EFFICIENT METHODS FOR VERIFICATION AND PERFORMANCE EVALUATION OF CONCURRENT SYSTEMS BY PETRI NETS

A dissertation submitted to the faculty of the University of Manitoba

Doctor of Philosophy

Andrei Kovalyov

Electrical and Computer Engineering Department

UNIVERSITY OF MANITOBA

September 2001

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

EFFICIENT METHODS FOR VERIFICATION AND PERFORMANCE EVALUATION OF CONCURRENT SYSTEMS BY PETRI NETS

BY

ANDREI KOVALYOV

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

of

DOCTOR OF PHILOSOPHY

Acknowledgements

First of all, I would like to thank my advisor Prof. Dr. Robert D. McLeod for accepting me as his student, for reading and correcting my articles and papers and for providing financial support. He always was very kind and helpful in guidance through the research and life.

I thank Prof. Dr. Steve Onyshko and Prof. Dr. Sylvanus A. Ehikioya from the University of Manitoba and Prof. Dr. Gurdeep Singh Hura from the University of Idaho at Idaho Falls, for their agreeing to participate in my thesis committee. I also appreciate their discussions and support.

I would also like to thank Prof. Dr. Joerg Desel and two of my former advisors Prof. Dr. Ruediger Valk and Prof. Dr. Javier Esparza in Germany, for their reading and correcting my publications and funding the research, and also for their guidance, help and support.

I also appreciate Prof. Dr. Alex Yakovlev in the U.K. for providing me with material for the research.

I would also like to thank my former advisors Prof. Dr. Arcadiy D. Zakrevsky and Prof. Dr. Yuriy V. Pottosin in Belarus for their help and introducing me to the research.

Most of all, I would like to thank my parents for their love and support.

Abstract

Petri nets are an efficient and powerful formal tool for modeling, analysis and synthesis of asynchronous, distributed systems with concurrency. Petri nets combine convenient modular graphical representation with mathematical formalism and possess such qualities as generality, simplicity and formality. They have a developed theory of analysis and synthesis.

Here we consider verification of concurrent system correctness. Correctness qualities can be classified into two groups: semantical and syntactical. Semantical qualities rely on specifics of the particular application field. Syntactical qualities are more generic. They can be defined without knowing the specific purpose of the system. Here correctness is considered as a syntactical quality. The correctness of the system can be expressed in terms of the behavioral properties of a Petri net modeling the system. Two major behavioral properties are commonly used for the definition of correctness of concurrent systems. They are *liveness* (the absence of partial and global deadlocks) and *boundedness* (the absence of overflows in finite stores). The developed universal methods for analyzing these properties face the state explosion problem. Actually, the main weakness of Petri nets is the complexity problem. The objectives of the thesis are the creation of efficient methods and algorithms for verification of concurrent systems.

We develop a set of simple local reduction rules for Extended Free Choice Petri nets. We give a complete reduction method for regular Petri nets and estimate its complexity as $O(|S| \times |T|)$.

We develop an $O(|S| \times |T|)$-algorithm to decide if a given Extended Free Choice Petri net is live and bounded, which is a reduction by one order of magnitude, compared to the previous algorithm ($O(|S|^2 \times |T|)$). We present an $O(|S| \times |T|)$-algorithm to

decide if a given Petri net is regular.

We prove several Rank Theorems for Extended Free Choice and general class of Petri nets and show that they can be used as an alternative technique for the verification of workflow procedures.

We propose a *cubic* algorithm for the computation of the concurrency relation of Live and Bounded Extended Free-Choice Petri nets. Prior to this work the problem was shown to be polynomial only for live Marked Graphs and 1-bounded Conflict-Free Petri nets. We generalize the previous algorithm for Free-Choice Petri nets, to regular Petri nets. The time complexity of the algorithm is $O(n^4)$, where n is the number of nodes in the net.

In [46] we propose polynomial algorithms for performance evaluation of communication networks using stochastic Petri nets. We used theoretical results and polynomial algorithms from [13, 19, 20, 57] for quantitative analysis of stochastic Petri nets.

Contents

List of Figures

List of Abbreviations

BP bounded and persistent

EFC extended free choice

FC free choice

FRT-nets nets with freely related T-semiflows

LB live and bounded

LCA Logical Control Algorithm

LPP linear programming problem

LSP live safe and persistent

LSPR live safe persistent and reversible

SCMG strongly connected marked graph

SCSM strongly connected state machine

SMD state machine decomposable

STG Signal Transition Graph

VME Versa Multibus (or Module) Euro

List of Mathematical Notations

$\cup$ union of sets

$\cap$ intersection of sets

$\setminus$ difference of sets

$\times$ Decart Product of sets

$\subseteq$ subset

$\subset$ proper subset

$\in$ belongs

$\notin$ does not belong

$|X|$ the number of elements of the set X

$\chi(X)$ the characteristic vector of the set X

$\exists$ there exists

$\forall$ for every

$|$ such that

$\cdot$ matrix multiplication

Chapter 1

Introduction

1.1 Research Subject

The Research subject is Petri nets [89, 83, 84, 22] and their derivatives: Signal Transition Graphs [75, 80, 15], Logic Control Algorithms [101] and stochastic Petri nets [2]. Petri nets are among the most popular, efficient and powerful tools for modeling, analysis and synthesis of concurrent systems. Petri nets were introduced in Carl Adam Petri's dissertation [85] in 1962 in Germany.

Applications of Petri nets are numerous and include: communication networks and protocols, computer architecture, semantics of programming languages, artificial intelligence, software engineering, complexity theory, distributed software systems, distributed database systems, concurrent and parallel programs, flexible manufacturing/industrial control systems, discrete event systems, multiprocessor memory systems, dataflow computing systems, fault tolerant systems, programmable logic and VLSI arrays, asynchronous circuits and structures, compiler and operating systems, office information systems, formal languages, logic programs, local area networks, legal systems, human factors, neural networks, digital filters, and decision models.

Petri nets combine convenient graphical representation with mathematical formalism and possess such qualities as generality, simplicity and formality. As a graphical tool,

Petri nets can be used as visual communication aids similar to flow charts, block diagrams, and networks. In addition, tokens are used to simulate the dynamic and concurrent activities of systems. The Petri net is a model of a concurrent system, in the same way that the finite automaton is a mathematical model of a sequential system. Simplicity is provided by some elementary concepts and assumptions in Petri net theory. Basic notions are causality, non-determinism and concurrency. Simplicity can be considered as a necessary condition for the usefulness of a model. The generality of Petri nets can be shown in 3 different ways. First, Petri nets can serve as a formalism in which many specific models can be translated. So Petri nets can serve as a tool for the comparison of properties of many models. Second, Petri nets allow us to easily define different behavioral semantics: Interleaving, Step and Partial Order. Third, many interesting properties can be expressed in a satisfactory way from Petri net theory.

Other advantages of Petri nets:

- an explicit specification of concurrency, synchronization and communication;
- compact representation of concurrent system with large number of global states;
- modular graphical representation;
- availability of strong theoretical results and efficient techniques for their analysis and synthesis.

1.2 Motivation, Problems and Research Objectives

Methods based on Petri nets belong to formal methods. Why do we need formal methods?

Modern engineering systems are so complex that human capabilities are not sufficient to design them by hand. Therefore, system design automation is very important.

Automation as a rule, is based on formalization: the development of formal models and languages of description of design objects, and also formal methods of design. With increasingly complex systems, the need for new models and languages has also grown. These new models must allow us to describe concurrent, asynchronous and discrete processes. In addition, the models should be formal, generic and simple. Petri nets and their derivatives satisfy these properties.

After model development, verification and/or performance evaluation is usually required.

Here we consider verification of concurrent system correctness. Correctness is one of system qualities which may be defined formally and without knowing the specific purpose of the system. The correctness of the system can be expressed in terms of the behavioral properties of a Petri net modeling the system. Two major behavioral properties are commonly used for the definition of correctness of concurrent systems. They are *liveness* (the absence of partial and global deadlocks) and *boundedness* (the absence of overflows in finite stores).

Universal methods developed for these new models face the state explosion problem. That is why the development of new efficient methods is very important for the models to be adopted. The research objectives here are the creation of efficient methods and algorithms for verification and performance evaluation of concurrent systems using Petri nets.

1.3 Publications, Personal Contribution and Research Experience

The research work related to the study here has resulted in 30 items included in the references.

All main results in the thesis were obtained by the author himself.

The author received his Master of Electronics Degree in 1982. He has been working in Petri net field since 1985. In 1985-1994 he worked as a researcher at the Academy of Science in Belarus. In 1994-1996 he did research at Hamburg and Munich Universities in Germany. Since April 1996 he has worked at the University of Manitoba, Canada.

1.4 Structure of the Thesis

In Chapter 2 we give the definition of Petri nets. Since the verification of systems is a main topic of our research, we pay special attention to qualitative properties of Petri nets such as liveness and boundedness. We also define some subclasses of Petri nets and give a short description of existing analysis and synthesis methods.

In Chapter 3 we describe our results in linear-algebraic methods of Petri net analysis.

In Chapter 4 we analyse Petri nets using reduction.

In Chapter 5 we consider the quadratic algorithm for the liveness and boundedness analysis of regular Petri nets.

Chapter 6 is devoted to the computation of the concurrency relation for regular Petri nets.

In Chapter 7 we give the method of unfoldings.

In Chapter 8 we consider an application of Petri nets to hardware design, namely to the analysis and design of asynchronous circuits.

In Chapter 9 we apply Petri nets to the verification of Logical Control Algorithms (LCAs). We define LCAs and their correctness, and consider the verification methods.

In Chapter 10 we consider the application of Petri nets to workflow management.

In Chapter 11 we investigate quantitative properties of Petri nets, namely timing characteristics, and their application to the performance evaluation of communication networks.

Chapter 12 is devoted to the modular synthesis of Petri nets.

Where appropriate each chapter summarizes the research contribution made by the author.

Chapter 2

Petri Nets, their Properties and Classes

2.1 Petri Nets

A Petri net is a directed graph, together with an initial state called the initial marking, M^0. The graph is a directed bipartite consisting of two kinds of nodes, called places and transitions.

Places usually model conditions, stores, channels or queues, and transitions model events, tasks, jobs or servers.

In Fig. 2.1 a Petri net, modeling a non-deterministic wait process, is shown. Places are graphically represented by circles and transitions by boxes.

Formally, a *net* is a triple $N = (S, T, F)$ with $S \cap T = \emptyset$ and $F \subseteq (S \times T) \cup (T \times S)$. S is the set of *places*, T is the set of *transitions*.

A *marking* of a net $N = (S, T, F)$ is defined as a mapping $M : S \to \mathbb{N}$. A place is called *marked* by M iff $M(s) > 0$. Let $S' \subseteq S$. Then S' is *marked* iff $\exists s \in S' | s$ is marked. A *Petri net* (or a *system*) is a pair (N, M^0).

Let $X = S \cup T$ be the set of nodes of N. Then for $x \in X$ the set of input nodes of

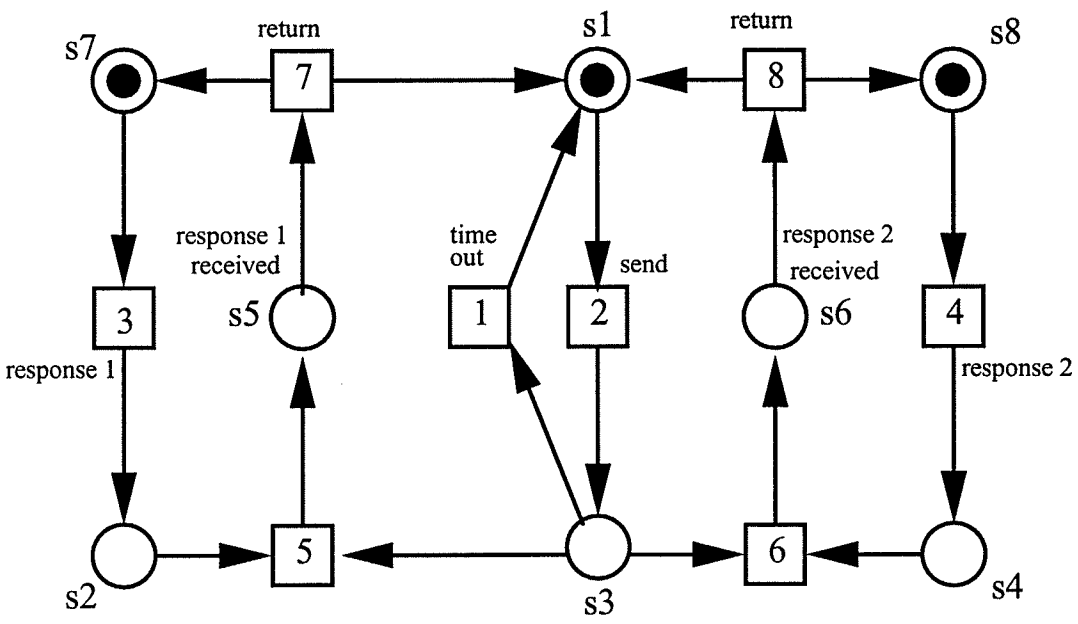

Fig. 2.1 A Petri Net.

x is denoted by ${}^\bullet x$ and is defined as ${}^\bullet x = \{y \in X | (y, x) \in F\}$, and the set of output nodes of x is denoted by $x^\bullet$ and is defined as $x^\bullet = \{y \in X | (x, y) \in F\}$.

From now on we assume that all nets we deal with are *finite* (i.e. X is finite), *connected* (i.e. $X \times X = (F \cup F^{-1})^*$, where R^* denotes the reflexive and transitive closure of a relation R), and have at least one place and one transition. N is *strongly connected* iff $X \times X = F^*$.

Dynamics in Petri nets are introduced with a firing rule. A state (or marking) is changed according to the following transition (firing) rule:

1) A transition t is said to be enabled if each input place s of t is marked with at least one token.

2) An enabled transition may fire.

3) A firing of an enabled transition t removes one token from each input place of t, and adds one token to each output place of t. So, Firing of t yields a new marking

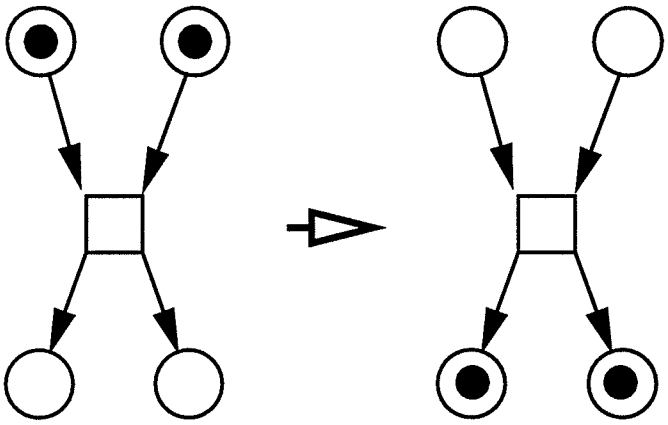

Fig. 2.2 A Firing Rule.

M' (denoted $M[t\rangle M'$), where

$$M'(s) = \begin{cases} M(s) + 1 \text{ iff } s \in t^\bullet \backslash {}^\bullet t \\ M(s) - 1 \text{ iff } s \in {}^\bullet t \backslash t^\bullet \\ M(s), \text{ otherwise} \end{cases}$$

A firing rule is illustrated in Fig. 2.2.

The expression $M_1[t\rangle M_2$, where M_1, M_2 are markings of N, denotes that M_1 enables transition t (e.g. t is fireable at M_1), and that the marking reached by the occurrence of t is M_2. The empty sequence ϵ is an occurrence sequence: we have $M[\epsilon\rangle M$ for every marking M.

For a marking M, the set of reachable markings $[M\rangle$ is defined as the least set of markings which satisfies $M \in [M\rangle$ and includes for every marking $M' \in [M\rangle$ the set of all successor markings of M'.

2.2 Properties of Petri Nets

The most important and common behavioral properties of Petri nets checked at the verification of systems are liveness and boundedness [22]. Liveness corresponds to the absence of global or partial deadlocks in a system, boundedness to the absence

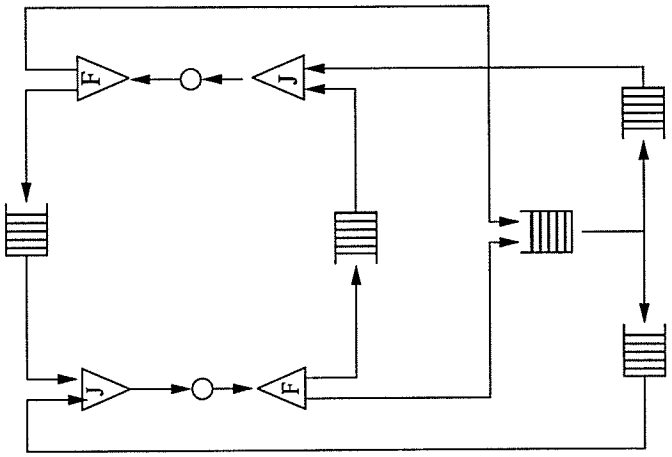

Fig. 2.3 A Queueing Network.

of overflows in stores. The complexity of the problems of liveness and boundedness for Petri nets is exponential depending on the size of the net. The liveness and boundedness are common criteria of correctness or soundness of systems.

A transition t is *live* at a marking M iff $\forall M' \in [M\rangle \; \exists M'' \in [M'\rangle \quad M''[t\rangle$. A system (N, M^0) is *live* iff every transition of T is live. A net N is *structurally live* iff there exists a live marking of N.

A place $s \in S$ is *n-bounded* ($n \in \mathbb{N}$) iff $\forall M' \in [M\rangle \quad M'(s) \leq n$. A place $s \in S$ is *safe* iff s is 1-bounded. A place $s \in S$ is *bounded* iff $\exists n \in \mathbb{N}$ such that s is n-bounded. A system (N, M^0) is *bounded* iff every place of S is bounded. A net N is *structurally bounded* iff every marking of N is bounded.

A marking M of the net N is *dead* iff there are no transitions fireable at M. A transition t is *dead* at a marking M iff $\forall M' \in [M\rangle \quad t$ is not fireable at M'.

A marking M is called *live* iff every transition is live. A marking M is *bounded* iff $\forall s \in S \; s$ is bounded.

A net N is *well-formed* iff there exists a live and bounded marking of N.

Consider the queueing network shown in Fig. 2.3.

In Fig. 2.4 we show the Petri net model of the queueing network from Fig. 2.3.

We can see that the network is designed with an error and will be in a deadlock. A

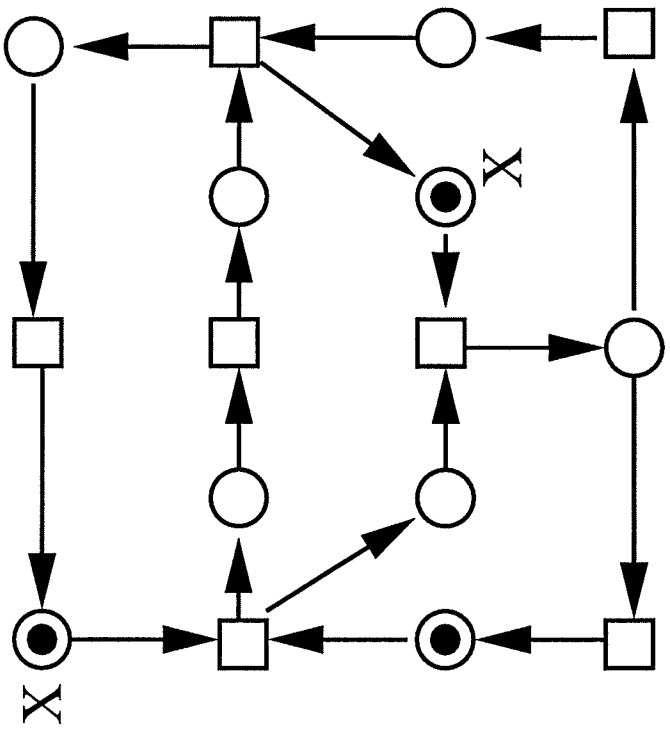

Fig. 2.4 The Petri Net Model of the Queueing Network of Fig. 2.3.

deadlock marking is shown by 'X'. Hence the system is not live.

Now we give an informal definition of liveness. A Petri net is *live* iff every transition can always occur again.

Now consider the queueing network in Fig. 2.5.

In Fig. 2.6 we show the Petri net model of the queueing network from Fig. 2.5.

We can see that the network is designed with an error and will have an overflow. The overflow place is shown by two 'X'. Hence the system is not bounded.

Now we give a definition of boundedness. A Petri net is *bounded* iff there exists a number b such that no reachable marking puts more than b tokens in any place. Places in a Petri net are often used to model buffers and registers for the storage of data. If a Petri net is Unbounded, than overflows can occur in these buffers or registers.

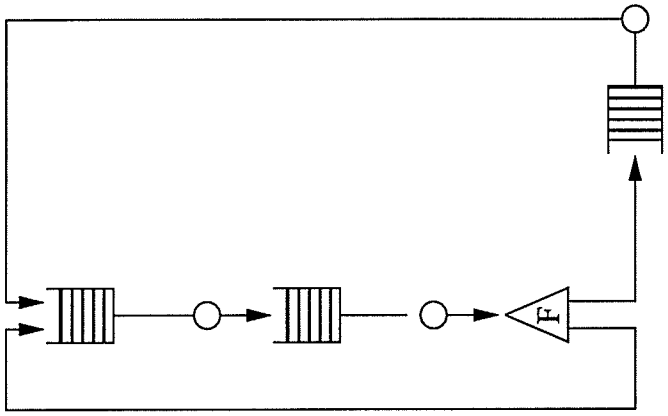

Fig. 2.5 A Queueing Network.

