) \neq \omega \Rightarrow V(i) = V'(i))$$

From this definition it follows that:

$$V \in \mathbb{N}^r,\ V' \notin \mathbb{N}^r:\quad V \approx V' \Rightarrow V < V'$$

$$V \in \mathbb{N}^r,\ V' \in \mathbb{N}^r:\quad V \approx V' \Rightarrow V = V'$$

Then a characterization of chain-completion is given by:

Theorem 2.7:

The chain-completion A^c of a monotone set $A \subseteq \mathbb{N}^r$ is such that

$$A^c = \{V \in \Omega^r \mid \forall V' \in \mathbb{N}^r:\ V' \approx V \Rightarrow V' \in A\}$$

Proof:

(a) Let $V \in A^c$, $V' \in \mathbb{N}^r$ and $V' \approx V$. Then $V' \leq V$, and since A^c is monotone (Theorem 2.6), $V' \in A^c$. Hence $V' \in A^c \cap \mathbb{N}^r$, which implies $V' \in A$ (Corollary 2.2).

(b) Let V be such that $\forall V' \in \mathbb{N}^r:\ V' \approx V \Rightarrow V' \in A$. Define a sequence of vectors $V_1, V_2 \ldots$ such that $V_j(i) =$ if $V(i) = \omega$ then j else $V(i)$, $1 \leq i \leq r$. Clearly, $\{V_1, \ldots, V_j, \ldots\}$ is an infinite chain whose lub is V, and such that $\forall j,\ V_j \approx V$, so that it is a chain in A. This implies $V \in A^c$ by the definition of chain-completion.

QED

Finally, we have

Theorem 2.8:

If $A \subseteq \mathbb{N}^r$ is monotone, then there exists a finite set $\{V_1, \ldots, V_k\} = \widehat{A^c}$, uniformly reducible to A^c, such that:

$$A = \{V \in \mathbb{N}^r \mid V \leq V_1 \ \underline{\text{or}} \ V \leq V_2 \ \underline{\text{or}} \ \cdots \ \underline{\text{or}} \ V \leq V_k\}$$

Proof:

This is a direct consequence of Theorem 2.5, Theorem 2.6 and Corollary 2.2.

QED

For results and proofs about semilinear sets, the reader is referred to Ginsburg and Spanier [14].

REFERENCES

1. Abraham, S., On Matrix Grammars, TR3, Computer Science, Technion, Haifa, Israel, 1970.

2. Agerwala, T., A complete model for representing the coordination of asynchronous processes, Hopkins Computer Research Report 32, Johns Hopkins University, Baltimore, Maryland, July 1974.

3. Baker, H. G., Petri nets and languages, Computation Structures Group Memo 68, Project MAC, M. I. T., Cambridge, Massachusetts, May 1972.

4. Baker, H. G., Rabin's Proof of the Undecidability of the Reachability Set Inclusion Problem of Vector Addition Systems, Computation Structures Group Memo 79, Project MAC, M. I. T., July 1973.

5. Bruno, J. and Altman, S. M., A Theory of Asynchronous Control Networks, IEEE Trans. Comp. C-20, No. 6, June 1971, pp 629-638.

6. Cardoza, E. W., Computational Complexity of the Word Problem for Commutative Semigroups, S.M. Thesis, Department of Electrical Engineering and Computer Science, M. I. T. (1975).

7. Commoner, F., et al., "Marked Directed Graphs", JCSS, Vol. 5, No. 5, pp 511-523 (October 1971).

8. Commoner, F., Deadlocks in Petri Nets, Report CA-7206-2311, Applied Data Research, Wakefield, Mass., June 1972.

9. Crespi-Reghizzi, S. and Mandrioli, D., Petri Nets and Commutative Grammars, Rapporto interno n. 74-5, Laboratorio di Calcolatori, Istituto di Elettrotecnica ed Elettronica del Politecnico di Milano, March 1974.

10. Davis, M., Putnam, H., and Robinson, J., "The decision problem for exponential diophantine equations", Annals of Mathematics, Vol. 74, pp 425-436 (1961).

11. Dennis, J. B., "Modular, Asynchronous Control Structures for a High Performance Processor", Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York, 1970, pp 55-80.

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

13. Estrin, G. and Turn, R., "Automatic assignment of computations in a variable structure computer system", IEEE Transactions on Computers, EC12, 6, pp 755-773 (December 1963).

14. Ginsburg, S and Spanier, E. H., "Semigroups, Presburger Formulas, and Languages", Pacific Journal of Mathematics Vol. 16, No. 2, pp 285-296 (1966).

15. Grandoni, F. and Zerbetto, P., "Description and Asynchronous Implementation of Control Structures for Concurrent Systems", International Computing Symposium 1973, A. Gunther et al. (Eds), North-Holland Publishing Co., 1974.

16. Habermann, N., On a solution and a generalization of the Cigarette Smoker's Problem, Department of Computer Science, Carnegie-Mellon University (August 1972).