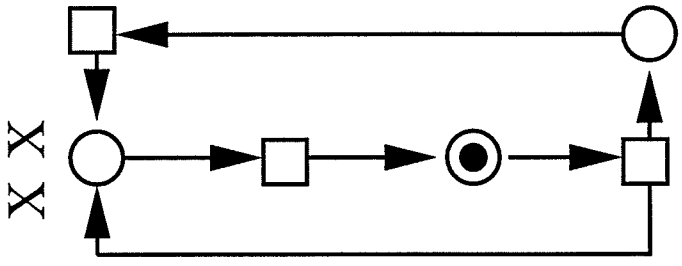

Fig. 2.6 The Petri Net Model of the Queueing Network of Fig. 2.5.

2.3 Classes of Petri Nets

The class of Petri nets defined above is called *ordinary.* All Petri nets considered here are ordinary. However, different applications require their own subclasses, modifications or extentions of Petri nets which are suitable for these applications.

The concept of time is not explicitly given in the original definition of Petri nets. However, for performance evaluation and scheduling problems of dynamic systems, it is necessary and useful to introduce time delays associated with transitions and/or places in their models. Such a Petri net model is known as a *timed* Petri net [88] if the delays are deterministically given, or as a *stochastic* Petri net [2] if the delays are probabilistically specified. *High-level* Petri nets include *predicate/transition* Petri nets [33], *colored* Petri nets [36], *object* Petri nets [96] and Petri nets with *individual tokens* [90].

By imposing some restrictions on the structure and/or behavior of Petri nets we can define different subclasses of Petri nets.

There exists a trade-off between modeling power of a Petri net class and its analyzability. The more modeling power the less analyzability. Therefore, in order to reduce the complexity of analysis, for any application we should strive to choose the most restricted class which is sufficient for that application not to lose the modeling power.

Two interesting subclasses of ordinary Petri nets are free-choice Petri nets and regular Petri nets.

Free-choice and extended free-choice Petri nets can model conflict and concurrency, they are easy to analyze, but they are too restricted for some applications.

Another class of Petri nets is regular Petri nets. They are those Petri nets that satisfy the conditions of the Rank Theorem, which refers to the linear algebraic representation of Petri nets. Live and bounded extended free-choice Petri nets are a proper subset of regular Petri nets.

For the applications considered here, the classes of extended free-choice Petri nets

and regular Petri nets are fairly sufficient. These classes are subclasses of ordinary Petri nets.

Other subclasses of Petri nets we are considering here are for auxiliary purposes only.

A net (S,T,F) is a *T-net* (or *marked graph*) iff it satisfies the following two properties:

$$\forall s \in S \quad |s^\bullet \cap T| = 1 \tag{2.1}$$

$$\forall s \in S \quad |{}^\bullet s \cap T| = 1 \tag{2.2}$$

A net (S,T,F) is a *S-net* (or *state machine*) iff it satisfies the following two properties:

$$\forall t \in T \quad |t^\bullet \cap S| = 1 \tag{2.3}$$

$$\forall t \in T \quad |{}^\bullet t \cap S| = 1 \tag{2.4}$$

A net $N = (S,T,F)$ is called an *1-out net* iff $\forall s \in S \ |s^\bullet| = 1$.

A *cluster* of a net N is a connected component of the relation $F_1 \cup F_1^{-1}$, where $F_1 = F \cap (S \times T)$. $[x]$ denotes the cluster containing $x \in S \cup T$). The set of all clusters is denoted by $\mathcal{A}$.

A net $N = (S,T,F)$ is called a *free-choice net* (*FC-net* for short) [34] iff $\forall s \in S \, \forall t \in T$ $(s,t) \in F$ implies $s^\bullet = \{t\}$ or ${}^\bullet t = \{s\}$.

A net $N = (S,T,F)$ is called an *extended free-choice net* (*EFC-net* for short) [34] iff $\forall s \in S \quad \forall t \in T \ (s,t) \in F$ implies ${}^\bullet t \times s^\bullet \subseteq F$.

Let S and T be arbitrarily but fixed ordered. Let $Q \subseteq S (Q \subseteq T)$. Then we denote $\chi(Q)$ the characteristic vector of the set Q. The *incidence matrix* $C : S \times T \to \{-1,0,1\}$ of N is defined by $C(-,t) = \chi[t^\bullet] - \chi[{}^\bullet t]$. We denote every vector $(0,0,...,0)$ by 0 and every vector $(1,1,...,1)$ by 1. A vector J is called *S-invariant(T-invariant)* iff $J \cdot C = 0$ $(C \cdot J = 0)$. For two vectors J and J', $J \geq J'$ iff for every i-th component

$J(i) \geq J'(i)$. An invariant J is *semi-positive* iff $J \geq 0$ and $J \neq 0$. An invariant J is *positive* iff $J \geq 1$. The *support* $\langle J \rangle$ of a vector J is the set of elements x satisfying $J(x) \neq 0$. For a set $Q \subseteq T$, $dim(Q)$ denotes the dimension of the space of such T-invariants J which satisfy $\forall t \notin Q \; J(t) = 0$.

A system (N, M^0) (not necessarily EFC) is called *regular* [22] iff the following four conditions hold.

$$\text{there exists a positive S-invariant} \tag{2.5}$$

$$\text{there exists a positive T-invariant} \tag{2.6}$$

$$rank(C) = |\mathcal{A}| - 1 \tag{2.7}$$

$$\text{for every semi-positive S-invariant } I \text{ of } N, I \cdot M^0 > 0 \tag{2.8}$$

Regularity is a sufficient condition for a Petri net to be live and bounded, and can be efficiently decided [57].

While being wider than the class of live and bounded free-choice Petri nets, the class of regular Petri nets is still restricted. To show the usefulness of regular Petri nets in the design flow we propose to identify the correctness property of a model as the regularity of the Petri net. Then we can use modular design of a correct model similar to the one for free-choice Petri nets [31].

Regular Petri nets are a subclass of a wider class of state machine decomposable Petri nets. A state machine decomposable Petri net consists of several state machines. In practice, the design of systems traditionally uses state machine models which are very familiar to engineers and designers. Often the system consists of several concurrently running and communicating state machines. Each subsystem (state machine) can

be represented in the model autonomously. Then such representations should be coordinated to describe the whole system. It is modular approach to system design.

2.4 Analysis Methods for Petri Nets

The problem of analysis has been intensely studied since the early seventies. The results of this research point out a very clear trade-off between expressive power and analyzability. Even though most interesting properties are decidable for arbitrary Petri nets, the decision algorithms are extremely inefficient. In this situation it is important to explore the analyzability border, i.e., to identify a class of Petri nets, as large as possible, for which strong theoretical results and efficient analysis algorithms exist.

Analysis methods are classified into 5 groups:

1) Reachability graph [84]. In this method we enumerate the global states of the system. From the initial marking we can get new markings, applying enabled transitions. From new markings we get more new marking and so on. At the end we get the Reachability graph. Nodes are marking (global states) and arcs are transitions. So every Petri net can be mapped to the corresponding state machine. This method is universal, but applicable only to small system, because of the state explosion problem. Other methods are more efficient, but applicable only for subclasses of Petri nets or particular cases.

2) Model Checking [27] is another approach for the analysis of finite-state concurrent systems. The problem of deciding if a concurrent system satisfies a logic formula is called the model checking problem.

3) Linear Algebraic Methods.

One of linear algebraic methods is State equation [83].

Matrix equations govern the dynamic behavior of concurrent systems modeled by Petri nets. However, the solvability of of these equations is limited because of the

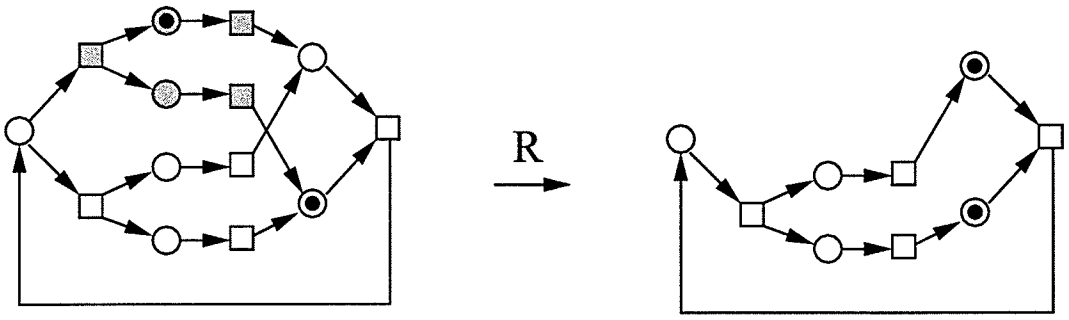

Fig. 2.7 A Reduction Rule.

constraint that solutions must be found as non-negative integers.

State equation gives us a necessary condition of reachability of a marking M from some other marking M^0.

$$M^0 = M + C\sigma$$

where σ is column vector of nonnegative integers and is called the firing count vector. The i-th entry of σ denotes the number of times that transition i must fire to transform M^0 to M.

Other linear algebraic methods are invariants [83, 82] and Rank Theorems [13, 20].

4) Reduction and decomposition [10, 11]. To facilitate the analysis of a large system, we often reduce the system model to a simpler one, while preserving the system properties to be analyzed.

A reduction rule is shown in Fig. 2.7

5) Structural methods.

One example is siphon - trap method [34]. Behavioral properties of free-choice Systems and some other subclasses of Petri nets are closely related to the structural properties. Two basic structural notions are a siphon and a trap. A set $D \subseteq S$ is a *siphon* (a *trap*) iff ${}^\bullet D \subseteq D^\bullet$ ($D^\bullet \subseteq {}^\bullet D$). A set of places is called *marked* iff at least one of its places has a token. Those sets of places play an important role in the behavior of Petri nets. We can notice that unmarked siphons remain unmarked at

any firing of transitions, and marked traps remain marked. It can be easily shown that live systems have no unmarked siphons.

Another example is circuit - theoretic analysis [83]. Liveness and boundedness of marked graphs are closely related to the structural properties, and the basic structural notion in marked graphs is a circuit.

2.5 Synthesis Methods for Petri Nets

Analysis techniques detect non-correct systems, but in general do not give any hint about how to proceed in order to improve the design. In practice the designer tries to synthesize a system, and then to verify it. If the system is not correct in some sense, the designer redesigns it and so on.

Synthesis methodologies are an interesting alternative to these trial and error methods based on analysis and modification. In these cases, the designer restricts him/herself to modifying and developing the model using only some very specific rules of top-down transformation and/or composition, which can be safely applied because they are known to preserve the properties desired for the system.

The synthesis problem can be stated as follows: given a set of properties of good behavior, how does one construct systems satisfying them?

There are two basic synthesis methodologies for Petri nets.

1) Stepwise refinement (a kind of top-down approach). In the top-down design methodology, to which this section is devoted, the synthesis procedure starts from an elementary system. This elementary system is usually small or simple, and satisfies the properties of good behavior. This initial system is then enlarged in a stepwise way using the synthesis rules kit.

2) Modular synthesis (a kind of bottom-up approach). The second of the two basic synthesis methodologies is modular synthesis. It deals with the case when modules (subsystems) are merged (composed) into new systems. The system is divided into

modules that can be easily modeled. Then such representations should be coordinated to describe the whole system. So, the problem of modular synthesis can be stated as follows: given a set of well-behaved modules, how does one compose the modules to yield a well-behaved system. Usually the two central properties of Petri nets: liveness and boundedness are considered when determining the criteria of good behavior.

Chapter 3

New Rank Theorems

Rank Theorems give us a necessary and/or sufficient conditions of liveness and boundedness. These conditions are expressed in linear algebraic terms. Moreover, the theorems allow us to check the behavioral properties in polynomial time.

There are already some Rank Theorems for different classes of Petri nets [13, 20, 24]. In [51, 60, 56] new Rank Theorems for Petri nets are presented. These formulations strengthen the known ones in the sense that the new necessary and sufficient conditions for a Petri net to be live and bounded are less strict than the known ones.

3.1 Subnets

Let $N = (S, T, F)$, $S' \subseteq S$, $T' \subseteq T$. Then (S', T', F') is a *subnet* of N iff $F' = F \cap ((S' \times T') \cup (T' \times S'))$. Since F' is completely defined by S' and T', we will write (S', T') instead of (S', T', F').

Lemma 3.1.1 [82] If J is a semi-positive $S-(T-)$invariant then the support $\langle J \rangle$ satisfies

$$\langle J \rangle^{\bullet} = {}^{\bullet}\langle J \rangle \tag{3.1}$$

■

A set $T_1 \subseteq T$ is a *T-component* iff $(S_1 = {}^\bullet T_1, T_1)$ is a strongly connected subnet of N satisfying 3.1, 2.1 and 2.2.

Let $N = (S, T, F)$ be a net. Then $N^{-d} = (T, S, F^{-1})$ is *reverse-dual* net of net N. The notion of an *S-component* is reverse-dual to that of a T-component. I.e. a set $S_1 \subseteq S$ is a *S-component* iff $(S_1, T_1 = S_1^\bullet)$ is a strongly connected subnet of N satisfying 3.1, 2.3 and 2.4.

A net is *T-coverable* iff every transition of it belongs to some T-component. We call a cover of T-components by *T-cover*.

Lemma 3.1.2 [22] If T_1 is a T-component of N then $\chi(T_1)$ is a minimal T-invariant of N.

■

Lemma 3.1.3 [22] If J is a minimal T-invariant of a well-formed EFC-net N then the support $\langle J \rangle$ is a T-component of N.

■

Lemma 3.1.4 [8] If N is well-formed then it is strongly connected.

■

The following result is well-known from Linear Algebra.

Lemma 3.1.5 $rank(C) = |T| - dim(T)$.

■

In [20] a necessary and sufficient condition of liveness and boundedness of EFC-systems is given. It is the Rank Theorem.

Theorem 3.1.6 [20] Let N be an EFC-net. Then (N, M^0) is live and bounded iff (N, M^0) is regular.

■

The Rank Theorem does not hold for arbitrary systems, but provides a sufficient condition for liveness and boundedness.

Let $[x]$ be a cluster of N. A *feedback* [19] of $[x]$ is an arc $(t, s) \in F$ such that $s \in [x], t \in [x], (s, t) \notin F$. A net N is *feedback-free* [22] iff no cluster has feedbacks.

Lemma 3.1.7 [19] Every regular system is live, bounded and feedback-free.

■

Let N be feedback-free. The mapping η is defined by $\eta(N) = (S, T, F \cup \widehat{F} \cup \widehat{F}^{-1})$ where $\widehat{F} = \{(s,t) \in S \times T \mid [s] = [t] \wedge (s,t) \notin F\}$ and $\widehat{F}^{-1} = \{(t,s) | (s,t) \in \widehat{F}\}$. The net $\eta(N)$ is called the *EFC-representation* of N in [19].

Lemma 3.1.8 [19] Let N be feedback-free. Then

(a) $\eta(N)$ is an EFC-net.

(b) $C = C'$, where C' is the incidence matrix of $\eta(N)$ (i.e. both nets have identical incidence matrices).

(c) If (N, M) is regular then so is $(\eta(N), M)$.

■

Now we give a necessary and sufficient condition of regularity expressed in terms of the EFC-representation.

Lemma 3.1.9 Let N be a net.

(a) [57] (N, M) is regular iff $(\eta(N), M)$ is live and bounded and N is feedback-free.

(b) If N is strongly connected then so is $\eta(N)$.

Proof:

(a) $\Rightarrow$. By Lemma 3.1.7, N is feedback-free. By Lemma 3.1.8a, $\eta(N)$ is an EFC-net. By Lemma 3.1.8c, (N', M) is regular. Hence by Lemma 3.1.7, $(\eta(N), M)$ is live and bounded.

$\Leftarrow$ By Lemma 3.1.6, $(\eta(N), M)$ is regular. Since N is feedback-free, by Lemma 3.1.8b, (N, M) is regular.

(b) The proof is obvious.

■

Now we give a simple algorithm to construct the EFC-representation of N, which simultaneously checks whether a net is feedback-free.

Algorithm 3.1.1 *EFC-representation*

Input: a net N.

Output: the EFC-representation $N' = \eta(N)$ of N or the **message** "N is not feedback-free and hence not regular".

begin

$N' := N$; $A_1 := A$ (the set of the clusters of N)

while $A_1 \neq \emptyset$ **do**

choose $a \in A_1$; $A_1 := A_1 \backslash a$; $S_a := S \cap a$; $T_1 := T \cap a$;

while $S_a \neq \emptyset$ **do**

choose $s \in S_a$; $S_a := S_a \backslash s$; $T_a := T_1$;

while $T_a \neq \emptyset$ **do**

choose $t \in T_a$; $T_a := T_a \backslash t$;

if $(s,t) \notin F$ and $(t,s) \in F$ **then Stop, Output Message, endif**

if $(s,t) \notin F$ and $(t,s) \notin F$ **then** $F' := F' \cup (s,t) \cup (t,s)$ **endif**

endwhile

endwhile

endwhile

end

Proposition 3.1.10 [57] The problem to decide if net N is feedback-free and to construct EFC-representation of N has the complexity $O(|S| \times |T|)$.

Proof:

By definition a cluster is a connected (and also strongly connected) component of the relation $F_1 \cup F_1^{-1}$, where $F_1 = F \cap (S \times T)$. Hence the set of all clusters can be computed with the $O(|S| \times |T|)$-algorithm of [94]. Since the clusters are a partition on the set of nodes, the result follows.

■

Lemma 3.1.9a reduces the problem of regularity to the problem of liveness and boundedness of EFC-systems.

Lemma 3.1.11 [57] Let N satisfy 2.5, 2.6, 2.7, and $N' = \eta(N)$ be its EFC-representation.

(a) $F \subseteq F'$.

(b) Nodes $dom(F' \backslash F) \cup cod(F' \backslash F)$ belong to clusters with more than one place and more than one transition in both N and N'.

(c) T_1 is a T-component of N iff T_1 is a T-component of N'.

(d) S_1 is an S-component of N iff S_1 is a minimal S-invariant of N.

Proof:

(a) and (b) easy to follow from the definition of the EFC-representation.

(c) $\Leftarrow$. Let T_1 be a T-component of N'. Then by Lemmata 3.1.8b and 3.1.11a, the subnet $({}^\bullet T_1, T_1)$ of N has the same set of transitions T_1, but (possibly) does not have

all the self-loops compared to the subnet $({}^\bullet T_1, T_1)$ of N'. So the subnet $({}^\bullet T_1, T_1)$ of N is strongly connected and satisfies 2.2 and 2.1. By Lemma 3.1.2, $\chi(T_1)$ is a minimal T-invariant of N'. By Lemma 3.1.8b, the incidence matrices of N and N' are the same, and hence $\chi(T_1)$ is a minimal T-invariant of N. By Lemma 3.1.1, ${}^\bullet T_1 = T_1^\bullet$ in N. Hence by definition, T_1 is a T-component of N.

$\Rightarrow$. Let T_1 be a T-component of N. By Lemma 3.1.2, $\chi(T_1)$ is a minimal T-invariant of N. By Lemma 3.1.8b, and hence $\chi(T_1)$ is a minimal T-invariant of N'. By Lemma 3.1.3, T_1 is a T-component of N'.

(d) follows from Lemmata 3.1.2, 3.1.3, 3.1.11 and reverse-duality.

■

Let N be a net. The net $\phi(N)$ is defined as the result of performing the following operations for every cluster $[x]$ of N and every feedback (t, s) of c:

(i) remove the arc (t, s);

(ii) add a new place s_1 and a new transition t_1;

(iii) add arcs (t, s_1), (s_1, t_1) and (t_1, s).

It is easy to see that this transformation always terminates, and does not depend on the order in which the feedbacks are treated. The following lemma provides some properties of the transformation $\phi(N)$.

Lemma 3.1.12 Let N be a net and $N' = \phi(N)$.

(a) [22] The net N' is feedback-free.

(b) If N is T-coverable so is N'.

(c) Let n be the number of feedbacks of N. Then $rank(C') = rank(C) + n$.

Proof:

(b),(c) The proof is easy to see.

■

3.2 Siphons

In this section we will show that every strongly connected EFC-net is covered by minimal siphons.

A set $D \subseteq S$ is a *siphon* (a *trap*) iff ${}^\bullet D \subseteq D^\bullet (D^\bullet \subseteq {}^\bullet D)$.

Theorem 3.2.1 [34] An EFC-system (N, M^0) is live iff every siphon has a marked trap.

■

Theorem 3.2.2 [34] Let N be a well-formed EFC-net. Then every minimal siphon is an S-component.

■

To simplify the finding of siphons in EFC-nets, we define a simple transformation rule γ, i.e. a mapping, which domain is the class of EFC-nets and range 1-out nets.

Let N be an EFC-net. The net $\gamma(N)$ is defined as the result of transformation of the transitions of every cluster $[x]$ into a single transition t of $\gamma(N)$:

${}^\bullet t = {}^\bullet t_1,\ t_1 \in a$;

$t^\bullet = \bigcup_{t_1 \in a} t_1^\bullet$.

In Fig. 3.1 we see a cluster $\{s_1, s_2, t_1, t_2\}$ of an EFC-net N together with output places, and in Fig. 3.2 the corresponding transition t of 1-out net $\gamma(N)$ together with input and output places.

Lemma 3.2.3 Let $N = (S, T, F)$ be an EFC-net. Then

(a) D is a siphon of N iff D is a siphon of $\gamma(N)$;

(b) D is a minimal siphon of N iff D is a minimal siphon of $\gamma(N)$;

(c) Let $R \subseteq S$. D is a maximal siphon of N in R iff D is a maximal siphon of $\gamma(N)$ in R.

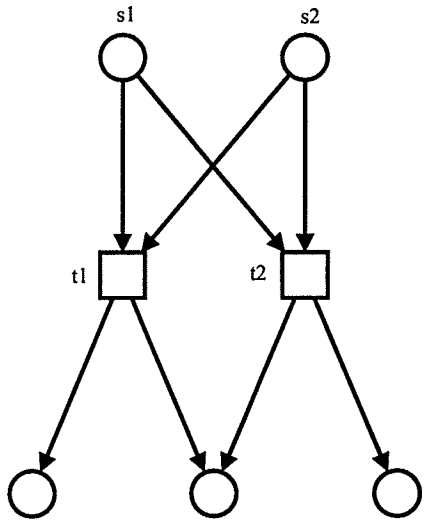

Fig. 3.1 A Cluster with Output Places

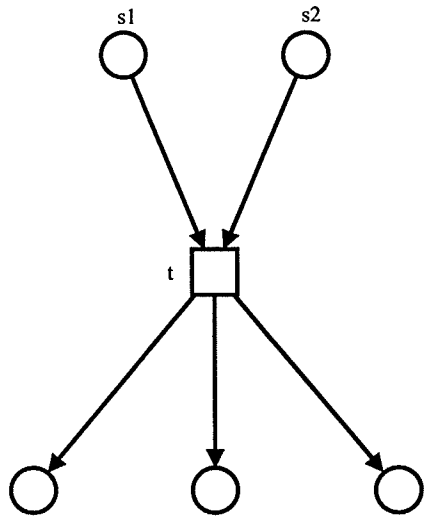

Fig. 3.2 A Transition after γ Transformation

Proof:

(a) $\Rightarrow$. Let $t \in {}^\bullet D$ in N. Since D is a siphon, $t \in D^\bullet$. By the EFC-property, all transitions of the same cluster have the same set of input places. Hence in $\gamma(N)$, the corresponding transition $t_c \in D^\bullet$ and ${}^\bullet D \subseteq D^\bullet$. Hence D is a siphon of $\gamma(N)$.

$\Leftarrow$. Analogously.

(b) and (c) easy follow from (a).

■

A *path* of N is a nonempty sequence $x_1, \ldots, x_k$ of nodes satisfying (x_1, x_2), ..., $(x_{k-1}, x_k) \in F$. It is a *circuit* iff $x_k = x_1$. An *elementary* path (circuit) is a path (circuit) which does not contain any node twice (except maybe $x_k = x_1$).

A subnet (S, T) is said to be *generated* by the set of places S iff ${}^\bullet S = T$.

The following lemma gives some structural properties of minimal siphons in EFC-nets.

Lemma 3.2.4 [22] Let D be a minimal siphon of an EFC-net. Then $(D, {}^\bullet D)$ is strongly connected and 2.4 holds for $(D, {}^\bullet D)$.

■

We give now some structural property of minimal siphons in 1-out nets.

Lemma 3.2.5 Let D be a set of places of an 1-out net. Then D is a minimal siphon iff $(D, {}^\bullet D)$ is a maximal strongly connected subnet and 2.4 holds for $(D, {}^\bullet D)$.

Proof:

$\Rightarrow$. By Lemma 3.2.4, $(D, {}^\bullet D)$ is strongly connected and 2.4 holds for $(D, {}^\bullet D)$. We continue indirect. Let $(D, {}^\bullet D)$ be not maximal with those properties. Then there exists a maximal strongly connected subnet $(S_1, T_1) \supset (D, {}^\bullet D)$ satisfying 2.4. We have two cases.

(a) $D = S_1$. Then ${}^\bullet D \subset T_1$. Hence $T_1 \backslash {}^\bullet S_1 \neq \emptyset$ and (S_1, T_1) is not strongly connected. A contradiction.

(b) $D \subset S_1$. Since (S_1, T_1) is strongly connected, there exists a path $(S_1 \backslash D) \ni s_1 t s_2 \in D$. Since $t \in {}^\bullet D$, and $(D, {}^\bullet D)$ is strongly connected, $|{}^\bullet t \cap S_1| \geq 2$. A contradiction to 2.4.

$\Leftarrow$. Lemma 3.2.4.

■

Lemma 3.2.6 [57] Let $N = (S, T, F)$ be a strongly connected 1-out net. Then it is covered by

(a) elementary circuits;

(b) subnets generated by minimal siphons.

Proof:

(a) trivial.

(b) By (a), N is covered by its elementary circuits. By definition, every elementary circuit is a strongly connected subnet of N with 2.4. By Lemma 3.2.5, N is covered by minimal siphons.

■

Theorem 3.2.7 Let $N = (S, T, F)$ be a strongly connected EFC-net. The arcs of $F \cap (T \times S)$ of N and hence the nodes $S \cup T$ are covered by subnets generated by minimal siphons.

Proof:

follows from Lemmata 3.2.6 and 3.2.3b.

■

An $S - \textit{allocation}$ is the set $\alpha \subseteq S$ such that for every cluster $[x] \in \mathcal{A}$ $|[x] \cap \alpha| = 1$.

Lemma 3.2.8 Let N be a strongly connected EFC-net and D be a minimal siphon of N. Then there exists an S-allocation α of N such that $(\alpha, \alpha^\bullet)$ is connected, D is only siphon satisfying $D \subseteq \alpha$ and D is the only source of $(\alpha, \alpha^\bullet)$.

Proof:

(Constructive).

$\alpha_0 := S_0 := D$;

$S_{i+1} = \bigcup\{S_c | S_c$ is a minimal siphon and $S_c^\bullet \cap S_i^\bullet \neq \emptyset\}$;

$\alpha_{i+1} := \alpha_i \cup \delta_i$, where δ_i is a maximal set satisfying $\delta_i \subseteq S_{i+1} \backslash (\cup_{s \in S_i}[s])$ and $\forall a \in \mathcal{A}$ $|a \cap \delta_i| \leq 1, i = 0, 1, \ldots$

By Theorem 3.2.4, $\exists k \in \mathbb{N}$ such that $S_0 \subseteq S_1 \subseteq \ldots \subseteq S_k = S$.

Claim. Let $D \subseteq \alpha$ be defined as above and $D' \subseteq \alpha$ be a minimal siphon. Then $D = D'$.

Proof of the claim. By induction on k.

Basis. $D' \subseteq S_k$ since $S_k = S$.

Step. if $D' \subseteq S_{i+1}$ then $D' \subseteq S_i, i < k$.

Since $D' \subseteq S_{i+1} \cap \alpha$, by the definition of S_{i+1} $D'^\bullet \cap S_i^\bullet \neq \emptyset$ and by the definition of α_{i+1} : $\emptyset \neq \alpha_{i+1} \cap {}^\bullet(D'^\bullet \cap S_i^\bullet) \subseteq \alpha_i \cap {}^\bullet(D'^\bullet \cap S_i^\bullet) \subseteq D' \cap S_i$. Hence $D' \cap S_i \neq \emptyset$. Since D' is a minimal siphon, by Lemma 3.2.4, $(D', {}^\bullet D')$ is strongly connected. Hence $\forall s, s' \in D'$ there exists a path $s = s_0 t_0 s_1 \ldots s_n = s'$ such that $\{s_0, \ldots, s_n\} \subseteq D'$, $\{t_0, \ldots, t_{n-1}\} \subseteq {}^\bullet D'$. Since D' and S_i are siphons then ${}^\bullet(\alpha_i \cap {}^\bullet(D^\bullet \cap S_i^\bullet)) \subseteq (D \cap S_i)^\bullet$. Since D is a minimal siphon and by Theorem 3.2.7, S_i is covered by minimal siphons, $D \subseteq S_i$.□ Claim.

We proceed indirectly. Let $(\alpha, \alpha^\bullet)$ be disconnected. Since α is a siphon, all the connected components of which are also siphons. This contradicts the claim. Let $(\alpha, \alpha^\bullet)$ have more than one source. Then it is easy to see that all they are subnets $(D, {}^\bullet D)$, where D are siphons. Again a contradiction to the claim.

■

3.3 Linear Algebra in Net Theory

Lemma 3.3.1 (a) [82] Let N be a net satisfying

$$\exists J \geq 1 \qquad C \cdot J \leq 0 \tag{3.2}$$

Then $\forall X$ if $(X \cdot C \geq 0)$ then $(X \cdot C = 0)$.

(b) [82] N is structurally bounded iff N satisfies

$$\exists J \geq 1 \qquad J \cdot C \leq 0 \tag{3.3}$$

(c) [89] If N is well-formed then it has a positive T-invariant.

(d) [22] If N has positive S- and T-invariants then it is strongly connected.

(e) [92] If N is structurally live and structurally bounded then it is strongly connected and has a positive T-invariant.

■

Lemma 3.3.2 (a) Let $Q \subseteq S \quad \forall t \in Q^\bullet \quad |{}^\bullet t \cap Q| = 1$. Then Q is a trap iff $\chi(Q) \cdot C \geq 0$.

(b) Let N be a net satisfying 3.2, and Q be a trap such that 2.4 holds then $\chi(Q)$ is an S-invariant and 3.1, 2.3 hold for Q.

(c) If N is covered by S-components then it has a positive S-invariant and satisfies 3.3.

(d) If N is covered by T-components then it has a positive T-invariant and satisfies 3.2.

Proof:

(a) is trivial.

(b) Let $Q \subseteq S$ be a trap such that 2.4 holds. Then by Lemma 3.3.2a, $\chi(Q) \cdot C \geq 0$. By Lemma 3.3.1a, $\chi(Q)$ is an S-invariant. Since 2.4 holds, and $\chi(Q)$ is an S-invariant, it is easy to see that 2.3 holds. By Lemma 3.1.1, 3.1 holds.

(c) The sum of characteristic vectors of all S-component of a covering is a positive S-invariant.

(d) is reverse-dual to (c).

■

Lemma 3.3.3 [20] Let N be an EFC-net, α be an S-allocation of N. Then

(a) If N is well-formed then $rank(C) \leq |\mathcal{A}| - 1$.

(b) $rank(C) \geq |\mathcal{A}| - dim(\alpha)$.

■

Lemma 3.3.4 Let N be an EFC-net, α be an S-allocation of N. If $\exists J$ such that $J \cdot C \geq 0$ and $\langle J \rangle \subseteq \alpha$ then $J^+ \cdot C \geq 0$.

Proof:

Let $t \in T$. Then $J^+ \cdot C(t) = \sum_{s \in t^\bullet} J^+(s) - \sum_{s \in {}^\bullet t} J^+(s)$. Let $\alpha \cap {}^\bullet t = \{p\}$. Two cases are possible.

(i) $J^+(p) = 0$. Then $J^+ \cdot C(t) \geq 0$.

(ii) $J^+(p) \geq 0$. By definition of J^+, $J^+(p) = J(p)$ and $\forall s \in t^\bullet \quad J^+(s) \geq J(s)$. Since $J \cdot C \geq 0$ we have $\sum_{s \in t^\bullet} J^+(s) \geq \sum_{s \in t^\bullet} J(s) \geq \sum_{s \in {}^\bullet t} J(s) = J(p) = J^+(p)$.

Hence $\forall t \quad J^+ \cdot C(t) \geq 0$.

■

Lemma 3.3.5 Let N be an EFC-net satisfying 3.2. Then

(a) If D is a minimal siphon, and $Q \subseteq D$ is a trap then D is an S-component.

(b) If N is strongly connected then $rank(C) \geq |\mathcal{A}| - 1$.

(c) If N is strongly connected and $rank(C) < |\mathcal{A}|$ then N is well-formed.

Proof:

(a) Since $Q \subseteq D$, by Lemma 3.2.4, $(Q, {}^\bullet Q)$ satisfies 2.3. By Lemma 3.3.2b, Q satisfies 3.1 and 2.3. Hence Q is a siphon. By minimality of D, $Q = D$. Hence 3.1 and 2.3 hold for D.

(b) Let D be a minimal siphon. By Lemma 3.2.8, there exists an α such that $D \subseteq \alpha$, and for every other siphon D', $D' \not\subseteq \alpha$. Let $J \in dim(\alpha)$. By Lemma 3.3.4, $J^+ \cdot C \geq 0$. By Lemma 3.3.1a, J^+ is also an S-invariant and semi-positive. By Lemma 3.1.1, $\mathcal{S}(J)$ is a siphon. By minimality of D, $\mathcal{S}(J) = D$. Hence $dim(\alpha) \leq 1$. The result follows by Lemma 3.3.3b.

(c) Let D be a minimal siphon. By Lemma 3.2.8, there exists an α such that $D \subseteq \alpha$, and for every other siphon D', $D' \not\subseteq \alpha$. Since $rank(C) < |\mathcal{A}|$, by Lemma 3.3.3b, $\dim(\alpha) > 0$. Hence there exists an S-invariant $J \in J_\alpha$. By Lemma 3.3.4, $J^+ \cdot C \geq 0$. By Lemma 3.3.1a, J^+ is an S-invariant. By Lemma 3.1.1, $\langle J^+ \rangle$ is a trap. Hence every minimal siphon has a trap, and by Theorem 3.2.1, N is structurally live. By Lemma 3.3.5a, every siphon is an S-component. By Theorem 3.2.7, N is covered by minimal siphons and hence by S-components. By Lemma 3.3.2c, N satisfies 3.3. By Lemma 3.3.1b, N is structurally bounded.

■

Lemma 3.3.6 If a net is T-coverable then $rank(C) \geq |\mathcal{A}| - 1$.

Proof:

Let N be T-coverable and n be the number of feedbacks of N. Then by Lemma 3.1.12, $\phi(N)$ is feedback-free and T-coverable. By Lemma 3.1.8a, the net $N' = \eta(\phi(N))$ is an EFC-net. By definition of the EFC-representation, the number of clusters in N and $\eta(N)$ is the same. Hence the number of clusters in N': $|\mathcal{A}'| = |\mathcal{A}| + n$. By Lemma 3.1.8b, rank of the incidence matrices of $\phi(N)$ and N' is the same. Hence by Lemma 3.1.12c, $rank(C') = rank(C) + n$. Since $\phi(N)$ is T-coverable, it is strongly connected and satisfies 3.2. Hence by Lemma 3.1.9b, N' is strongly connected, and

by Lemma 3.1.8b, satisfies 3.2. By Lemma 3.3.5b $rank(C') \geq |\mathcal{A}'| - 1$. Hence $rank(C) + n \geq |\mathcal{A}| + n - 1$, and $rank(C) \geq |\mathcal{A}| - 1$.

■

It is known that the theory of regular systems exhibits a nice duality [4], strictly speaking, reverse-duality, i.e. when a proposition for $N = (S, T, F)$ remains true for *reverse-dual* net $N^{-d} = (T, S, F^{-1})$.

Theorem 3.3.7 [22] Let N be a net. Then N is regular iff the reverse-dual net N^{-d} is regular.

■

Lemma 3.3.8 Let N be a net. Then

(a) [34] N is well-formed EFC-net iff N^{-d} is well-formed EFC-net;

(b) N is strongly connected iff N^{-d} is strongly connected;

(c) 3.2 holds for N iff 3.3 holds for N^{-d};

(d) $rank(C) = rank(C^{-d})$;

(e) $|\mathcal{A}| = |\mathcal{A}^{-d}|$;

(f) N satisfies 2.5 iff N^{-d} satisfies 2.6.

Proof:

(b)-(f) are obvious.

■

3.4 A Concise Proof of the Coverability Theorem for Live and Bounded Extended Free-Choice Nets

The known proofs of the well-known coverability theorem for free-choice nets consider system behavior, firing sequences or processes [5, 22] and thus tend to be long. Here a concise proof based only on structural properties of nets is given.

The notions of *T-allocation*, *absorber* and *generator* are reverse-dual to the notions of S-allocation, siphon and trap respectively. A net is *S-(T-)covered* iff it is covered by S-(T-)components.

Lemma 3.4.1 *A necessary condition of liveness*

Let N be an EFC-net having a live marking. Then every T-allocation of N contains a generator.

Proof:

Let $T_1 \subseteq T$ be a T-allocation, and M^0 be a live marking of N. Then the system $(S, T_1, F \cap ((S \times T_1) \cup (T_1 \times S), M^0)$ has non-empty set of live transitions $T_2 \subseteq T_1$. It is obvious that T_2 is a generator.

■

Theorem 3.4.2 *Coverability Theorem*

[5, 22] Let N be an EFC-net having a live and bounded marking. Then it is S-covered and T-covered.

Proof:

By Lemmata 3.1.4 and 3.3.1c, N is strongly connected and has a positive T-invariant. Hence N satisfies 3.2. Since N has a live marking, by Theorem 3.2.1, every minimal

siphon D contains a trap $Q \subseteq D$. Then by Lemma 3.2.4, $(Q, {}^\bullet Q)$ satisfies 2.4. By Lemma 3.3.2b, Q satisfies 3.1 and 2.3. Hence Q is a siphon. By minimality of D, $Q = D$. Hence 3.1 and 2.3 hold for D, and D is an S-component. By Lemma 3.2.7, N is S-covered.

By Lemma 3.1.4, N is strongly connected. Since N is S-covered, the sum of characteristic vectors of all S-components of a cover is a positive S-invariant. Let A be a minimal absorber of N. By reverse-dual version of Lemma 3.2.8, there exists a T-allocation α such that $A \subseteq \alpha$ and for every other absorber A', $A' \not\subseteq \alpha$. Since N has a live marking, by Lemma 3.4.1, every T-allocation contains a generator. Hence there exists a generator $G \subseteq \alpha$ and $\forall s \in {}^\bullet G \quad |s^\bullet \cap G| = 1$. By reverse-dual version of Lemma 3.3.2b, G satisfies 3.1 and $\forall s \in {}^\bullet G \quad |{}^\bullet s \cap G| = 1$. Hence G is an absorber. By minimality of A, $A = G$. Hence A is a T-component. By reverse-dual version of Lemma 3.2.7, N is T-covered.

■

3.5 New Rank Theorems

Theorem 3.5.1 Let N be an EFC-net. Then N is well-formed iff it is strongly connected and satisfies 3.2 and 2.7.

Proof:

$\Rightarrow$. By Lemma 3.3.3a, every well-formed EFC-net satisfies $rank(C) \leq |\mathcal{A}| - 1$. By Lemmata 3.1.4 and 3.3.1c, every well-formed EFC-net N is strongly connected and satisfies 3.1.4. Hence N satisfies 3.2. By Lemma 3.3.5b, $rank(C) \geq |\mathcal{A}| - 1$.

$\Leftarrow$. By Lemma 3.3.5c, N is well-formed.

■

Now we give the reverse-dual version of Theorem 3.5.1.

Theorem 3.5.2 Let N be an EFC-net. Then N is well-formed iff it is strongly connected, and 3.3 and 2.7 hold.

Proof:

By Lemma 3.3.8a, N well-formed iff N^{-d} is well-formed. By Theorem 3.5.1, N^{-d} is well-formed iff N^{-d} is strongly connected, and 3.2 and 2.7 hold for N^{-d}. By Lemma 3.3.8(b-e), N is strongly connected, and 3.3 and 2.7 hold for N.

■

By Lemma 3.3.1b, 3.3 is equivalent to structural boundedness, so we get a nice necessary and sufficient condition of structural liveness in structurally bounded EFC-nets.

Theorem 3.5.3 Let N be a structurally bounded EFC-net. Then N is structurally live iff it is strongly connected, and 2.7 holds.

Proof:

$\Rightarrow$. Let N be well-formed. Then by Theorem 3.5.2, it is strongly connected, and 2.7 holds.

$\Leftarrow$. Let N be a structurally bounded, strongly connected, and 2.7 holds. By Lemma 3.3.1b, N satisfies 3.3. By Theorem 3.5.2, N is well-formed.

■

Theorem 3.5.4 Let N be an EFC-net. Then N is well-formed iff it is T-covered, and 2.7 hold.

Proof:

$\Rightarrow$. By Lemma 3.5.1, 2.7 hold. By Theorem 3.4.2, N is T-covered.

$\Leftarrow$. Since N is T-covered, it is strongly connected and satisfies 3.2. By Lemma 3.5.1, N is well-formed.

■

Lemma 3.5.5 If a net satisfies 2.5, 2.6, 2.7 then it is T-coverable.

Proof:

The proof follows from Lemmata 3.1.11c and 3.4.2.

■

Lemma 3.5.6 If a net satisfies 2.5, 2.6, 2.7 then it is S-coverable.

Proof:

The proof is reverse-dual to the Proof of the previous Lemma.

■

Lemma 3.5.7 Let N be a feedback-free net. Then

(a) if N is strongly connected then $\eta(N)$ is strongly connected;

(b) if N satisfies 3.2 then $\eta(N)$ satisfies 3.2;

(c) if N satisfies 2.7 then $\eta(N)$ satisfies 2.7.

Proof:

(a) is obvious. (b) and (c) follow from Lemma 3.1.8b.

■

Lemma 3.5.8 Let N be strongly connected and satisfy 3.2 and 2.7. Then N is feedback-free, and $\eta(\phi(N))$ is well-formed.

Proof:

By Lemma 3.1.12a, $\phi(N)$ is feedback-free, strongly connected and satisfies 3.2 and 2.7. By Lemma 3.1.8a, $\eta(\phi(N))$ is an EFC-net. By Lemma 3.5.7, $\eta(\phi(N))$ is strongly connected and satisfies 3.2 and 2.7. By Theorem 3.5.1, $\eta(\phi(N))$ is well-formed, and by Theorem 3.4.2, it is covered by S-components. We continue indirect. Let

N have a feedback (t, s), and K be an S-component of $\eta(\phi(N))$ such that $s' \in K$ (see the definition of $\phi(N)$ above). By definition of an S-component (condition 3.1), $\exists s_2 \in t_1^\bullet \cap K$. Since $t_1^\bullet = \{s\}$, $s \in K$ and $s_2 = s$. So $\{s, s'\} \subset K \cap t^\bullet$. A contradiction to 2.3.

■

Theorem 3.5.9 If N is strongly connected and satisfies 2.7 and 3.2 then N satisfies 2.7, 2.5, 2.6 and is well-formed and feedback-free.

Proof:

By Lemma 3.5.8, N is feedback-free, and $\eta(\phi(N))$ is well-formed. By Lemma 3.1.9a, N satisfies 2.7, 2.5 2.6, and by Lemma 3.1.7, N is well-formed.

■

Now we give the reverse-dual version of the Theorem 3.5.9.

Theorem 3.5.10 If N is strongly connected, and satisfies 2.7 and 3.3 then N satisfies 2.7, 2.5 2.6 and is well-formed.

Proof:

By Lemma 3.3.8(b-e), N^{-d} is strongly connected and satisfies 3.2 and 2.7. By Theorem 3.5.9, N^{-d} satisfies 2.7, 2.5 2.6. By Lemma 3.3.8(d-f), N is satisfies 2.7, 2.5, 2.6. By Theorem 3.1.7, N is well-formed.