17. Hack, M., Analysis of Production Schemata by Petri Nets, Technical Report TR-94, Project MAC, M.I.T., February 1972. Corrections to "Analysis of Production Schemata by Petri Nets", Computation Structures Note No. 17, Project MAC, M.I.T., June 1974.

18. Hack, M., The Equivalence of Generalized (Multiple-Arc) Petri Nets and Ordinary (Single-Arc) Petri Nets, Computation Structures Note No. 9, Project MAC, M.I.T., April 1973.

19. Hack, M., The Gödelization of Petri Nets and Vector Addition Systems, Computation Structures Note No. 10, Project MAC, M.I.T., May 1973.

20. Hack, M., Decision problems for Petri Nets and Vector Addition Systems, MAC-TM 59, Project MAC, M.I.T., March 1975. Previously published as Computation Structures Group Memo 95, Project MAC, March 1974.

21. Hack, M., The recursive equivalence of the liveness problem and the reachability problem for Petri Nets and Vector Addition Systems, Computation Structures Group Memo 107, Project MAC, M.I.T., August 1974. Also in Proceedings of the 15th Annual Symposium on Switching and Automata Theory, New Orleans, La., October 1974.

22. Hack, M., Petri Nets and Commutative Semigroups, Computation Structures Note No. 18, Project MAC, M.I.T., July 1974.

23. Hack, M., The equality problem for Vector Addition Systems is undecidable, Computation Structures Memo 121, Project MAC, M.I.T., April 1975. Also to be published in the journal of Theoretical Computer Science.

24. Hack, M., Petri Net Languages, Computation Structures Group Memo 124, Project MAC, M.I.T. (June 1975). Re-issued as TR 159, M.I.T. Laboratory for Computer Science, March 1976.

25. Hack, M. and Peterson, J. L., "Petri Nets and Languages", Conference on Petri Nets and Related Methods, M.I.T., August 1-3, 1975.

26. Hilbert, D., "Mathematische Probleme. Vortag, gehalten auf dem internationalen Mathematiker-Kongress zu Paris 1900", Nachr. K. Ges. Wiss. Göttingen, Math.-Phys. Kl. 1900, pp 253-297. Translation: Bull. Amer. Math. Soc. 8 (1901-1902), pp 437-479.

27. Holt, A. W. et al., Final Report of the Information Systems Theory Project, Technical Report RADC-TR-68-305, Rome Air Development Center, Griffiss Air Force Base, New York, 1968.

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

29. Holt, R. C., On Deadlock in Computer Systems (January 1971), Technical Report CSRG-6, Computer Science Research Group, University of Toronto (July 1972).

30. Jones, N. D. and Lien, Y. E., "Complexity of some problems in Petri Nets", Conference on Petri Nets and Related Methods, M.I.T., August 1-3, 1975.

31. Jump, J. R. and Thiagarajan, P. S., "On the Equivalence of Asynchronous Control Structures", 13th Annual Switching and Automata Theory Symposium, October 1972, pp 212-223.

32. Jump, J. R. and Thiagarajan, P. S., On the Interconnection of Asynchronous Control Structures, Laboratory of Computer Science and Engineering, Rice University, September 1972.

33. Karp, R. M. and Miller, R. E., "Parallel Program Schemata: A Mathematical Model for Parallel Computation", IEEE Conference Record, 8th Annual Switching and Automata Theory Symposium, October 1967, pp 55-61.

34. Keller, R. M., Vector Replacement Systems: A Formalism for Modelling Asynchronous Systems, TR 117, Computer Science Laboratory, Princeton University, December 1972.

35. Keller, R., "A Fundamental Theorem of Asynchronous Parallel Computation", Parallel Processing (T. Feng, Editor), Proceedings of the Sagamore Computer Conference, August 20-23, 1974. Springer, Lecture Notes in Computer Science 24, 1975.

36. König, D., Theorie der endlichen und unendlichen Graphen, Akademische Verlagsgesellschaft, Leipzig, 1936.

37. Kosaraju, S. R., Limitations of Dijkstra's Semaphore Primitives and Petri Nets, Hopkins Computer Research Report 25, Johns Hopkins University, May 1973.

38. Lipton, R. J., "Limitations of Synchronization Primitives with Conditional Branching and Global Variables", 6th Annual ACM Symposium on the Theory of Computing, May 1974, pp 230-241.

39. Lipton, R., "The Reachability Problem is Exponential-Space-Hard", Conference on Petri Nets and Related Methods, M.I.T., August 1-3, 1975. Also Yale C.S. Report #62, Jan. 1976.

40. Matijasevič, Ju. V., "Enumerable sets are diophantine", Soviet Math. Dokl. 11, 2 (1970), pp 354-357.

41. Miller, R. E., "A Comparison of Some Theoretical Models of Parallel Computation", IEEE Trans. Comp. C-22, No. 8, August 1973.

42. Miller, R. E., Some relationships between various models of parallelism and synchronization, IBM Research Report RC5074, IBM T. J. Watson Research Center, Yorktown Heights, N.Y., October 1974.