■

Theorem 3.5.11 If N is a structurally bounded net, strongly connected, and 2.7 holds then N is structurally live.

Proof:

Let N be a structurally bounded, strongly connected and satisfies 2.7. By Lemma 3.3.1b, N satisfies 3.3. By Theorem 3.5.10, N is well-formed.

■

3.6 Additional Results

In [51, 60, 56] we prove several Rank Theorem for extended free-choice and general class of Petri nets, and show that they can be used as an alternative technique for the verification of workflow procedures.

Taking into account the next results from linear algebra, one can find two more "minimal" formulations of the Rank Theorem.

Lemma 3.6.1 [82] N has both a positive S-invariant and a positive T-invariant iff any of (a) or (b) hold

(a) N satisfies 3.3 and

$$\exists J \geq 1 \qquad C \cdot J \geq 0 \tag{3.4}$$

(b) N satisfies 3.2 and

$$\exists J \geq 1 \qquad J \cdot C \geq 0 \tag{3.5}$$

In these new formulations, the conditions 3.3 and 3.4 (correspondingly 3.2 and 3.5) should be the additional conditions to the rank equation.

Now we prove these new formulations.

Theorem 3.6.2 A net N is covered by T-components and satisfies 2.7 iff it satisfies 2.5, 2.6 and 2.7.

Proof:

$\Leftarrow$. By Theorem 3.5.5, N is covered by T-components.

$\Rightarrow$. Covering by T-components implies strong connectedness. By Lemma 3.3.2d, N satisfies 3.2. By Theorem 3.5.9, we are done.

■

Theorem 3.6.3 A net N is covered by S-components and satisfies 2.7 iff it satisfies 2.5, 2.6 and 2.7.

Proof:

follows from Theorem 3.6.2 and reverse-duality.

■

Now we show that all the existing Rank Theorems from [51, 60, 56] are simple corollaries of the new Rank Theorems and some known results from Petri net theory.

Theorem 3.6.4 [20] An EFC-net is well-formed iff it satisfies 2.5, 2.6 and 2.7.

Proof:

$\Rightarrow$. By Theorem 3.4.2, N is covered by S-components and T-components. By Lemma 3.3.2c, N satisfies 2.6 and 2.7. By Theorem 3.5.1, N satisfies 2.5.

$\Leftarrow$. By definition, the existence of a positive T-invariant implies 3.2. By Lemma 3.3.1d, the existence of both a positive S-invariant and a positive T-invariant implies strong connectedness. By Theorem 3.5.1, we are done.

■

Theorem 3.6.5 [24] An FC-net N is structurally live and structurally bounded iff it is covered by S-components and

$$rank(C) = |S| + |T| - |F \cap (S \times T)| - 1 \tag{3.6}$$

Proof:

Taking into account that the equation 2.7 is equivalent to 3.6 for the class of FC-nets, we can see that the present theorem is Theorem 3.6.3, applied for the class of FC-nets.

■

Theorem 3.6.6 [13] Let N be a strongly connected and structurally bounded FC-net. N is structurally live iff $rank(C) = |S| + |T| - |F \cap (S \times T)| - 1$.

Proof:

$\Rightarrow$. Theorem 3.5.1.

$\Leftarrow$. Theorem 3.5.3.

■

3.7 Summary

In [51, 60, 56] we prove several Rank Theorem for extended free-choice and general class of Petri nets, and show that they can be used as an alternative technique for the verification of concurrent systems. The Rank Theorems establish a close relationship between behavioral and linear-algebraic properties of Petri nets. We prove stronger versions of the Rank Theorems in the sense that existing Rank Theorems are corollaries of our theorems.

Chapter 4

Reduction of Petri Nets

In [49] we improve the complete reduction method of [25]. We develop the method in three more aspects. First, we apply the method for a wider class of systems. We give a reduction rule and prove its strong soundness and completeness for regular systems. Second, we give more precise evaluation of complexity of this method ($O(|S| \times |T|)$), comparing to [25] (polynomial in [25] without specifying the degree of polynomial). Third, in the method of [25] one of the reduction rules ($R3$) checks if all the siphons are nonempty at every step of its application in order to be strongly sound (original system is live and bounded iff reduced system is live and bounded). This check is cumbersome and actually is necessary only once at the first application of $R3$ because at each step of the reduction all the deadlocks are marked. In our method such a check is not required.

To simplify the reduction process and the corresponding propositions, we will use only one reduction rule R instead of two or more as in [22] and [25]. In this case we are forced to use live and bounded Marked Graphs as atomic systems instead of strongly connected systems consisting of two nodes. The analysis of Marked Graphs is very simple [3].

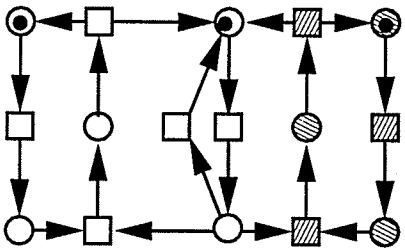

Fig. 4.1 A Regular System and one of its TCP-subnet (dashed nodes).

4.1 TCP-subnets in T-coverable Nets

Let N be a T-coverable net. Then non-empty and connected subnet $\widehat{N} = (\widehat{S}, {}^\bullet\widehat{S} \cup \widehat{S}^\bullet)$ is a *TCP-subnet* of N iff

(i) there exists a T-cover $\mathcal{C}$ of N and a T-component $N_1 \in \mathcal{C}$ such that

(ii) $\widehat{N} \subset N_1$.

(iii) $\overline{N} = (\overline{S}, \overline{T}) = (S \backslash \widehat{S}, T \backslash ({}^\bullet\widehat{S} \cup \widehat{S}^\bullet))$ contains some transition, and is strongly connected.

We denote $\widehat{T}_i = \overline{S}^\bullet \cap ({}^\bullet\widehat{S} \cup \widehat{S}^\bullet)$, $\overline{\mathcal{A}}_o$ the set of clusters of $\overline{N}$ which have common nodes with $\widehat{N}$ in N.

Theorem 4.1.1 [22] Let N be a well-formed EFC-net and $\widehat{N}$ be a TCP-subnet of N. Then $(\overline{S}, \overline{T})$ is well-formed and

$$|\widehat{T}_i| = 1 \tag{4.1}$$

■

The system in Fig. 4.1 is a regular system (a model of a non-deterministic wait process from [Mu]), and one of its TCP-subnets (dashed nodes) satisfies 4.1.

For the class of T-coverable nets we may notice that some transitions $t_i \in \widehat{T}_i$ may have pre-sets ${}^\bullet t_i$ containing the places of different clusters of $\overline{N}$, because we no longer have the EFC-property. In other words, after removal of a TCP-subnet, the number of clusters in the resulting net $\overline{N}$ may increase. In Fig. 4.2 we show the example,

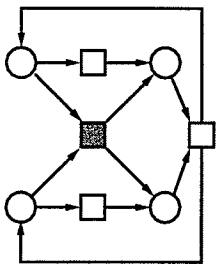

Fig. 4.2 Violation of Condition 4.2.

where a dashed transition represents a TCP-subnet and shares two of its input places with two clusters of $\overline{N}$. So we can have the situation when $\overline{N}$ satisfies 2.7, but N does not, or vice versa. To preserve Rank equation we need an additional condition:

$$|\overline{\mathcal{A}}_o| = 1 \tag{4.2}$$

Lemma 4.1.2 [49] Let $\widehat{N}$ be a TCP-subnet of a net N. Then

(a) $\widehat{N}$ is a T-net

(b) $Rank(\widehat{C}) = |\widehat{T}| - 1$

(c) $Rank(C) \geq Rank(\overline{C}) + Rank(\widehat{C})$

(d) $dim(T) \geq dim(\overline{T}) + 1$

(e) If a TCP-subnet $\widehat{N}$ of a net N satisfies 4.1 then $|\mathcal{A}| = |\overline{\mathcal{A}}| - |\overline{\mathcal{A}}_o| + |\widehat{T}|$

Proof:

(a) Let $\widehat{N}$ be a TCP-net. By definition of a TCP-net, there exists a T-component N_1 of N such that that $\widehat{N} \subset N_1$.

(b) By definition of a TCP-subnet, $\widehat{N}$ is connected. By Lemma 4.1.2a, $\widehat{N}$ is a T-net. Hence every T-invariant of $\widehat{N}$ is multiple of the vector $\overline{1}$. So $dim(\widehat{T}) = 1$. By Lemma 3.1.5, we get $Rank(\widehat{C}) = |\widehat{T}| - 1$.

(c) If we assume without loss of generality that the first rows of the matrix C correspond to the places of $\overline{N}$ and its first columns to the transitions of $\overline{N}$, then N can be decomposed in the following way:

$$C = \begin{pmatrix} \overline{C} & A \\ 0 & \widehat{C} \end{pmatrix}$$

for some matrix A.

(d) Every T-component containing $\widehat{N}$ is a T-invariant not belonging to the space of T-invariants of $\overline{N}$.

(e) Claim. Every cluster of N containing one node of $\widehat{N}$ contains exactly one transition of $\widehat{N}$.

By Lemma 4.1.2a, $\widehat{N}$ is a T-net. Hence every cluster containing one of $\widehat{N}$ contains at least one node of $\widehat{N}$.

We continue indirect. Let $t_1 \neq t_2$ and $t_1, t_2 \in {}^\bullet\widehat{S} \cup \widehat{S}^\bullet$ belong to the same cluster of N. By Lemma 4.1.2a, $\widehat{N} = (\widehat{S}, {}^\bullet\widehat{S} \cup \widehat{S}^\bullet)$ is a T-net. Then $\{t_1, t_2\} \subseteq T_i$. A contradiction to 4.1. □ Claim.

Divide the clusters of N into (1) those contained in $\overline{N}$, (2) those contained in $\widehat{N}$, and (3) those which contain nodes of both $\widehat{N}$ and $\overline{N}$.

The number of clusters of the first kind is $|\overline{\mathcal{A}}| - |\overline{\mathcal{A}}_o|$. By Claim, the number of clusters of the second kind is $|\widehat{T}| - 1$. The number of clusters of the third kind is 1.

■

4.2 A Reduction Rule, its Strong Soundness for T-coverable Systems and its Completeness for Regular Systems

Now we are ready to give a reduction rule for T-coverable systems.

Rule R. Let (N, M^0) be a feedback-free T-coverable system and not a T-system. $(\overline{N}, \overline{M}^0) = R(N, M^0)$ iff

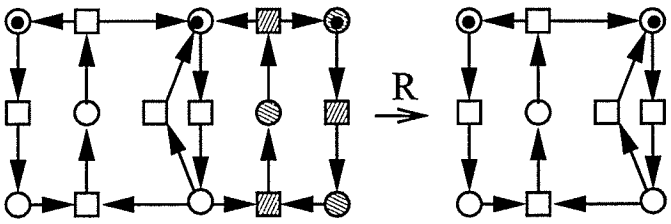

Fig. 4.3 Reduction Rule R.

Conditions on (N, M^0).

C1. N has a TCP-subnet $\widehat{N}$ of N with 4.1.1, 4.1.2 and

$$(\overline{S}, \overline{T}) \text{ is } T - \textit{coverable} \tag{4.3}$$

C2. Every loop of $\widehat{N}$ is marked.

Changes in (N, M^0) to produce $(\overline{N}, \overline{M}^0)$.

R1. $\forall s \in \widehat{T}^\bullet \cap \overline{S}$

$$\overline{M}^0(s) = \begin{cases} M^0(s) \text{ iff } \exists \text{ a non-marked path from } \widehat{T}_i \text{ to } s \text{ in } \widehat{N} \\ M^0(s) + 1, \text{ otherwise} \end{cases}$$

R2. Remove $\widehat{N}$ from N.

Fig. 4.3 illustrates Reduction Rule R.

Theorem 4.2.1 *(strong soundness of R)*

R is strongly sound with respect to the class of regular systems.

Proof:

Let $(\overline{N}, \overline{M}^0) = R(N, M^0)$. Using the definition of regular systems, we need to show the following:

(i) N satisfies 2.7, 2.5, 2.6 iff $\overline{N}$ satisfies 2.7, 2.5, 2.6.

(ii) M^0 satisfies 2.8 in N iff $\overline{M}^0$ satisfies 2.8 in $\overline{N}$.

(i) By Condition C1 of the Rule R, $(\overline{S},\overline{T})$ is T-coverable. Taking into account Lemmata 3.3.6 and 3.6.2, we need only to prove $rank(C) \leq |\mathcal{A}|-1$ iff $rank(\overline{C}) \leq |\overline{\mathcal{A}}|-1$.

$\Rightarrow$. By Lemmata 4.1.2bc, $Rank(C) \geq Rank(\overline{C}) + Rank(\widehat{C}) = Rank(\overline{C}) + |\widehat{T}| - 1$. So we get $Rank(\overline{C}) \leq Rank(C) - |\widehat{T}| + 1 = |\mathcal{A}| - |\widehat{T}|$. By Lemma 4.1.2e, $Rank(\overline{C}) \leq |\overline{\mathcal{A}}| - 1$.

$\Leftarrow$. By Lemma 4.1.2d, $dim(T) \geq dim(\overline{T}) + 1$. By Lemma 3.1.5, $Rank(C) \leq |T| - |\overline{T}| + Rank(\overline{C}) - 1$. Since $|T| = |\overline{T}| + |\widehat{T}|$, we have $Rank(C) \leq |\widehat{T}| + Rank(\overline{C}) - 1 = |\widehat{T}| + |\overline{\mathcal{A}}| - 2$. By Lemma 4.1.2e, $Rank(C) \leq |\mathcal{A}| - 1$.

We now show (ii). Taking into account Lemma 3.1.11d, we need to show that every S-component of (N, M^0) is marked iff every S-component of $(\overline{N}, \overline{M}^0)$ is marked.

$\Rightarrow$. Consider an S-component $\overline{S}_1$ of $\overline{N}$. Since $\overline{N}$ is a subnet of N, we have two cases.

(a) There is an S-component S_1 of N such that $\overline{S}_1 = S_1$. Since S_1 is marked in N, by the definition of R1, $\overline{S}_1$ is marked in $\overline{N}$.

(b) There is an S-component S_1 of N such that $\overline{S}_1 \subset S_1$. Then by the definition of R2, intersection of the subnet $(S_1, S_1^\bullet)$ with $\widehat{N}$ is a path in $\widehat{N}$ from $\{t_i\} = T_i$ to a place $s \in \widehat{T} \cap \widehat{S}$. By the definition of R1, if S_1 is marked by M^0 in N then $\overline{S}_1$ is marked by $\overline{M}^0$ in $\overline{N}$. So every S-component S_1 of N is marked.

$\Leftarrow$. Consider an S-component S_1 of N. Since $\overline{N}$ is a subnet of N, we have three cases.

(a) There is an S-component $\overline{S}_1$ of $\overline{N}$ such that $\overline{S}_1 = S_1$. Since $\overline{S}_1$ is marked in $\overline{N}$, by R1, S_1 is marked in N.

(b) There is an S-component $\overline{S}_1$ of $\overline{N}$ such that $\overline{S}_1 \subset S_1$. Continuation is similar to (b) above.

(c) $(S_1, S_1^\bullet)$ is a loop of $\widehat{N}$. Then by condition C2, it is marked in M^0.

■

To simplify the reduction process and the corresponding propositions, we will use only one reduction rule instead of two or more as in [22, 25]. In this case we are forced to use live and bounded T-systems as atomic nets instead of strongly connected system

consisting of two nodes. The complexity of analysis of T-systems (Marked Graphs) is lower, comparing to EFC-nets or regular nets [3].

Theorem 4.2.2 *(completeness of R)*

R is complete for the class of regular systems.

Proof:

Let (N, M^0) be a regular system.

We only need to show the following: (N, M^0) is a live and bounded T-system or has a TCP-subnet satisfying 4.1, 4.2, 4.3, and every loop of $\widehat{N}$ is marked.

Since (N, M^0) is regular, by Lemma 3.5.5, N is T-coverable and by Lemma 3.1.7, N is feedback-free.

We have 2 cases.

(i) N is a T-net. Then we are done.

(ii) N is not a T-net. We take a minimal cover $\mathcal{C}$ of N by T-components. Since N is not a T-net, we have $|\mathcal{C}| > 1$. We construct the (non-directed) graph $G = (V, E)$ as follows. V is the set $\mathcal{C}$ and E is the set of pairs (N_i, N_j) such that N_i and N_j have at least one common node. The graph G is connected because $\mathcal{C}$ is a cover of N and N is connected. Moreover, G has at least two nodes because $|\mathcal{C}| > 1$. We choose a spanning tree of G, and select one of its leaves, say N_1. We then construct a maximal set of nodes X of N_1 satisfying the following properties: (a) the net generated by X is connected, and (b) no element of X belongs to a T-component of $\mathcal{C} \setminus \{N_1\}$. The set X is nonempty, because $\mathcal{C}$ is a minimal cover. Since N_1 is a leaf of G, the subnet $\widehat{N}$ generated by X is connected, and by definition is a TCP-subnet. Since $\overline{N}$ covered by the set $\mathcal{C} \setminus N_1$, 4.3 holds. So it is only left to prove that 4.1 and 4.2 hold. Let $N' = \eta(N)$ be the EFC-representation of N. By Lemma 3.1.9a, N' is a well-formed EFC-net. By Lemma 3.1.11c, T_1 is a T-component of N iff T_1 is a T-component of N'. So the subnets $\widehat{N}'$ and $\overline{N}'$ of N' have the same sets of transitions as $\widehat{N}$ and $\overline{N}$ of

N. By Theorem 4.1.1, $\overline{N}'$ is well-formed, and $|\widehat{T}_i'| = 1$. By Lemma 3.1.11a, 4.1 holds for $\widehat{N}$. By Lemma 3.1.9a, $\overline{N}$ is well-formed. By Lemma 3.5.5, $\overline{N}$ is T-coverable. By Lemmata 4.1.1bc,

(ii1) $Rank(C) \geq Rank(\overline{C}) + |\widehat{T}| - 1$.

Since N and $\overline{N}$ both satisfy 2.5, 2.6 and 2.7, we have

(ii2) $Rank(C) = |\mathcal{A}| - 1$ and $Rank(\overline{C}) = |\overline{\mathcal{A}}| - 1$.

Substituting (ii2) into (ii1) we got $|\mathcal{A}| \geq |\overline{\mathcal{A}}| + |\widehat{T}| - 1$. Then by Lemma 4.1.2e, $|\overline{\mathcal{A}}| - |\overline{\mathcal{A}}_o| + |\widehat{T}| \geq |\overline{\mathcal{A}}| + |\widehat{T}| - 1$ and $|\overline{\mathcal{A}}_o| = 1$. So 4.2 holds and hence condition C1 holds.

Since (N, M^0) is regular, every loop of $\widehat{N}$ is marked by M^0. So condition C2 holds.

■

4.3 The Algorithm of Reduction and its Complexity

The algorithm below has two passes (two while loops). During the first pass, the algorithm checks if a given system (N, M^0) is T-coverable, finds a T-cover and corresponding TCP-subnets, and simultaneously memorizes TCP-subnets in a STACK. Actual reduction is done during the second pass (second while-loop).

Algorithm 4.3.1 Reduction (N)

Input:

(N, M^0) is a system

Output is one of the following messages:

1: "(N, M^0) is completely reduced "

2: "(N, M^0) is not completely reduced "

Function: $get - T - component(N, t)$

This function finds an T-component of N containing transition t. It is similar (and reverse-dual) to the function mark-S-component [41], but gives (as the output) the subnet $(\overline{S}, \overline{T})$ of transitions of a T-component containing t. If it finds a minimal set $T_1 \subseteq T$ such that $T_1^\bullet \subseteq {}^\bullet T_1$ and $T_1^\bullet \neq {}^\bullet T_1$, the algorithm stops and gives a message "(N, M^0) is not completely reduced".

Function: $strongconnect(N)$

This function checks if the graph of N is strongly connected and is based on the depth-first search algorithm. This function is described in [94].

Function: $get - TCP - subnet(\overline{N}, N_1, t)$

This function finds a TCP-subnet $\overline{N} \subseteq N_1$ of $\overline{N} \cup N_1$ containing t.

begin

if $strongconnect(N) = No$ **then**

  Stop with "N is not completely reduced"

endif

choose $t \in T$;

$(\overline{S}, \overline{T}) := get - T - component(N, t)$;

$T^o := \overline{S}^\bullet \backslash \overline{T}$;

while $(T^o \neq \emptyset)$ **do**

  choose $t \in T^o$; $T^o := T^o \backslash \{t\}$;

  $T_1 = get - T - component(N, t)$;

  $(\widehat{S}, \widehat{t}) := get - TCP - subnet(\overline{N}, N_1, t)$;

  $\overline{N} := \overline{N} \cup \widehat{N}$; $Push(STACK, \widehat{N})$;

endwhile

while $(STACK \neq \emptyset)$ **do**

  $\widehat{N} := Pop(STACK)$;

if $(\widehat{N}, M^0)$ satisfies (9,10) **then**

$(N, M^0) := R(N, M^0)$;

else Stop with "(N, M^0) is not completely reduced"

endif

endwhile

if (N, M^0) is live and bounded T-net **then**

Stop with "(N, M^0) is completely reduced";

else Stop with "(N, M^0) is not completely reduced";

endif

end

Proposition 4.3.2 The reduction algorithm has the complexity $O(|S| \times |T|)$.

Proof:

The complexity of function "strongconnect" for a directed graph $G = (V, E)$ is $O(|V| + |E|)$ [94]. The complexity for function "mark-S-component" is given in [41] ($O(|T|)$). Since the set of the TCP-subnets (under consideration) defines a partition on the set of nodes of the net, the complexity of the finding and checking of the TCP-subnets is $O(|S| \times |T|)$. Hence the upper Bound of the time complexity of the algorithm is $O(|S| \times |T|)$. The complexity of verifying if a given T-system is live and bounded is $O(|S| \times |T|)$.

■

4.4 Summary

An improvement to the reduction method given in [25] was presented. First, we generalized the method to regular Petri nets. Initially the method was used only for extended free-choice Petri nets. Second, we gave more precise evaluation to the

complexity of the method ($O(|S| \times |T|)$), versus polynomial without specifying the degree of polynomial in [25]. Third, we improve the Reduction Rule. We eliminate one computationally expensive step and reduce overall complexity of the reduction algorithm.

Chapter 5

The Algorithm to Decide if a Petri Net is Regular and its Complexity

In this chapter we present an $O(|S| \times |T|)$-algorithm to decide if a given net is regular [57].

We give a necessary and sufficient condition of regularity of a system. This condition proves to be liveness and boundedness of the EFC-representation of the system. Hence we reduce the problem of regularity to the problem of liveness and boundedness of EFC-systems.

To decide the well-formedness of a given EFC-net we give an inductive criterion of well-formedness. This criterion is based on complete reducibility of the class of nets [23].

In [54] we present an $O(|S| \times |T|)$-algorithm to decide if a given EFC-system is live and bounded. In the paper we refine this algorithm and apply it to decide if a system is regular.

To prove the soundness of our algorithm we use a theorem on covering of strongly connected EFC-nets by minimal siphons given above. The theorem was not formulated or proved anywhere previously, although for FC-nets the similar proposition follows from Theorem 4.2 [28]. The theorem from [28] uses an algorithm of constructing of

minimal siphon beginning from an arbitrary place. In our paper more formal proof is given.

It is well known from [34] that every initial marking of a well-formed EFC-net is bounded. Hence we need to check only liveness of a given marking.

To decide the liveness of an initial marking for well-formed EFC-nets, we define a simple transformation rule, i.e. a mapping, whose domain is the class of EFC-nets and range 1-out nets (every place has at most one output transition). This mapping proved to be useful twofold: when checking whether an EFC-net is well-formed, and when checking whether an initial marking of a well-formed EFC-net is live. It is shown that a well-formed EFC-system is live iff (if and only if) the maximal unmarked siphon of the corresponding 1-out net is the empty set. An $O(|S| \times |T|)$-algorithm to find the maximal unmarked siphon of 1-out net is given.

5.1 Liveness of Initial Marking

Since well-formedness is only a necessary condition for a system to be live and bounded, we need to decide if a given initial marking of a well-formed EFC-system is live and bounded or not.

Lemma 5.1.1 [34] Let N be an EFC-net. Then

(a) an initial marking M^0 of N is live iff every siphon contains a marked trap;

(b) if N is well-formed then every initial marking of N is bounded.

■

Lemma 5.1.2 [57] Let N be a well-formed EFC-net and M be a marking of N. Then the following equivalent:

(a) (N, M) is live and bounded;

(b) the maximal unmarked siphon of (N, M) is the empty set;

(c) the maximal unmarked siphon of $(\phi(N), M)$ is the empty set.

Proof:

$(a) \Rightarrow (b)$ (Indirect.) Let the maximal unmarked siphon D of (N, M) be not empty. From definition of a siphon it follows easy that an unmarked siphon stay unmarked at any reachable marking of N. Then all the transitions $D^\bullet$ can not be live. A contradiction.

$(a) \Leftarrow (b)$ Let D be a minimal siphon of N. Since the maximal unmarked siphon of N is the empty set, D is marked. By Theorem 3.2.2, D is an S-component, hence a trap. By Lemma 5.1.1a, (N, M) is live. By Lemma 5.1.1b, (N, M) is bounded.

$(b) \Leftrightarrow (c)$ follows from Lemma 3.2.3c.

■

Lemma 5.1.2 reduces the problem of liveness of a well-formed EFC-net N to the problem to decide if the maximal unmarked siphon of $\phi(N)$ is the empty set.

The maximal unmarked siphon of an 1-out net N can be obtained by the following algorithm, a modification of an algorithm in [91].

Algorithm 5.1.1 *get-max-unmarked-siphon*

Input: a 1-out system (N, M).

Output: maximal unmarked siphon D of (N, M).

begin

$D := S \backslash \{s \in S | M(s) > 0\}$; $T_d := T \backslash D^\bullet$; $T_1 := T \backslash T_d$

while $T_d \neq \emptyset$ **do**

$D := D \backslash T_d^\bullet$; $T_d := T_1 \backslash D^\bullet$; $T_1 := T_1 \backslash T_d$

endwhile

end

Theorem 5.1.3 [57] The following problems have the complexity $O(|S| \times |T|)$:

(a) to decide whether a strongly connected 1-out system (N, M) is live;

(b) to decide whether a well-formed EFC-system (N, M) is live.

Proof:

(a) Let 1-out net N be presented as two functions $S_o : S \to T$, $T_o : T \to 2^S$, defined as $S_o(s) = s^\bullet$, $T_o(t) = t^\bullet$. Since the sets T_d do not intersect in any two different iterations of the algorithm, the complexity of computation:

D (in all iterations) $- O(|S| \times |T|)$;

T_d (in all iterations) $- O(|S| \times |T|)$;

T_1 (in all iterations) $- O(|T| \times |T|)$.

Since $|T| \leq |S|$ in strongly connected 1-out nets, the upper bound of the time complexity is $O(|S| \times |T|)$.

(b) It is easy to see that the simple algorithm to construct $\phi(N)$ has the complexity $O(|S| \times |T|)$. Then the result follows from (a) and Lemma 5.1.2.

■

5.2 The Algorithm

Algorithm 5.2.1 *decide-well-formed (N)*

Input:

$N = (S, T, F)$ is a strongly connected EFC-net

Output is one of the following messages:

1: "N is well-formed "

2: "N is not well-formed "

Function: $get - S - component(N, s)$

This function finds an S-component of N containing place s. It is similar to the function mark-S-component [41], but gives (as the output) the subnet $(\overline{S}, \overline{T})$ of places of an S-component containing s. If it finds a minimal siphon not generating an S-component, the algorithm stops and gives a message "N is not well-formed".

Function: $get - TCP - subnet(N, s, S_o)$

This function finds a TCP-subnet of N containing place s. It finds also the set of the output places $\widehat{S}_o$ of the TCP-subnet. It is a version of the function get-S-component.

Function: $strongconnect(N)$

This function checks if the graph of N is strongly connected and is based on the depth-first search algorithm. This function is described in [94].

begin

if $strongconnect(N) = No$ **then Stop** with "N is not well-formed" **endif**

choose $s \in S$; $(\overline{S}, \overline{T}) := get - S - component(N, s)$; $S^i := {}^\bullet\overline{T} \backslash \overline{S}$;

while $(S^i \neq \emptyset)$ **do**

choose $s \in S^i$; $S^i := S^i \backslash \{s\}$; $(\widehat{S}, \widehat{T}, \widehat{S}_o) := get - TCP - subnet(N, s)$;

if $(|\widehat{S}_o| > 1)$ **then Stop** with "N is not well-formed" **endif**

endwhile

Stop with "N is well-formed";

end

Proposition 5.2.1 [57] The following problems have the complexity $O(|S| \times |T|)$:

(a) to decide well-formedness of EFC-nets;

(b) to decide whether a system (N, M) is regular.

Proof:

(a) The complexity of function "strongconnect" for a directed graph $G = (V, E)$ is $O(|V| + |E|)$ [94]. The complexity for function "mark-S-component" is given in [41]

($O(|T|)$). Since the set of the TCP-subnets (under consideration) defines a partition on the set of nodes of the net, the complexity of the finding and checking of the TCP-subnets is $O(|S| \times |T|)$. Hence the upper bound of the time complexity of the algorithm is $O(|S| \times |T|)$.

(b) It follows from Theorem 5.1.3b, Lemma 3.1.9a, Propositions 3.1.10 and 5.2.1a.

■

5.3 Summary

In [43, 40] $O(|S|^2 \times |T|)$-algorithm to decide if a given EFC- net is well-formed was given. This algorithm checks the conditions 2.5, 2.6, 2.7 (Theorem 3.1.6). To check the condition 2.5 (if a net has a positive S-invariant), this algorithm finds a cover of S-components. Checking of the condition 2.6 has been done analogously (finding a cover of S-component on reverse-dual net). Hence two main steps of the algorithm in [40] are: to find a cover of S-components and to compute the rank of the incidence matrix. Finding a cover of S-components has been done in [41] in $O(|S| \times |T|)$-time. Since the calculation of a matrix rank requires $O(|S|^2 \times |T|)$, the Rank Theorem can be checked in $O(|S|^2 \times |T|)$.

Our algorithm avoids the computation of rank. It results in a reduction by one order of magnitude, compared to the algorithm in [40]. Hence the absolute minimum of the complexity has been reached, because the input information on the structure of the net takes the capacity $O(|S| \times |T|)$.

The mapping γ was used to decide the liveness of an initial marking. But it proves to be useful for another goal. In [42] the net N itself is used to find a cover of S-components for the FC-net N. But the class of EFC-net is wider than the class of FC-nets. Therefore, the application of the method from [42] is not possible. To overcome this difficulty, in [41] the cluster graph is used as auxiliary data to find a cover of S-components. We propose the use of the mapping γ instead. Since 1-out

nets are a subclass of FC-nets, Lemma 3.2.3b allows us to compute a cover of minimal siphons using only one graph $\gamma(N)$ instead of two (N itself and cluster graph) in [41]. Next step is to check if every minimal siphon is an S-component. For that check we should use the net N itself, not $\gamma(N)$.

Chapter 6

Concurrency Relations

In general to find an efficient polynomial algorithm for the computation of the concurrency relation between nodes of a net, and to prove it for nontrivial classes of Petri nets is extremely hard task. The problem is EXPSPACE-hard for arbitrary systems, and PSPACE-complete for 1-bounded systems [14] (we use the terms Petri net and system interchangeably).

In [55] we proposed an $O(n^3)$ algorithm for the computation of the concurrency relation of extended Free-Choice Petri nets. Before that time the problem has been shown to be polynomial only for live Marked Graphs [63] and 1-bounded Conflict-Free systems [100, 26] (the algorithm of [26] can be easily generalized to the n-bounded case).

In [49] we generalized the algorithm for the computation of the concurrency relation to regular Petri nets. The time complexity of the algorithm is $O(n^4)$, where n is the number of nodes of the net.

The concurrency relation is usually defined as a set of pairs of transitions. We use a more general definition.

Let (N, M_0) be a system, and let X be the set of nodes of N. Given $x \in X$, define the marking M_x of N as follows:

- if x is a place, then M_x is the marking that puts one token on s, and no tokens elsewhere;
- if x is a transition, then M_x is the marking that puts one token on every input place of x, and no tokens elsewhere.

The *concurrency relation* $\| \subseteq X \times X$ contains the pairs (x_1, x_2) such that $M \geq M_{x_1} + M_{x_2}$ for some reachable marking M. In particular, two transitions t_1, t_2 belong to the concurrency relation if they can occur concurrently from some reachable marking, and two places belong to the concurrency relation if they are simultaneously marked at some reachable marking.

The concurrency relation is directly related to the *co* relation used in the theory of nonsequential processes [9]: (x_1, x_2) belongs to the concurrency relation if and only if some nonsequential process of (N, M_0) contains two elements in *co* labelled by x_1 and x_2. This is in fact a more elegant definition, but, since it requires the introduction of a number of concepts, we use the one above.

We now define the structural concurrency relation, first presented in [63]. Let (N, M_0) be a system, where $N = (S, T, F)$, and let $X = S \cup T$. The *structural concurrency relation* $\|^A \subseteq X \times X$ is the smallest symmetric relation such that:

(i) $\forall s, s' \in S$: $M_0 \geq M_s + M_{s'} \Rightarrow (s, s') \in \|^A$

(ii) $\forall t \in T$: $(t^\bullet \times t^\bullet) \setminus id_T \subseteq \|^A$

(iii) $\forall x \in X \; \forall t \in T$: $\{x\} \times {}^\bullet t \subseteq \|^A \Rightarrow (x, t) \in \|^A \wedge \{x\} \times t^\bullet \subseteq \|^A$

where id_T denotes the identity relation on T.

Loosely speaking, condition (i) states that any two places marked at the initial marking are structurally concurrent (actually, this is the case for a pair (s, s) only if M_0 puts at least two tokens on s). Clearly, this condition is fulfilled by the concurrency relation $\|$. Condition (ii) states that all the output places of a transition are structurally concurrent. This condition is fulfilled by the concurrency relation $\|$ only for

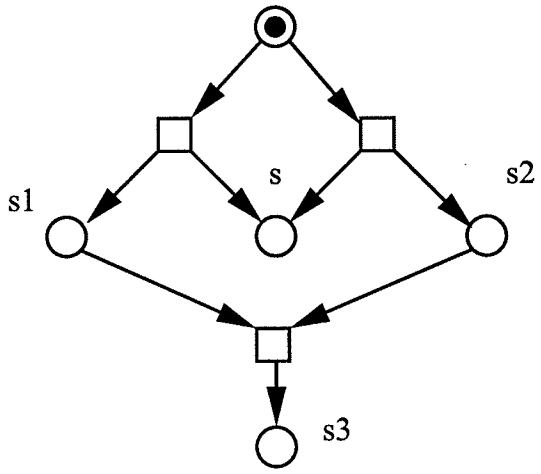

Fig. 6.1 A System for which Concurrency Relation and Structural Concurrency Relations are not Equal.

non-dead transitions. At first sight it could seem that $\|$ also fulfills condition (iii), but this is not the case. This condition states that if a node is structurally concurrent with all the input places of a transition, then it is also structurally concurrent with all its output places. Figure 6.1 shows a system in which $\|$ does not satisfy (iii): (s, s_1) and (s, s_2) are concurrent, because there are two *different* reachable markings which mark s, s_1 and s, s_2, respectively, but there is no reachable marking which puts tokens simultaneously on s, s_1 and s_2. So for this system we have $\| \neq \|^A$. Another example in which the system is live and 1-bounded can be found in [49].

We prove in this chapter that $\|^A$ and $\|$ coincide for regular systems. The proof has much in common with the proof of the Second Confluence Theorem [22], which we now recall.

6.1 The Second Confluence Theorem

The Second Confluence Theorem [22] states that if two live markings M_1 and M_2 of a regular system agree on all S-invariants, then they have a common successor, i.e., there exists a marking that is reachable from both M_1 and M_2. Since it can be easily

shown that any two reachable markings agree on all S-invariants, it follows that any two reachable markings have a common successor. The result can be generalized to a set $M_1, \ldots, M_n$ of markings which agree pairwise on all invariants, and in the sequel we consider this more general version.

Let us first recall the notions of markings that agree on all S-invariants.

Two markings M and M' of N *agree* on all S-invariants if $I \cdot M = I \cdot M'$ for every S-invariant I of N.

Theorem 6.1.1 [22] Let (N, M_0) be a system, and let M be a reachable marking. Then M and M_0 agree on all S-invariants.

■

The proof of the Second Confluence Theorem distinguishes two cases, according to whether the EFC-net N is a T-net or not. The first case is easily solved using the following result, which states that for T-systems the converse of Theorem 6.1.1 holds:

Theorem 6.1.2 [22] Let (N, M_0) be a live T-system. A marking M is reachable iff it agrees with M_0 on all S-invariants.

■

Since $M_1, \ldots, M_n$ are live and bounded and agree on all S-invariants, they are all reachable from each other. Therefore, any of them is a common successor of all the others.

In the second case, when N is not a T-net, the proof makes use of a reduction procedure given in the previous sections. N is split into two: a TCP-subnet $\widehat{N} = (\widehat{S}, \widehat{T})$, and the subnet $\overline{N}$ generated by all the nodes that do not belong to $\widehat{N}$.

Theorem 4.2.2 guarantees that N can be split.

Once N is split, we let n particular sequences occur from $M_1, \ldots, M_n$. These sequences contain only transitions of $\widehat{N}$ which are not way-in transitions.

Proposition 6.1.3 [22] Let N be an EFC-net, and let $M_1, \ldots, M_n$ be live and bounded markings of N that agree on all S-invariants. Let $\widehat{N}$ be a CP-subnet of N, let $\widehat{T}_i$ be the set of way-in transitions of $\widehat{N}$, and let $\overline{N} = N \setminus \widehat{N}$. There exist occurrence sequences $M_1 \xrightarrow{\sigma_1} M_1', \ldots, M_n \xrightarrow{\sigma_n} M_n'$, where $\sigma_1, \ldots, \sigma_n$ contain only transitions of $\widehat{T} \setminus \widehat{T}_{in}$, such that

(a) No transition of $\widehat{T} \setminus \widehat{T}_{in}$ is enabled at $M_1', \ldots, M_n'$,

(b) $\widehat{M_1'} = \cdots = \widehat{M_n'}$, where $\widehat{M}$ denotes the projection of M onto the places of $\widehat{N}$,

(c) $\overline{M_i} \leq \overline{M_i'}$ for $1 \leq i \leq n$, where $\overline{M}$ denotes the projection of M onto the places of $\overline{N}$, and

(d) $\overline{M_1'}, \ldots, \overline{M_n'}$ are live and bounded markings which agree on all S-invariants of $\overline{N}$.

■

Using Lemma 4.1.2, Proposition 6.1.3 can be easily generalized to regular systems. After the occurrence of these sequences we 'freeze' the transitions of the CP-subnet, i.e., we forbid them to occur again, and so preserve the equality $\widehat{M_1'} = \cdots = \widehat{M_n'}$. If $\overline{N}$ is a T-net, then Theorem 6.1.2 can be applied, and we are done. Otherwise, by Theorem 4.2.2 and Proposition 6.1.3c we can iterate the procedure until we get two markings which coincide everywhere, and are therefore the same. This marking is a common successor of $M_1, \ldots, M_n$. Instead of freezing the transitions of the CP-subnet, we can equivalently remove them and consider thereafter the remaining net $\overline{N}$.

6.2 The New Result

In order to adapt these results to the concurrency problem for regular systems, we take a closer look at the proof of Theorem 6.1.1. The proof is based on the notion of T-component.

It is easy to see that if a net is T-coverable then every node of the net, and not only every transition, belongs to some element of a cover.

Now, in order to find a TCP-subnet, we proceed as follows. We take a minimal cover C of N by T-components. Since N is not a T-net, we have $|\mathcal{C}| > 1$. We construct the (non-directed) graph $G = (V, E)$ as follows. V is the set $\mathcal{C}$ and E is the set of pairs (N_i, N_j) such that N_i and N_j have at least one common node. The graph G is connected because $\mathcal{C}$ is a cover of N and N is connected. Moreover, G has at least two nodes because $|\mathcal{C}| > 1$.

We choose a spanning tree of G, and select one of its leaves, say N_1. We then construct a maximal set of nodes X of N_1 satisfying the following properties: (a) the net generated by X is connected, and (b) no element of X belongs to a T-component of $\mathcal{C} \setminus \{N_1\}$. The set X is nonempty, because $\mathcal{C}$ is a minimal cover. The subnet N_x generated by X is a TCP-subnet.

We prove preliminary results, and then our main theorem.

Lemma 6.2.1 [18] Let (N, M^0) be a live T-net.

(a) Let M be a marking of N. Then there exists a marking $M' \in [M^0\rangle$ such that $M' \geq M$ iff $M^0(S') \geq M(S')$ for every directed circuit $N' = (S', T', F')$ of N.

(b) Every directed circuit is marked at M^0.

(c) M^0 and M are reachable from each other iff $M^0(S') = M(S')$ for every directed circuit $N' = (S', T', F')$ of N.

■

Lemma 6.2.2 [74] Let M, M^0 be markings of a T-net $N = (S, T, F)$ and the system (N, M^0) be live. Then there exists a marking $M^1 \in [M^0\rangle$ such that $M^1 \geq M$ iff $M^0(S_1) \geq M(S_1)$ for every elementary directed circuit $N_1 = (S_1, T_1, F_1)$ of N.

Proof:

Each directed circuit of a T-net can be represented as an union of elementary circuits. Then apply Lemma 6.2.1.

■

Theorem 6.2.3 [74]

(a) For every net system, $\| \subseteq \|^A$.

(b) For every live T-system, $\| = \|^A$.

Proof:

(a) It is sufficient to prove $\forall M \in [M^0\rangle \forall s_1, s_2 \in S$, if $M \geq \bar{s}_1 + \bar{s}_2$ then $(s_1, s_2) \in \|^A$. Let $M^0[\sigma\rangle M$. We prove by induction on $\mid \sigma \mid$.

Basis. ($\mid \sigma \mid = 0$) follows from (i).

Step. ($\mid \sigma \mid > 0$). Let $\sigma = \sigma' t$ and $M^0[\sigma'\rangle M'[t\rangle M$. By the induction hypothesis $\forall s_1, s_2 \in S$ such that $M' \geq \bar{s}_1 + \bar{s}_2$, we have $(s_1, s_2) \in \|^A$. We need to prove this for M. Let $s_1, s_2 \in S$ such that $M \geq \bar{s}_1 + \bar{s}_2$. There exist three cases:

(1) $\{s_1, s_2\} \cap (t^\bullet \backslash {}^\bullet t) = \emptyset$. Then $M' \geq \bar{s}_1 + \bar{s}_2$ and by induction hypothesis $(s_1, s_2) \in \|^A$. It is also true, if $s_1 = s_2$.

(2) $s_1, s_2 \in t^\bullet \wedge s_1 \neq s_2$. Then by (ii) $(s_1, s_2) \in \|^A$.

(3) $s_1 \in t^\bullet \backslash {}^\bullet t \wedge (s_1 = s_2 \vee s_2 \notin t^\bullet \backslash {}^\bullet t)$. Then $M' \geq \bar{s}_2 +^\bullet \bar{t}$ and by induction hypothesis we have $\forall s \in {}^\bullet t$ $(s, s_2) \in \|^A$. Then by (iii) we have $(s_1, s_2) \in \|^A$.

(b) Inclusion $\|^A \subseteq \|^A$ is Theorem 6.2.3(a). We prove now $\|^A \supseteq \|^A$. The condition (i) of the $\|^A$ definition is trivial. (ii) follows from liveness of (N, M^0). We now prove (iii).