43. Minsky, M., Computation: Finite and Infinite Machines, Prentice-Hall, Inc., Englewood Cliffs, N.Y., 1967, pp 255-258.

44. Nash, B. O., "Reachability Problems in Vector Addition Systems", Amer. Math. Monthly 80, (1973), pp 292-295.

45. Noe, J. D. and Nutt, G. J., "Macro-E-Nets for Representation of Parallel Systems", IEEE Trans. Comp. C-22, No. 8, August 1973.

46. Parikh, R. J., Language Generating Devices, M.I.T. Research Laboratory of Electronics, Quarterly Progress Report 60, 1961, pp 191-212.

47. Parnas, D. L., "On a Solution to the Cigarette Smoker's Problem (without conditional statements)", CACM, Vol. 18, No. 3, pp 181-183 (March 1975).

48. Patil, S. S., Macromodular Design of Asynchronous Circuits, Computation Structures Group Memo 41, Project MAC, M.I.T., May 1969.

49. Patil, S. S., Coordination of Asynchronous Events, Report MAC-TR-72, Project MAC, M.I.T., Cambridge, Massachusetts, June 1970.

50. Patil, S. S., "Closure Properties of Interconnections of Determinate Systems", Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, June 1970, pp 107-116.

51. Patil, S. S., Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination among Processes, Computation Structures Group Memo 57, Project MAC, M.I.T., February 1971.

52(a) Peterson, J. L., Modelling of Parallel Systems, Ph.D. Thesis, Department of Electrical Engineering, Stanford University, Stanford, California, December 1973.

52(b) A condensed version of Reference 52(a), "Computation Sequence Sets", is to be published in the Journal of Computer and Systems Sciences.

53. Peterson, J. L. and Bredt, T. H., "A Comparison of Models of Parallel Computation", Information Processing 1974, North Holland Publishing Company, 1974.

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

55. Petri, C. A., "General Net Theory", Conference on Petri Nets and Related Methods, M.I.T., August 1-3, 1975.

56. Rabin, M., private communication, Fall 1972.

57. Rogers, H., Theory of Recursive Functions and Effective Computability, McGraw-Hill, 1967.

58. Schmid, H. A., "An Approach to the Communication and Synchronization of Processes", International Computing Symposium 1973, A. Gunther et al. (Eds.), North-Holland Publishing Co., 1974.

59. Shapiro, R. and Saint, H., Representation of Algorithms, Report RADC-TR-69-313, Vol. II, Griffiss Air Force Base, New York, September 1969.

60. Slutz, D. R., The Flow Graph Schemata Model of Parallel Computation, Technical Report TR-53, Project MAC, M.I.T., September 1968.

61. Taiclin, M. A., "On Elementary Theories of Commutative Semigroups", Algebra i Logika, Vol. 5 (1966) pp 50-69 (in Russian).

62. Van Leeuwen, J., Rule-labeled Programs, Ph. D. Thesis, Mathematics Department, University of Utrecht, Netherlands, 1972.

63. Van Leeuwen, J., "A Partial Solution to the Reachability Problem for Vector Addition Systems", 6th Annual ACM Symposium on Theory of Computing, May 1974, pp 303-309.

BIOGRAPHICAL NOTE

Michel Hack was born in Luxembourg on May 8, 1947. He lived in Belgium, Germany and Luxembourg before starting High School in Fontainebleau, France.

He obtained his Baccalaureate in Mathematics from the Academy of Paris in July 1964, while studying at the Lycée Louis-le-Grand in Paris. From 1966 to 1969 he studied at the Ecole Nationale Supérieure des Télécommunications in Paris, graduating with an Engineer's degree in electronics. During this period, he spent summers doing electronic design for A.E.G. in Germany, Portescap in Switzerland, and C.G.C.T. in France.

He entered M.I.T. in September 1969, where he joined Jack Dennis' Computation Structures Group. He obtained his M.S. in Electrical Engineering in February 1972 for a study of the properties of Free Choice Petri Nets.

While at M.I.T. he has been a Research Assistant and a Teaching Assistant.

He has also worked summers and part-time with Dr. Anatol W. Holt at Massachusetts Computer Associates (formerly the Research Division of Applied Data Research), from 1970 to 1974.

He has now joined the IBM T. J. Watson Research Center in Yorktown Heights, New York.

Michel Hack is a member of the Association des Lauréats du Concours Général, the Society of the Sigma Xi, and the Association for Computing Machinery.

CS-TR Scanning Project
Document Control Form

Date : 12/11/95

Report # LCS-TR-161

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

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

Document Type:

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

Document Information

Number of pages: 194 (199-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

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

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

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

Check each if included with document:

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

Page Data:

Blank Pages (by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:
IMAGE MAP: (1-194) UNMARKED TITLEPAGE, 2-194
(195-199) SCANCONTROL, PRINTERS NOTES, TRGT'S(3)

Scanning Agent Signoff:
Date Received: 12/11/95 Date Scanned: 1/12/96 Date Returned: 1/18/96

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

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

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