Assume $s \in S, t \in T$ and $\forall s_1 \in {}^\bullet t \quad (s, s_1) \in \|$. We take a marking $M =^\bullet = {}^\bullet \bar{t} + \bar{s}$. It is obvious that $|{}^\bullet t \cap S_1| \leq 1$ for every elementary directed circuit $N_1 = (S_1, T_1, F_1)$. Hence for every elementary circuit N_1 containing places s and $s_1 \in^\bullet t$ (s and s_1 can

coincide) we have $M(S_1) = 2$. Since $\forall s_1 \in {}^\bullet t \quad (s, s_1) \in \|$, we have $M^0(S_1) \geq 2$ for the same circuits. For every other elementary circuit N_1 we have $M(S_1) \leq 1$, and since (N, M^0) is live, we have $M^0(S_1) \geq 1$. Hence $M^0(S_1) \geq M(S_1)$ for every elementary circuit N_1 of (N, M^0). From Lemma 6.2.2 it follows that $\exists M^1 \in [M^0\rangle$ such that $M^1 \geq M$. It is obvious that $M[t >$ and hence $M^1[t\rangle$. Assume $M^1[t\rangle M^2$. Then $M^2 \geq \bar{t}^\bullet + \bar{s}$. Hence $\forall s_2 \in t^\bullet \quad (s, s_2) \in \|$.

■

Proposition 6.2.4 Let (N, M_0) be a regular system, and let s and t be a place and a transition of N such that ${}^\bullet t = \{r_1, \ldots, r_n\}$. Assume that for every $1 \leq i \leq n$ there exists a reachable marking M_i such that $M_i \geq M_s + M_{r_i}$. Then there exists a reachable marking $M \geq M_s + \sum_{i=1}^{n} M_{r_i}$.

Proof:

Let $\mathcal{C}$ be a minimal T-cover of N, which exists by Lemma 3.5.5. We consider two cases:

(a) Some T-component N_1 of $\mathcal{C}$ contains both s and t.

Let G be the graph described above, and let G_s be a spanning tree of G. We construct systems $(N_1, M_1^f), \ldots, (N_1, M_n^f)$ as follows. If G_s has only one node, then $N = N_1$, and we take $(N_1, M_i^f) = (N_0, M_i)$. If G_s has more than one node, we select one of its leaves, different from N_1 (this is possible, because a spanning tree with at least two nodes has at least two leaves, and so we are never forced to select N_1). Once such a leaf N_i is chosen, we consider its TCP-subnets one by one. For each TCP-subnet, we execute the occurrence sequences of Proposition 6.1.3 from the markings $M_1, \ldots, M_n$. After that, we remove the TCP-subnet. We proceed like this with all the TCP-subnets of N_i. We thus obtain systems $(N', M_1'), \ldots, (N', M_n')$, where N' is minimally covered by $\mathcal{C}' = \mathcal{C} \setminus \{N_i\}$. Moreover, the graph G' corresponding to the minimal cover

$\mathcal{C}'$ is the graph obtained from G by removing the node N', and the graph G'_s obtained from G_s by removing the node N' is a spanning tree of G'. If G'_s contains more than one node, we iterate the procedure, this time starting from $(N', M'_1), \ldots, (N', M'_n)$ and G'_s.

Since each iteration removes one node from G, the procedure terminates when the spanning tree contains only one node. Since N_1 is never removed, this node is n_1. So the procedure outputs systems $(N_1, M_1^f), \ldots, (N_1, M_n^f)$.

Let $M_{11}, \ldots, M_{1n}$ be the projection of $M_1, \ldots, M_n$ onto the places of N_1. Since Proposition 6.1.3b can be applied each time we remove a TCP-subnet, we have:

(i) $M_i^f \geq M_{1i}$ for $1 \leq i \leq n$, and

(ii) $M_1^f \ldots, M_n^f$ agree on all the invariants of N_1.

By (i), $M_i^f \geq M_s + M_{r_i}$. The result follows from (ii) and Theorem 6.2.3b.

(b) No T-component of $\mathcal{C}$ contains both s and t.

Let N_1 be a T-component of $\mathcal{C}$ containing s. We choose a spanning tree G_s of G, and proceed as in (a), iteratively selecting a leaf different from N_1. However, we no longer stop when the spanning tree contains a node, but as soon as t belongs to a TCP-subnet $\widehat{N}$ of some leaf. Notice that this eventually happens, because otherwise the reduction process could continue until only N_1 remains, which contradicts the assumption that no T-component of $\mathcal{C}$ contains both s and t.

Let N' be the net obtained after termination, and let $M_1^f, \ldots, M_n^f$ be the corresponding markings. Further, let $M'_1, \ldots, M'_n$ be the projection of $M_1, \ldots, M_n$ onto the places of N'. By Proposition 6.1.3c, $M_i^f \geq M'_i$, and therefore M_i^f marks both s and r_i. Now, by Proposition 6.1.3, there exist occurrence sequences $\sigma_1, \ldots, \sigma_n$ enabled at $M_1^f, \ldots, M_n^f$, which contain only transitions of $\widehat{T} \setminus \widehat{T}_{in}$, and lead to markings satisfying two conditions:

(i) the projections of $M_1^f, \ldots, M_n^f$ onto the places of $\widehat{N}$ coincide, and

(ii) no transition of $\widehat{T} \setminus \widehat{T}_{in}$ is enabled at $M_1^f, \ldots, M_n^f$.

Since $\widehat{N}$ is a T-net, (i) and (ii) can only hold if all of $\sigma_1, \ldots, \sigma_n$ contain the transition t. Since s remains marked along the execution of these sequences, some reachable marking marks simultaneously s and all the input places of t.

■

Proposition 6.2.4 leads to our main result:

Theorem 6.2.5 *Concurrency Theorem for regular systems*

The relations $\|$ and $\|^A$ coincide for regular systems.

Proof:

We have $\| \subseteq \|^A$ by Theorem 6.2.3a. We prove that the $\|$ relation of a regular system (N, M_0) satisfies the three conditions of the definition of the structural concurrency relation. Since $\|^A$ is the smallest symmetric relation satisfying these conditions, we have $\|^A \subseteq \|$, which finishes the proof. Condition (i) follows easily from the definition of $\|$. Condition (ii) is a direct consequence of the liveness of (N, M_0). Condition (iii) follows immediately from Proposition 6.2.4.

■

6.3 Computing the Structural Concurrency Relation

In [63], we present a $O(n^5)$ algorithm for the computation of $\|^A$ in an arbitrary system, where n is the number of places and transitions of the net. In [55] we have shown that $\|^A$ can be computed in $O(n^4)$ time, and in $O(n^3)$ time for free-choice systems.

For self-containment of the article we repeat the algorithm given in [55] for ordinary nets.

Algorithm 6.3.1

Input: A system (N, M_0), where $N = (S, T, F)$.

Output: $R \subseteq X \times X$.

begin

$R := \{(s, s') \mid M_0 \geq M_s + M_{s'}\} \cup \bigcup_{t \in T} t^\bullet \times t^\bullet$;

$E := R \cap (X \times S)$;

while $E \neq \emptyset$ **do**

choose $(x, s) \in E$; $E := E \setminus \{(x, s)\}$;

for every $t \in s^\bullet$ **do**

if $\{x\} \times {}^\bullet t \subseteq R$ **then**

$E := E \cup ((\{x\} \times t^\bullet) \setminus R)$;

$R := R \cup \{(x, t)\} \cup (\{x\} \times t^\bullet)$;

endif

endfor

endwhile

end

Proposition 6.3.2

Algorithm 6.3.1 terminates, and after termination $R = \|^A$.

Proof:

Observe that $E \subseteq R$ is an invariant of the while loop and holds initially. Therefore, each execution of the while loop removes from E an element of $E \cap R$. This element is never added to E again. So the algorithm terminates.

Let Q be the value of R after termination. We prove:

(1) $Q \subseteq \|^A$.

We have $\{(s, s') \mid M_0 \geq M_s + M_{s'}\} \subseteq \|^A$ and $\bigcup_{t \in T} t^\bullet \times t^\bullet \subseteq \|^A$ by definition.

So initially $R \subseteq \|^A$.

Moreover, it follows easily from the definition of $\|^A$ that $R \subseteq \|^A$ is an invariant of the while loop. So we have $Q \subseteq \|^A$.

(2) Q satisfies the three conditions of the definition of the structural concurrency relation.

Conditions (i) and (ii) follow immediately from the initialization of R. For condition (iii), let $x \in X$ and $t \in T$. We have to prove:

$$\{x\} \times {}^\bullet t \subseteq Q \implies (x, t) \in \|^A \wedge \{x\} \times t^\bullet \subseteq Q$$

If $\{x\} \times {}^\bullet t$ is not included in Q, we are done. So assume $\{x\} \times {}^\bullet t \subseteq Q$.

Let (x, s) be the last element of $\{x\} \times {}^\bullet t$ which is removed from E during the execution of the algorithm. As we have seen above, (x, s) is never added to E again.

Assume that immediately after (x, s) is removed from E, we have $(x, s') \notin R$ for some $s' \in {}^\bullet t$. We prove that (x, s') is never added to R later on. Every new element added to R is also added to E, and every element of E is removed before termination. Therefore, if (x, s') were added to R it would later be removed from E, which contradicts our assumption about (x, s).

Since $\{x\} \times {}^\bullet t \subseteq R$ and no element of $\{x\} \times {}^\bullet t$ is added to R after (x, s) is removed from E, we already have $\{x\} \times {}^\bullet t \subseteq R$ immediately after (x, s) is removed from E. Then, the next execution of the for loop adds $(\{x\} \times t^\bullet)$ to Q. So $(\{x\} \times t^\bullet) \subseteq Q$ after termination.

$Q = \|^A$ follows from (1), (2) and the minimality of $\|^A$.

■

We calculate the complexity of the algorithm when the subsets $X \times X$ (in particular, the incidence relations of the net) are encoded as bidimensional arrays $X \times X \to \{0,1\}$. In this case, the algorithm needs $O(|X|^2)$ space.

The initialization of Q, E and N takes $O(|S|^2 \cdot |T|)$ time. The while loop is executed at most $O(|S| \cdot |X|)$ times, because each iteration removes one element from E which is never added to it again. In each iteration the execution of the for loop takes $O(|S| \cdot |T|)$ time ($O(|T|)$ iterations with $O(|S|)$ time each). So the algorithm runs in $O(|S|^2 \cdot |T| \cdot |X|)$ time.

6.4 Summary

We have presented an $O(n^4)$ algorithm for the computation of the concurrency relation of regular systems, where n is the number of nodes of the net.

The work adds one more to the list of results on the concurrency problem, i.e., the problem of deciding if two given transitions are concurrently enabled at some reachable marking. The problem is EXPSPACE-hard for arbitrary systems, and PSPACE-complete for 1-bounded systems [14]. It has been shown to be polynomial for live T-systems [63], 1-bounded conflict-free systems [100, 26] (the algorithm of [26] can be easily generalized to the n-bounded case), live and bounded extended free-choice systems [55], and in [49] for regular systems.

Our algorithm also can be used to solve the 1-boundedness problem: that is, deciding if a given regular system is 1-bounded. It follows easily from the definition of the concurrency relation that a system is 1-safe iff its concurrency relation is irreflexive. So the 1-boundedness problem can be solved in $O(n^4)$ as well. This improves the complexity of earlier algorithms based on linear programming.

Chapter 7

The Method of Unfoldings for the Analysis of Persistent Nets

The method of unfoldings is used for the verification of net properties because of the following obvious idea. The unfolded system belongs to a more restricted class of systems than the original one. And for this restricted class a lot of analysis problems can be decided easier than for a wider class. Hence we reduce the complexity of analysis.

Persistence is one of important behavior properties of Petri nets and other models of concurrency and has been intensively studied by several authors [39, 76]. Persistence means that for any two fireable transitions, the firing of one of them does not disable the other. In this chapter some new properties of persistent systems are proven.

Conflict-free systems are a proper subclass of persistent ones. Conflict-freeness is a structural property. Persistence of conflict-free systems is based on their structure.

To unfold a system we use partial order semantics (POS) [6, 5]. POS represent concurrency in a "true" way and facilitates the proofs of some results [6, 5]. It is especially useful for analysis of persistent systems, because they have no behavior conflicts. We say that two systems are behavior equivalent if they have the same processes.

The method of unfoldings for the class of live safe persistent systems (LSP-systems for short) by labelled live and safe marked graphs was considered by a number of authors [88, 87]. However in these papers only intuitive discussion has been given and no propositions about behavior equivalence have been proved. In [11] it was proposed to split conflict places of a bounded persistent system (BP-system) to obtain a bounded marked graph. However, some problems remained outside of these discussions.

The goal of this chapter is to solve these problems, to provide a method of unfoldings for ordinary Petri nets by conflict-free ones and to prove a theorem about behavior equivalence of two classes of safe Petri nets: persistent systems and labelled conflict-free systems, which are the result of unfoldings of persistent systems. The method of unfoldings and the behavior equivalence theorem are used for the verification of whether the given system is safe and persistent.

7.1 Definitions and Preliminaries

We recall some notions. $L(N, M^0) = \{\sigma | M^0[\sigma\rangle\}$ is said to be the *language* of (N, M^0); $L^\infty(N, M^0) = L(N, M^0) \cup \{\sigma | M^0[\sigma\rangle$ and σ is infinite$\}$; $[M^0\rangle = \{M | \exists \sigma \in L(M^0) M^0[\sigma\rangle M\}$ (the *reachability set* of (N, M^0)). If σ is a sequence of symbols and u is a symbol, then $|\sigma|_u$ denotes the number of times u appears in σ. If N is fixed we will write $L(M)$ instead of $L(N, M)$.

In [17] 5 levels of transition liveness are given:

$0) T^0(M) = \{t \in T | \forall \sigma \in L(M) | \sigma|_t = 0\}$,

$1) T^1(M) = \{t \in T | \exists \sigma \in L(M) | \sigma|_t > 0\}$,

$2) T^2(M) = \{t \in T | \forall k \in \mathbb{N} \exists \sigma \in L(M) | \sigma|_t) \geq k\}$,

$3) T^3(M) = \{t \in T | \exists \sigma \in L^\infty(M) \forall k \in \mathbb{N} | \sigma|_t > k\}$,

$4) T^4(M) = \{t \in T | \forall M^1 \in [M\rangle t \in T^1(M^1)\}$.

A transition t in a system (N, M^0) is said to be *Lk-live in* $M \in [M^0\rangle$ iff $t \in T^k(M), k =$

$0, 1, 2, 3, 4$. A system (N, M^0) is said to be *Lk-live* iff every transition in the system is Lk-live, $k = 0, 1, 2, 3, 4$.

Let $(t, t') \in indep \subseteq T \times T$ iff $({}^\bullet t \cup t^\bullet) \cap ({}^\bullet t' \cup t'^\bullet) = \emptyset$.

The following lemma easily follows from the definitions.

Lemma 7.1.1 Let (N, M^0) be a system. Then:

(a) $\forall M \in [M^0\rangle$

$T^4(M^0) \subseteq T^3(M^0) \subseteq T^2(M^0) \subseteq T^1(M^0)$;

$T^4(M) \subseteq T^3(M) \subseteq T^2(M) \subseteq T^1(M)$;

$T^1(M) \subseteq T^1(M^0)$;

$T^2(M) \subseteq T^2(M^0)$;

$T^3(M) \subseteq T^3(M^0)$;

$T^4(M^0) \subseteq T^4(M)$;

(b) let (N, M^0) be safe, $\forall t \neq t' \in T\ \forall M \in [M^0\rangle if (M[tt'\rangle \wedge M[t't\rangle \wedge (t, t') \notin indep)$ then ${}^\bullet t \cap^\bullet t' = t^\bullet \cap t'^\bullet \neq \emptyset$.

■

A transition is said to be *strictly Lk-live in M* iff it is Lk-live in M, but not $L(k+1)$-live in $M, k = 1, 2, 3$.

A *labelled* system $(N, M^0)^l = ((N_1, M_1^0), X', l)$ consists of a system (N_1, M_1^0), an alphabet X' and labelling homomorphism $l : (S_1 \cup T_1)^* \to X'^*$ such that $l(S_1) \cap l(T_1) = \emptyset$.

For a given firing sequence σ $Pk : L(N, M^0) \to \mathbb{N}^{|T|}$ is the *Parikh map* defined by:

$Pk(\sigma) := [|\sigma|_{t_1}, \ldots, |\sigma|_{t_T}]^T$

Let us introduce some denotations: $\Delta M = M' - M$, where $M[\sigma\rangle M'$; $Ln(M) = \{\sigma \in L(M) | \Delta(\sigma) \geq 0\}$ is the set of *repeatable* firing sequences at M; $Lr(M) = \{\sigma \in L(M) | \Delta(\sigma) = 0\}$ is the set of *reproduction* firing sequences at M; $Pn(M) =$

$\{Pk(\sigma)|\sigma \in Ln(M)\}; Pn = \cup_{M\in[M^0\rangle} Pn(M)$. Let $V \subseteq \mathbb{N}^k, k \in \mathbb{N}$, then we denote $V' = \min(V)$ iff:

$(1)V' \subseteq V$,

$(2)\forall v, v' \in V'[v \neq v'] \rightarrow \neg[v \leq v']$,

$(3)\forall v \in V \exists v' \in V' v' \leq v$;

mPn= $\min(Pn)$, mPr= $\{v \in \text{mPn}|\Delta(\sigma) = 0\}$ is the set of minimal *reproduction* Parikh vectors.

A *process* $\pi = (N, p) = (B, E, K, p)$ of a system (S, T, F, M^0) consists of an occurrence net $N = (B, E, K)$ together with a labelling $p : B \cup E \Leftrightarrow S \cup T$ which satisfy the properties such that π can be interpreted as a concurrent run of (N, M^0). We distinguish *initial* processes which start with the initial marking M^0 and general processes which may start at any successor marking of M^0. A process which corresponds to a reproduction firing sequence σ with $Pk(\sigma) \in$ Pr is said to be a *reproduction* process. A reproduction process π which corresponds to a firing sequence σ with $Pk(\sigma) \in mPr$ is said to be a *minimal* reproduction process. $\Pi(N, M^0)$ denotes the set of all initial processes of (N, M^0).

A system (N, M^0) and a labelled system $(N^l, M^{0l}) = ((N_1, M_1^0), S \cup T, l)$ are *behavior equivalent* iff $\Pi(N, M^0) = \Pi(N^l, M^{0l})$.

A system (S, T, F, M^0) is said to be *persistent* (*P-system* for short) iff $\forall t_1 \neq t_2 \in T$ $\forall M \in [M^0\rangle\ M[t_1\rangle \wedge M[t_2\rangle \rightarrow M[t_1 t_2\rangle$. In Fig. 7.1 a live safe persistent system (LSP-system for short) is shown with its initial reproduction process is shown in Fig. 7.2.

For a given process π the Parikh map $Pk : \Pi(N, M^0) \rightarrow \mathbb{N}^{|T|}$ is defined analogously.

The following notions can be found in [76].

$EPk : T^* \rightarrow \mathbb{N}^{|T|+|S|}$ is the *extended Parikh map* defined by: $EPk(\sigma) = Pk(\sigma) \cdot \Delta(\sigma)$;

$Pf = \{Pk(\sigma)|EPk(\sigma)\ \in \cup_{M\in[M^0\rangle} \min(\{EPk(\sigma)|\sigma \in Ln(M)\})\}$.

Define an operation $\div$ on T^*.

$\sigma \div \lambda := \sigma$;

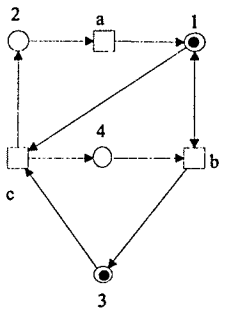

Fig. 7.1 LSP-system.

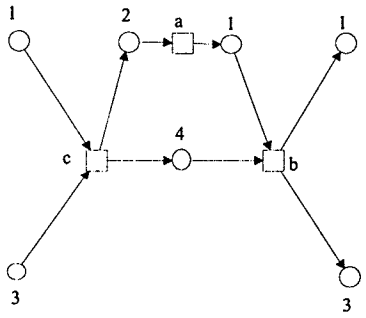

Fig. 7.2 Reproduction Process.

$\sigma \div t := \sigma$ if $|\sigma|_t = 0$, otherwise delete leftmost occurrence of t in σ;

$\sigma \div (\tau t) := (\sigma \div \tau) \div t$ $(\sigma, \tau \in T^*, t \in T)$.

Lemma 7.1.2 gives some properties of P-systems. Specifically:

(a) if some repeatable firing sequence can fire from the given marking M then for every marking M' reachable from M some other firing sequence, whose Parikh mapping is the same, can fire from M';

(b) a maximum of every two repeatable sequences is also repeatable;

(c) if one of two repeatable sequences is a reproduction one then the intersection of their supports is the empty set;

(d) there does not exist any strictly L2- and L3-live transitions in a P-system, and the set $T^4(M)$ is an invariant at any evolution of the given system and equals the union of all supports of firing sequence Parikh maps from the sets Pf(Pn).

These properties do not hold for arbitrary systems.

Lemma 7.1.2 Let (N, M^0) be a P-system. Then:

(a) $\forall M \in [M^0\rangle \forall \sigma \in Ln(M)\ \forall M^1 \in [M\rangle \exists \sigma_1 \in L(M^1) Pk(\sigma) = Pk(\sigma_1)$,

(b) $\forall \sigma_1, \sigma_2 \in Ln(M)\ \sigma_1(\sigma_2 \div \sigma_1) \in Ln(M)$,

(c) $\forall v_1 \neq v_2 \in$mPn if $v_1 \in$mPr or $v_2 \in$mPr then $\langle v_1\rangle \cap \langle v_2\rangle = \emptyset$,

(d) $\forall M \in [M^0\rangle \quad T^4(M^0) = T^3(M^0) = T^2(M^0) = T^4(M) = T^3(M) = T^2(M) = \cup_{v \in Pf}\langle v\rangle = \cup_{v \in Pn}\langle v\rangle$,

(e) if (N, M^0) is bounded then $\{\langle v\rangle \mid v \in$mPn$\}$ is a partition on the set T^4.

Proof:

The proof is given in the appendix for this chapter.

■

The property of Lemma 7.1.2e does not hold if we remove persistence or boundedness.

A system (N, M^0) is said to be *reversible* iff $\forall M \in [M^0\rangle M^0 \in [M\rangle$.

7.2 Motivation

The method of unfoldings from [88] is an intuitive application of partial order semantics [6]. An initial process π is called in [88] a behavior graph and is built in an obvious way: take the initial marking as $\pi_{\min}$, choose a fireable transition t, add t to π together with places $t^\bullet$ and all the arcs incident to t, take a new marking $M := (M \backslash {}^\bullet t) \cup t^\bullet$ and so on. In Fig. 7.1 an LSP-system is shown. In Fig. 7.2 the corresponding process. To get a marked graph from the process we propose to find two different B-cuts $c < c'$ with the same labels ($M_c = M_{c'}$). Such cuts always exist for any LSP-system. Then we need to take the fragment of the process $[c, c']$, merge every pair of places ($s \in c, s' \in c'$) such that $l(s) = l(s')$ (we will call *merge* c and c'), and add one token to each merged place to get a labelled live and safe marked graph (L means here L4-liveness). The result of such unfolding for an LSP-system (Fig. 7.1) is shown in Fig. 7.3.

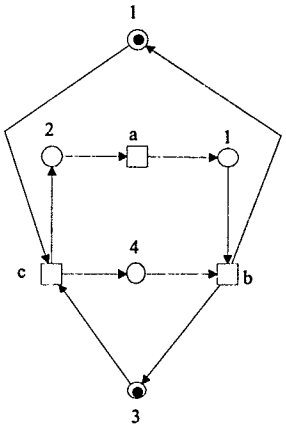

Fig. 7.3 Unfolding.

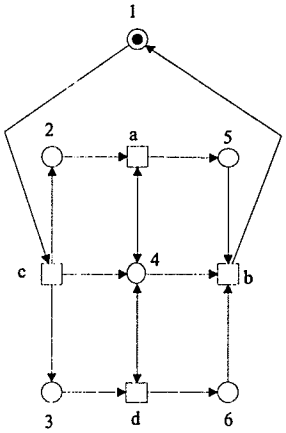

Fig. 7.4 Labelled safe conflict-free system.

It is easy to see the problems of such an approach. The first problem appears when a given initial marking is not reproducible (i.e. a system is not reversible). No one marked graph constructed by the method of [88] has the same language as the original system because no initial process is a reproduction one.

To solve this problem we propose to build a labelled conflict-free system instead of a marked graph. To do so we propose to take the prefix of the process $[\pi_{\min}, c']$ instead of $[c, c']$, merge the cuts c and c' as before and then add one token to each place $s \in \pi_{\min}$ to get a labelled safe conflict-free system (see Fig. 7.4).

The second problem concerns so called fictitious conflict. We will say that transitions t and t' are in a *fictitious* conflict iff ${}^\bullet t \cap {}^\bullet t' \neq \emptyset, M[tt'\rangle, M[t't\rangle$. The conflict is called fictitious because the conflict is structural (${}^\bullet t \cap {}^\bullet t' \neq \emptyset$) but not behavioral. In the

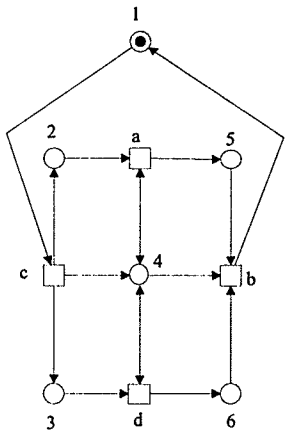

Fig. 7.5 Live safe persistent system.

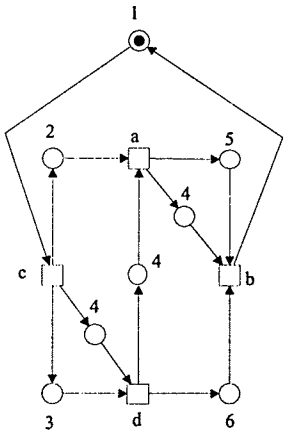

Fig. 7.6 Unfolding.

LSP-system (N, M^0) of Fig. 7.5 transitions a and d are in conflict (${}^\bullet b \cap {}^\bullet c \neq \emptyset$) but $\exists M = \{2,3,5\} \in [M^0\rangle M[bc\rangle$ and $M[cb\rangle$. The result of the method of unfoldings [88] (Fig. 7.6) is not behavior equivalent to the original system.

To handle this problem it is proposed to split the places of fictitious conflict before unfolding. This is done in Fig. 7.7 (the place 3 of the system in Fig. 7.5 is split to 3 and 7).

It is easy to see that the transformation preserves the language of a system and a lot of the behavioral properties such as liveness, boundedness, safeness, persistence and so on. Particularly, if a place s of a fictitious conflict such that ${}^\bullet s = s^\bullet$ and $M^0(s) = 1$ then s can be deleted by the deleting of a place-loop reduction rule [64]. But in the case of Fig. 7.7 removing of place 5 can lead to violation of the condition

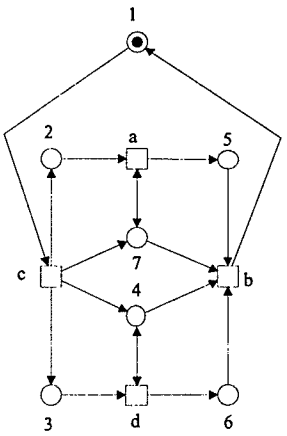

Fig. 7.7 Unfolding.

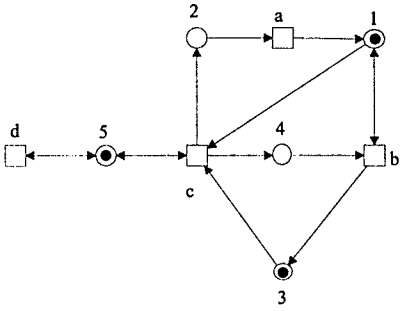

Fig. 7.8 LSP-system.

$T \subseteq \mathrm{dom(F)} \cap \mathrm{cod(F)}$. In this case it is recommended to leave the place for those transitions, which are candidates for the violation of the condition $T \subseteq \mathrm{dom(F)} \cap \mathrm{cod(F)}$. The result of splitting for the system, shown in Fig. 7.8, consists of the systems of Fig. 7.1 and the trivial system consisting of the transition d and the place 5, connected with two opposite arcs.

If the result of splitting is not a connected system then we can unfold and analyze the parts in isolation.

The third problem appears when we have several initial reproduction processes (systems of Fig. 7.8, 7.9). The system of Fig. 7.8 is an LSP-system. But after the splitting of the fictitious conflict places, the system is transformed into two isolated LSP-systems with the single minimal reproduction processes. As shown in Theorem 7.2.1, every LSP-system without fictitious conflicts has one and only one minimal reproduction process.

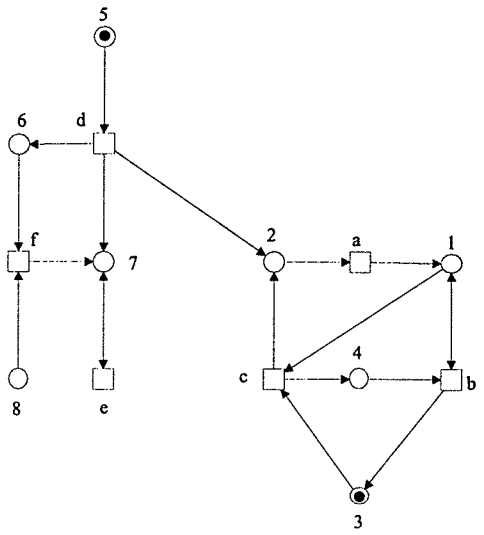

Fig. 7.9 SP-system.

Two transitions $t_1 \neq t_2$ are in the *fair* relation iff $\exists n \in \mathbb{N} \forall M \in [M^0\rangle \forall \sigma \in L(M) \cap (T \backslash \{t_1\})^* | \sigma |_{t_2} \leq$n. A system (N, M^0) is said to be fair iff $\forall t_1 \neq t_2 \in T$ t_1 and t_2 are in the fair relation.

Theorem 7.2.1 Let (N, M^0) be an SP-system having no fictitious conflicts. Then:

(a) (N, M^0) is L4-live iff $mPr = \{v\}$ and $\langle v\rangle = T$;

(b) if (N, M^0) is L4-live then (N, M^0) is reversible iff $\exists \sigma \in Ln(M^0) \backslash \{\lambda\}$;

(c) Two transitions $t_1 \neq t_2$ are in the fair relation iff they are both L1-live or there exists a $v \in mPr$ such that $\{t_1, t_2\} \subseteq \langle v\rangle$;

(d) If (N, M^0) is L4-live then it is fair.

Proof:

The proof is given in appendix of this chapter.

■

In this work we do not want to restrict ourselves to LSP-systems. We want to prove the result of behavior equivalence for widest possible class of systems. The system of Fig.7.9 is an SP-system having no fictitious conflict and having two minimal reproduction processes.

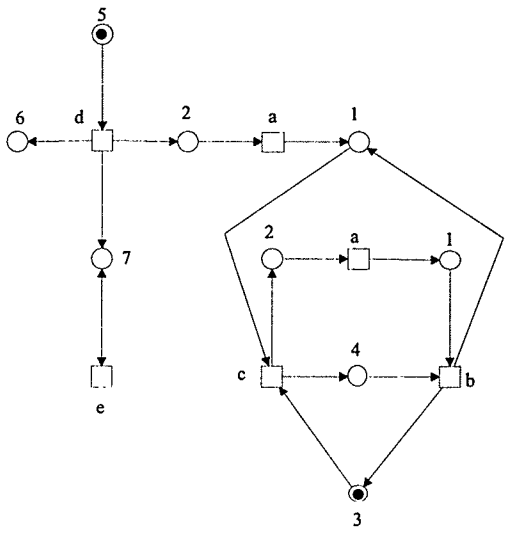

Fig. 7.10 Unfolding.

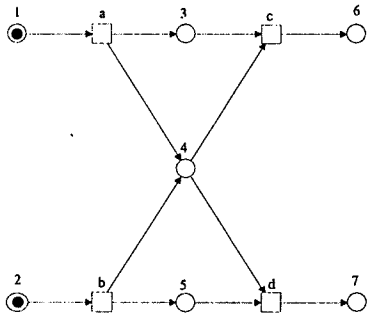

Fig. 7.11 BP-system.

To solve the third problem we propose to construct systems corresponding to minimal reproduction processes each and then unite them (see Fig. 7.10).

Now we consider the drawbacks of other approaches. In [11] it is proposed to split conflict places of a BP-system to get a bounded marked graph. Again when the initial marking is not reproducible, the marked graph obtained is not behavior equivalent to the original system. The second problem of splitting is illustrated by Fig. 7.11, 7.12, 7.13.

There are several possible outcomes from splitting. For example, the unfoldings obtained (Fig. 7.12, 7.13) are not behavior equivalent. From the example it can be seen that for non-safe systems the order of transitions, used for building the process, affects the result of unfoldings, and we cannot always obtain a behavior equivalent

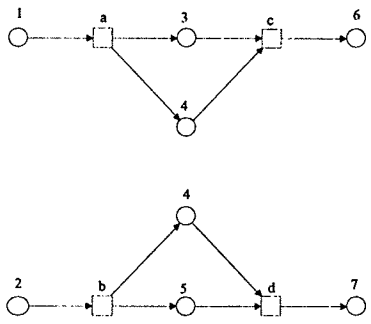

Fig. 7.12 Splitting Result.

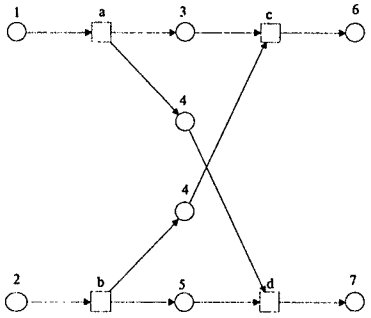

Fig. 7.13 Splitting Result.

system. Hence we restrict ourselves to SP-systems.

7.3 The Algorithm of Unfoldings and a Theorem on Behaviour Equivalence

So we can see that for a system with formal conflicts we can not obtain a behavior equivalent conflict-free system. Hence we will split fictitious conflicts before unfoldings and build behavior equivalent conflict-free systems for such persistent systems without fictitious conflicts.

Let us define the operation of fictitious conflict splitting formally:

$\forall s_i \in S$ such that $|{}^\bullet s_i \cap s_i^\bullet| \geq 2$ denote $T_i := {}^\bullet s_i \cap s_i^\bullet$;

$\forall t \in T_i$ create a new place s_i^t,

$S := S \cup \{s_i^t\}$,

$F := F \cup \{(t, s_i^t), (s_i^t, t)\}$;

$\forall s_i^t\ \forall t_1 \in {}^\bullet s_i \backslash T_i\ F := F \cup \{(t_1, s_i^t)\}$,

$\forall t_1 \in s_i^\bullet \backslash T_i\ F := F \cup \{(s_i^t, t_1)\}$;

remove s_i.

If it is not confusing, sometimes we will consider a marking M as a set of marked places if $\forall s \in S\ M(s) \leq 1$.

The algorithm.

input system (S, T, F, M^0) having no fictitious conflicts

output (N^l, M^{0l})

Program:

begin

if $\exists s \in S\ M^0(s) > 1$ *then Stop with message* "(N, M^0) is not safe, (N^l, M^{0l}) is empty";

$(N^l, M^{0l}) = ((N_1, M_1^0), S \cup T, l)$, where $(N_1, M_1^0) := (S_1, T_1 := \emptyset, F_1 := \emptyset, M_1^0 := S_1)$,

S_1 is such that $|S_1| = |\{s \in S | M^0(s) = 1\}|$,

$l : S_1 \to S$ is such that $\forall s \neq s' \in S_1\ l(s) \neq l(s')$;

$T' := T; M := M^0$;

Step 1. $\pi := (B, E := \emptyset, K := \emptyset, p)$, where B is such that $|B| = |\{s \in S | M(s) = 1\}|$,

$p : B \to S$ is such that $\forall s \neq s' \in B\ p(s) \neq p(s')$;

while $\exists t \in T'$ is fireable in (S, T, F, M);

begin

choose t, fireable in M

if $((M \backslash {}^\bullet t)) \cap t^\bullet \neq \emptyset$

then Stop with message "(N, M^0) is not safe, (N^l, M^{0l}) is empty"

add t and $t^\bullet$ to π; $M := (M \backslash {}^\bullet t) \cup t^\bullet$;

if $\exists$ two B-cuts $c < c'$ of π such that $M_c < M_{c'}$

then Stop with message "(N, M^0) is not safe, (N^l, M^{0l}) is empty"

if $\exists$ two B-cuts $c < c'$ of π such that $M_c = M_{c'}$

then take the leftmost such c and c';

$\pi := [\pi_{\min}, c']$;

merge c and c' to get (N_1^l, M_1^{0l}) from π ;

$(N^l, M^{0l}) := (N^l, M^{0l}) \cup (N_1^l, M_1^{0l})$;

$T' := T' \backslash \langle Pk([c, c']) \rangle$;

$M := M \backslash \langle Pk([c, c']) \rangle^\bullet$;

go to step 1;

end

$(N^l, M^{0l}) := (N^l, M^{0l}) \cup \pi$;

output (N^l, M^{0l})

end

For an SP-system (N, M^0) of Fig. 7.9, the system (N^l, M^{0l}) (the result of algorithm) is shown in Fig. 7.10.

Proposition 7.3.1 The algorithm stops at some step.

Proof:

The proof is given in appendix of the chapter.

■

In theorems and lemmata below (N^l, M^{0l}) is assumed to be nonempty.

We recall the concurrency relation [63] which can be computed with a polynomial algorithm.

The *concurrency relation* $\| \subseteq S \times S$ is defined above.

For (N, M^0) and corresponding (N^l, M^{0l}) we introduce a useful property ($\|$ is constructed for (N^l, M^{0l})):

$\forall t, t' \in T$ if $\exists S' \subseteq S_1$ such that $l(S') = {}^\bullet t \cup {}^\bullet t' | S' | = |{}^\bullet t \cup {}^\bullet t'|, \forall s, s' \in S'(s, s') \in \|$ then ${}^\bullet t \cap {}^\bullet t' = \emptyset$

(7.1)

For a class of systems satisfying the next property:

$$\forall \pi_1, \pi_2 \in \Pi(N, M^0) \exists \pi \in \Pi(N, M^0) \text{ such that } \pi_1 \subseteq \pi, \pi_2 \subseteq \pi \qquad (7.2)$$

partial order semantics can be reduced. That is, $\Pi(N, M^0)$ can be represented by the only process (possibly infinite).

Lemma 7.3.2 presents properties of (N^l, M^{0l}) and relations between the properties 7.1, 7.2 and persistence.

Lemma 7.3.2 Let (N, M^0) be a system and (N^l, M^{0l}) be its unfolding. Then:

(a) (N^l, M^{0l}) is safe and conflict-free and hence satisfies 7.2;

(b) if (N, M^0) is safe and persistent then 7.1 holds for (N^l, M^{0l});

(c) (N, M^0) and (N^l, M^{0l}) are behavior equivalent iff 7.2 holds for (N, M^0).

Proof:

(a) easily follows by construction of (N^l, M^{0l}) and the firing rule.

(b) (Indirect). Let (N, M^0) not satisfy the property 7.1. Then $\exists t_1 \neq t_2 \in T \quad \exists M \in [M^0\rangle (M[t_1\rangle) \wedge (M[t_2\rangle) \wedge ({}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset)$. By persistence $M[t_1 t_2\rangle$ and $M[t_2 t_1\rangle$. Since (N, M^0) is safe, by Lemma 7.1.1 we have ${}^\bullet t_1 \cap {}^\bullet t_2 = t_1^\bullet \cap t_2^\bullet \neq \emptyset$ and hence a fictitious conflict. A contradiction.

(c) By Lemma 7.3.2a (N^l, M^{0l}) satisfies 7.2. The rest follows from definitions and the algorithm.

■

Theorem 7.3.3 represents a result about behavior equivalence between safe persistent systems and safe conflict-free systems, which are the result of unfoldings of persistent systems.

Theorem 7.3.3 Let (N, M^0) be a system and (N^l, M^{0l}) be its unfolding. Then (N, M^0) is an SP-system iff (N, M^0) and (N^l, M^{0l}) are behavior equivalent.

Proof:

$\Leftrightarrow$. Let (N, M^0) be an SP-system. Then by Lemma 7.3.2b (N, M^0) satisfies 7.1. Since (N, M^0) is safe, it is easy to see that 7.1 implies 7.2 for (N, M^0). By Lemma 7.3.2c (N, M^0) and (N^l, M^{0l}) are behavior equivalent.

$\Leftarrow$. By Lemma 7.3.2a (N^l, M^{0l}) satisfies 7.2. Since (N, M^0) and (N^l, M^{0l}) are behavior equivalent, (N, M^0) also satisfies 7.2. It is easy to see that 7.2 implies persistence of (N, M^0).

Claim. Property 7.2 implies safety of (N, M^0);

Proof of the claim. (Indirect.) Let (N, M^0) not be safe. Since (N^l, M^{0l}) is not empty, it is easy to see that $\exists c \in BC(\pi) \exists b_1 \neq b_2 \in c$ such that $p(b_1) = p(b_2), b_1 \neq \emptyset$ or $b_2 \neq \emptyset$. Then 7.2 does not hold. A contradiction.

It follows from the claim that (N, M^0) is safe.

■

However Theorem 7.3.3 is weak for verification because it uses the notion of behavior which can be infinite. In order to simplify the problem of verification we give Theorem 7.3.4, which does not use the notions of behavior, language, process or $[M^0\rangle$. This theorem uses only $\|$ for (N^l, M^{0l}).

Theorem 7.3.4 Let (N, M^0) be a system and (N_1^l, M_1^{0l}) be its unfolding. Then (N, M^0) is an SP-system iff 7.1 and

$$\forall s_1 \neq s_2 \in S_1 \text{ such that } l(s_1) = l(s_2)(s_1, s_2) \notin \| \tag{7.3}$$

hold.

Proof:

$\Rightarrow$. $\Pi(N_1^l, M_1^{0l}) \subseteq \Pi(N, M^0)$ follows from construction of (N^l, M^{0l}) and the firing rule. Then the safety of (N, M^0) implies 7.3. Since (N, M^0) is an SP-system, by Lemmma 7.3.2b, 7.1 holds.

$\Leftarrow$. It is easy to see that 7.1 implies the persistence of (N, M^0) and that (N, M^0) and (N^l, M^{0l}) have the same $[M^0\rangle$. Then 7.3 implies safety of (N, M^0).

■

To verify if the given SP-system is L4-live (LSP), Theorem 7.2.1a is applied. And to verify if the given SP-system is L4-live and reversible (LSPR), Theorem 7.3.5 below is applied.

Theorem 7.3.5 Let (N, M^0) be an SP-system and (N^l, M^{0l}) be its unfolding. Then (N, M^0) is $L4$-live and reversible iff (N^l, M^{0l}) is strongly connected.

Proof:

The proof is easy to see.

■

7.4 Summary

The main results of the chapter are following.

1) We present a simple algorithm of unfolding of ordinary Petri nets by conflict-free ones.

2) We prove a proposition about behavior equivalence.

3) We give a necessary and sufficient condition of persistence which does not use the notions of behavior or state space.

4) We also solve the problems of the previous approaches. The system we got as a result of unfolding, is allowed to be a conflict-free system and not necessarily live, instead of requiring live safe marked graphs as in previous papers. The admission

extends the class of unfolded systems to the non-reversible ones. Our method of unfoldings allows us to handle the problem of fictitious conflicts. This method can also be applied in those cases when the number of reproduction processes exceeds one.

The limitation of this method is a comparatively small class of the correct systems (safe and persistent Petri nets).

This method can be applied in those areas where the requirements of correctness for modeled systems include safeness and persistence.

7.5 Appendix

Lemma 7.5.1 (a) If $(\forall v_1, v_2 \in V \subseteq \mathbb{N}^k \neg(v_1 \leq v_2))$ then V is finite.

(b) Let $\sigma = v_1 v_2 ... v_i ...$, be an infinite sequence, where $v_i \in \mathbb{N}^k, k \in \mathbb{N}$. Then there exists infinite subsequence $\sigma' = v^1 v^2 \dots v^j \dots$ of σ such that $\forall j \in \{1, 2, \dots\} v^j \leq v^{j+1}$.

Proof:

(a) The proof is given in [76] (Lemma 4.1).

(b) The proof is given in [84] (Lemma 4.3).

■

Lemma 7.5.2 Let (N, M^0) be a system.

(a) $T^3(M^0) = \cup_{v \in Pn} \langle v \rangle$

(b) Pf is finite.

Proof:

(a) $T^3(M^0) \supseteq \cup_{v \in Pn} \langle v \rangle$ is trivial. Now prove $T^3(M^0) \subseteq \cup_{v \in Pn} \langle v \rangle$. By definition of $L3$-liveness $\exists \sigma \in L(M^0) |\sigma|_t = \infty$. By Lemma 7.5.1b σ can be represented as

$\sigma = \sigma_0\sigma_1\sigma_2..\sigma_i.$, where $M^0[\sigma_0\rangle M^1[\sigma_1\rangle M^2[\sigma_2\rangle..M^i[\sigma_i\rangle.., \forall i = 0,1,2,..\sigma_i$ is finite and $M^i \leq M^{i+1}$. Hence $\forall i = 1,2,..\sigma_i \in Ln(M^i)$. Since $|\sigma|_t = \infty, \exists i \in \{0,1,2,..\}$ such that $|\sigma_i|_t > 0$.

(b) The proof is given in [76] (Lemma 4.2).

■

The following lemma gives a necessary and sufficient condition of persistence.

Lemma 7.5.3 (N, M^0) is a P-system iff $[\sigma_1 \in L(M), \sigma_2 \in L(M)] \rightarrow [\sigma_1(\sigma_2 \div \sigma_1) \in L(M)]$.

Proof:

The proof is given in [76] (Lemma 3.1).

■

Lemma 7.5.4 gives some known properties for persistent Petri nets:

(a) every two markings of $[M^0\rangle$ have at least one common successor;

(b) there do not exist strictly $L2$-live and $L3$-live transitions;

(c) every nonfinising firing sequence can be decomposed on elementary repeatable firing sequences.

Lemma 7.5.4 Let (N, M^0) be a P-system. Then:

(a)$\forall M^1, M^2 \in [M^0\rangle[M^1\rangle \cap [M^2\rangle \neq \emptyset$;

(b) $\forall M \in [M^0\rangle T^2(M) \subseteq T^4(M)$;

(c) if $\sigma \in Ln(M)$ then there exists a sequence $\sigma_1\sigma_2 \ldots \sigma_r, r \geq 1, \sigma_i \in T^+$ such that

1)$Pk(\sigma_i) \in Pf, i \in \{1,2,\ldots,r\}$;

2)$C \cdot Pk(\sigma_1\sigma_2 \ldots \sigma_r) = C \cdot Pk(\sigma)$;

3)$\sigma_1\sigma_2 \ldots \sigma_r \in Ln(M)$.

Proof:

(a) and (b) are Theorem 1.8 and Lemma 4.7 [39] adapted for Petri nets. (c) is Lemma 4.3 [76].

■

Proof:

of Lemma 7.1.2. (a) If $\exists\sigma_1 \in L(M^1)$ then since $\sigma \in Ln(M)$ and $Pk(\sigma) = Pk(\sigma_1)$, it must be $\sigma_1 \in Ln(M^1)$. Let $M[\sigma'\rangle M^1$. We continue the proof by induction on $|\sigma'|$, where $|\sigma'|$ is the number of transitions firings in σ'.

Basis: $|\sigma'| = 0$ is trivial.

Step: $|\sigma'| > 0$. Let the lemma conditions hold for σ' i.e. $M[\sigma\rangle, M[\sigma'\rangle M^1[\sigma_1\rangle$ and $Pk(\sigma) = Pk(\sigma_1)$. We prove for $\sigma' t$ now. By Lemma 7.5.3, $M^1[t\rangle M^2[\sigma_1 \div t\rangle$. Two cases are possible.

1) $t \notin \langle Pk(\sigma_1)\rangle$. Then $\sigma_1 \div t = \sigma_1$ and $M^1[t\rangle M^2[\sigma_1\rangle$.*q.e.d.*

2) $t \in \langle Pk(\sigma_1)\rangle$. Then $\sigma_1 \div t \neq \sigma_1$ and $M^1[t\rangle M^2[\sigma_1 \div t\rangle M^3$. Since $Pk(t(\sigma_1 \div t)) = Pk(\sigma_1)$ and $\sigma_1 \in Ln(M), M^3 \geq M^1$. But $M^1[t\rangle$, hence $M^3[t\rangle$. Then $M^2[(\sigma_1 \div t)t\rangle$. It is obvious $Pk((\sigma_1 \div t)t) = Pk(\sigma_1)$.

(b) By Lemma 7.5.3, $M[\sigma_1(\sigma_2 \div \sigma_1)\rangle M_1$. By Lemma 7.1.2a, $\exists\sigma_1', \sigma_2' \in Ln(M_1)$ such that $Pk(\sigma_1') = Pk(\sigma_1), Pk(\sigma_2') = Pk(\sigma_2)$. Hence $M_1[\sigma_1'(\sigma_2' \div \sigma' 1)\rangle M_2$ and $Pk(\sigma_1(\sigma_2 \div \sigma_1)) = Pk(\sigma_1'(\sigma_2' \div \sigma' 1))$. We can continue the proof for M_2 as well as for M_1. We get an infinite sequence $M[\sigma^0\rangle M^1[\sigma^1\rangle M^2[\sigma^2\rangle \ldots$, where $Pk[\sigma^i\rangle = Pk[\sigma_1(\sigma_2 \div \sigma_1)\rangle, i \in \mathbb{N}$. Hence $\sigma_1(\sigma_2 \div \sigma_1) \in Ln(M)$.

(c) (Indirect). Let $\exists v_1 \neq v_2 \in$mPn such that $\langle v_1\rangle \cap \langle v_2\rangle \neq \emptyset$ and $v_1 \in mPr$. Then by Lemmata 7.1.2a and 7.5.4a $\exists M \in [M^0\rangle \exists\sigma_1, \sigma_2 \in L(M)$ such that $v_1 = Pk(\sigma_1), v_2 = Pk(\sigma_2)$. By Lemma 7.1.2b, $\sigma_1(\sigma_2 \div \sigma_1) \in Ln(M)$. Since $v_1 \in$mPr, $Pk(\sigma_2 \div \sigma_1) \in Pn$. Since $v_1, v_2 \in mPn$, $\sigma_2 \div \sigma_1 \neq \lambda$. Since $\langle v_1\rangle \cap \langle v_2\rangle \neq \emptyset$, $\sigma_2 \div \sigma_1 \neq \sigma_2$. Hence $Pk(\sigma_2) \notin mPn$. A contradiction.

(d) By Lemmata 7.1.1a and 7.5.4b $\forall M \in [M^0\rangle T^4(M^0) = T^3(M^0) = T^2(M^0) =$

$T^4(M) = T^3(M) = T^2(M)$, by Lemma 7.5.2a $T^3(M^0) = \cup_{v \in Pn}\langle v\rangle$, by Lemma 7.5.4c $\cup_{v \in Pn}\langle v\rangle = \cup_{v \in Pf}\langle v\rangle$.

(e) Since (N, M^0) is bounded, mPn=mPr=Pf. Then apply Lemma 7.1.2cd.

■

Proof:

of Theorem 7.2.1. (a) $\Leftrightarrow$ (Indirect). Let $|mPr| > 1$. By Lemma 7.1.2e, $\{\langle v\rangle | v \in$mPr$\}$ is a partition on the set T^4. Since N is connected, $\exists t_1 \in \langle v_1\rangle \exists t_2 \in \langle v_2\rangle$ such that $\langle v_1\rangle \cap \langle v_2\rangle = \emptyset$ and $(t_1, t_2) \notin$ *indep*. By Lemmata 7.5.4a and 7.1.2a $\exists M \in [M^0\rangle M[t_1\rangle \wedge M[t_2\rangle$. By Lemma 7.1.1b we have a fictitious conflict. A contradiction. Hence |mPr| = 1. Let mPr= $\{v\}$ and $T\backslash\langle v\rangle \neq \emptyset$. Since $mPr = \{v\}$, by Lemma 7.1.2d $(T\backslash\langle v\rangle) \cap T^4 = \emptyset$. A contradiction to L4-liveness of (N, M^0). Hence $\langle v\rangle = T$.

$\Leftarrow$ Since (N, M^0) is bounded $mPn = mPr = Pf$. Then $T^4(M^0) = T$ and (N, M^0) is L4-live.

(b) is simple.

■

Proof:

of Proposition 7.3.1. By Lemma 7.5.2b the set mPr is finite. After finding of every π' such that $Pk(\pi') \in mPn\backslash mPr$, the algorithm stops because of non-safety of (N, M^0). After finding every π' such that $Pk(\pi') \in$mPr the marking M and the set T_1 of transitions decrease, so that we can not find any other π" with $Pk(\pi') = Pk(\pi$"). After finding of all mPr, only strictly $L1$-live transitions are left which can be fired only a finite number of times.

■

Chapter 8

An Efficient Modular Synthesis of Regular Petri Nets by Simple Composition Rules

The synthesis problem can be stated as follows: given a set of properties of good behavior, how does one construct systems satisfying them? One of basic methodologies of synthesis is modular synthesis. It deals with the case when modules (subsystems) are merged (composed) into new systems.

The system is divided into modules that can be easily modeled. In practice, the design of systems traditionally uses state machine model which is very familiar to engineers and designers. Each module can be represented in the model autonomously. Then such representations should be coordinated to describe the whole system. The corresponding Petri net model is a state machine decomposable net (SMD-net for short). There is a broad spectrum of models supporting modular approach to system design, for instance, SDL [32], CSP [35], CCS [81], LCS [78], COSY [77].

The problem of modular synthesis can be stated as follows: given a set of well-behaved modules, how does one compose the modules to yield a well-behaved system.

What are criteria of good behavior in concurrent systems? We consider two central

properties of Petri nets: liveness and boundedness.

There are three major concerns in modular synthesis:

1) The class of modeled systems should be wide enough.

2) Algorithms for synthesis should be efficient.

3) The synthesis procedure should produce the well-behaved result.

In different papers only subsets of the problems were solved [4, 29, 30, 31]. In [29, 30, 31] the condition of well-formedness of the resulting net is formulated using the conditions of well-formedness of modules. However, the class of nets under consideration is free-choice nets, quite a narrow class.

In [4] behavioral and structural composition rules preserving liveness for high-level Petri nets are given. However, they consider only a special class of Coloured FIFO nets. Moreover, no efficient algorithm for checking of preserving liveness is given.

In this chapter we handle all three above problems together and present two efficient methods for modular synthesis of regular Petri nets, a subclass of SMD-nets. We also give exact conditions for the preservation of well-formedness under synchronization of nets (a particular kind of composition in which transitions are merged), and fusion (places are merged).

8.1 Synchronizations

The notion of a CP-subnet can be found in [22]. For our convenience we slightly changed the definition and introduced two kinds of CP-subnets: one is a TCP-subnet (for T-cover), and another is a SCP-subnet (for S-cover). So the notion of a SCP-subnet is reverse-dual to that of a TCP-subnet.

Theorem 8.1.1 [46] Let N be a feedback-free S-coverable net and not a SCSM. Let N have a SCP-subnet $\widehat{N}$ of N with 4.2 and the following two conditions:

$$|\widehat{S}_i| = 1 \tag{8.1}$$

$$(\overline{S}, \overline{T}) \text{ is } S - \textit{coverable} \tag{8.2}$$

Then N is regular iff $\overline{N}$ is regular.

Proof:

The proof follows from reverse-dual version of Theorem 4.2.1.

■

Now we consider a composition of subsystems, and how to interconnect the subsystems to yield a well-behaved system. Within our context, the problem can be formulated as follows: given several regular nets, characterize the composition that preserves well-formedness.

Since composition of k nets can be split into $k-1$ composition of 2 nets, we consider only this particular case.

N is a *composition* of N_1 and N_2 iff

(1) N_1 and N_2 are subnets of N and

(2) $N = N_1 \cup N_2 = (S_1 \cup S_2, T_1 \cup T_2, F_1 \cup F_2)$.

N is a *synchronization* of N_1 and N_2

iff

(1) N is a composition of N_1 and N_2 and

(2) $s \in S_1 \cap S_2 \Rightarrow {}^\bullet s \cup s^\bullet \in T_1 \cap T_2$.

Let N_1 and N_2 be two nets. N is an *FC-synchronization* of two FC-nets N_1 and N_2 iff N is also FC.

The next theorem is the Monotonicity result, and gives us a necessary condition for well-formedness of an FC-synchronization. It says that if one of two synchronized

FC-systems is not well-formed then the synchronization can not be well-formed, and can not become well-formed after further synchronizations.

Theorem 8.1.2 [31] Let N be an FC-synchronization of N_1 and N_2. If N is well-formed then N_1 and N_2 are well-formed.

■

Now we extend this result to regular nets.

Theorem 8.1.3 Let N be a synchronization of N_1 and N_2. If N is a regular net then N_1 and N_2 are regular nets.

Proof:

Follows from Theorems 8.1.2 and 3.1.9a.

■

Theorem 8.1.3 allows us to use a SCSM as N_1 or N_2 without loss of generality, because by Lemma 3.4.2, regular nets are state machine decomposable.

As we noted in the introduction, one of the major problems of modular synthesis is that the synthesis procedure should produce the well-behaved result. Theorem 8.1.3 gives us the necessary condition of well-formedness of the result. However the condition is not sufficient, as we see in Fig. 8.1. Both free-choice modules are well-formed, but the result of FC-synchronization is not.

In Fig. 8.2 the synchronization of the same modules produces the well-formed result. Therefore there should be special requirements for the synchronization procedure in order to get well-formed result. For the FC-synchronization such requirements were given in [4, 29, 30, 31]. The next theorem gives such requirements for regular nets.

Theorem 8.1.4 Let N be a synchronization of a SCSM N_1 and a regular net N_2. If N has a SCP-subnet $\widehat{N} \subset N_1$ with 4.2, 8.1 and 8.2 then N is regular.

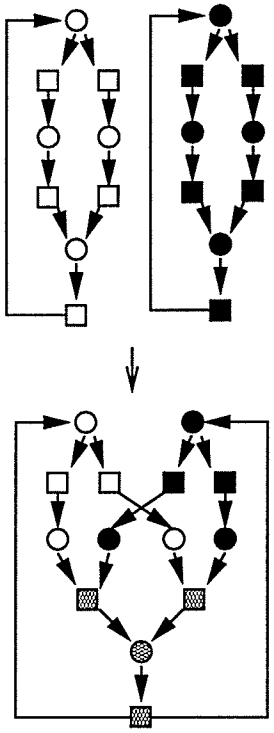

Fig. 8.1 An FC-synchronization which is not Well-formed.

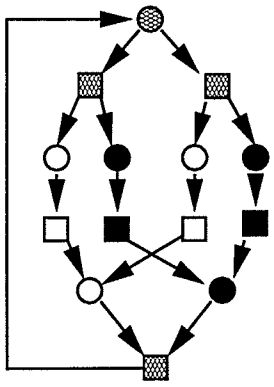

Fig. 8.2 An FC-synchronization which is Well-formed.

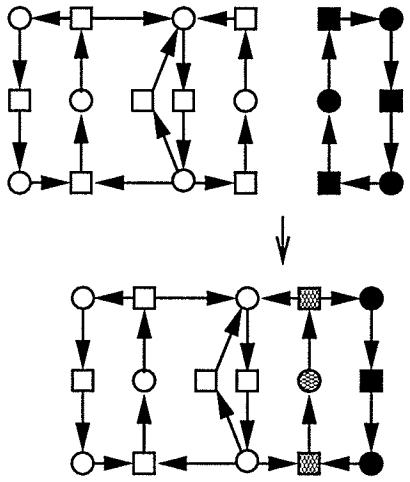

Fig. 8.3 A Synchronization of a Regular Net and a SCSM.

Proof:

By induction on n, where N is the number of SCP-subnets $\widehat{N}$ of N such that $\widehat{N} \subset N_1$.

Basis. $n = 1$. Then $\overline{N} = N_2$, and by Theorem 8.1.1 we are done.

Step. $n > 1$. We consider the system $\overline{N} = N \setminus \widehat{N}$. By 8.2 it is SMD. Since $\widehat{N} \subset N_1$, $N_2 \subset \overline{N}$. Hence $\overline{N}$ is a synchronization of N_2 and several SCSM N_3, N_4, ..., N_k. First we consider N_s, the synchronization of N_2 and the SCSM N_3. Since $N_s \subset N$, we have $n_s < n$, where n_s is the number of the SCP-subnets $\widehat{N}$ of N_s such that $\widehat{N} \subset N_3$. By the induction hypothesis, N_s is a regular net. Repeating this argument with N_4, ..., N_k, we eventually get that $\overline{N}$ is a regular net. By Theorem 8.1.1, N is a regular net.

■

Example. A synchronization of two regular nets is shown in Fig. 8.3. The result is also a regular net.

Proposition 8.1.5 The problem to check if a synchronization of a SCSM N_1 and a regular net N_2 is regular, has the complexity $O(c \times |T|)$, where $|T|$ is the number of transitions of the synchronization, and c is the number of all the CP-subnets $\widehat{N} \subset N_1$.

Proof:

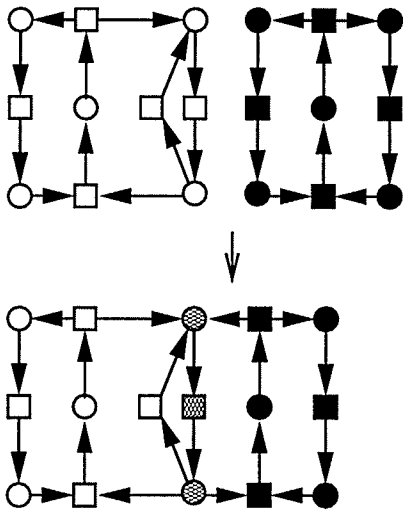

Fig. 8.4 A Fusion of a Regular Net and a SCMG.

Properties 4.2 and 8.1 are local and can be verified in $O(|T_1|)$ time. Property 8.2 requires one to obtain a cover by S-components. Since N_2 is regular, by Lemmma 3.5.6, it is SMD. Hence we need only to check whether all SCP-subnets of N_1 are covered by S-components. In [57] the complexity of the problem is given as $O(|T|)$. Hence the overall complexity of checking of the synchronization is $O(c \times |T|)$.

■

8.2 Fusions

N is a *fusion* of N_1 and N_2 iff

(1) N is composition of N_1 and N_2 and

(2) $t \in T_1 \cap T_2 \Rightarrow^{\bullet} t \cup t^{\bullet} \in S_1 \cap S_2$

Example. A fusion of two nets is shown in Fig. 8.4.

For Propositions 8.1.3, 8.1.4 and 8.1.5 we have their dual counterparts.

Theorem 8.2.1 Let N be a fusion of N_1 and N_2. If N is regular then N_1 and N_2 are regular.

■

Theorem 8.2.2 Let N be a fusion of a SCMG N_1 and a regular net N_2. If N has a TCP-subnet $\widehat{N} \subset N_1$ with 4.2, 4.1 and 4.3 then N is regular.

■

Theorem 8.2.3 The problem to check if a fusion of the SCMG N_1 and a regular net N_2 is regular has the complexity $O(c \times |S|)$, where $|S|$ is the number of transitions of the fusion, and c is the number of all the TCP-subnets $\widehat{N} \subset N_1$.

■

8.3 Summary

We have presented two efficient methods of modular synthesis of regular Petri nets, a subclass of state machine decomposable Petri nets.

The first method deals with the synchronization of nets (transitions are merged). The main problem is to check if a synchronization of a regular net and a strongly connected state machine is regular. The complexity of this checking is $O(|T|)$, where $|T|$ is a number of transitions of the state machine. This method is due to exact condition of regularity of synchronization.

The second method deals with fusion of nets. It is the dual version of the first. The main problem is to check if a fusion of a regular net and a strongly connected marked graph is regular. The complexity of this checking is $O(|S|)$, where $|S|$ is a number of places of the marked graph.

The advantage of our approach is that we solve all three major problems of modular synthesis: relatively wide class of nets, efficient algorithms and well-behaved result of synthesis.

Our study provides deeper insight into the relationship of the behavior of Petri nets and their structure.

Applications of our methods are numerous, particularly for synthesis of systems in programming languages, such as SDL [32], CSP [35], CCS [81], LCS [78], COSY [77].

When we compose systems (nets with initial markings) the composition net (synchronization or fusion) can be regular, but the system can be live or non-live depending on initial marking. To check if the initial marking of a composition is live, we use the $O(|S| \times |T|)$-algorithm of [57], where S and T are the sets of places and transitions of the synchronization.

In the future we are going to develop an efficient method of modular synthesis for the whole class of state machine decomposable nets.

Chapter 9

Application to Hardware Design

When the complexity of design increases, it is helpful to move from semi-formal specification of hardware (for example, timing diagrams) to a more abstract form, similar to software specification. Because in this case doing modification, reuse, modularity, hierarchy and technology independence are much easier, than with using of semi-formal models.

One of formal models of hardware is Finite State Machine. The shortcomings of Finite State Machine are due to their problems with adequate interpretation of concurrency and synchronization. However, the concurrency is inherent to electronic circuits, because the signals run concurrently, and we have large number of global states in digital hardware. So, Finite State Machine becomes very large and inconvinient. There is also such concurrent phenomena as races. Especially, Petri Nets are similar to asynchronous circuits.

9.1 Signal Transition Graphs

Signal Transition Graphs (STGs) [15, 80] are a class of interpreted Petri nets for modeling, verification and synthesis of *speed independent circuits.* STGs are especially useful for the design of hardware with reactive behavior. STG model allows automatic

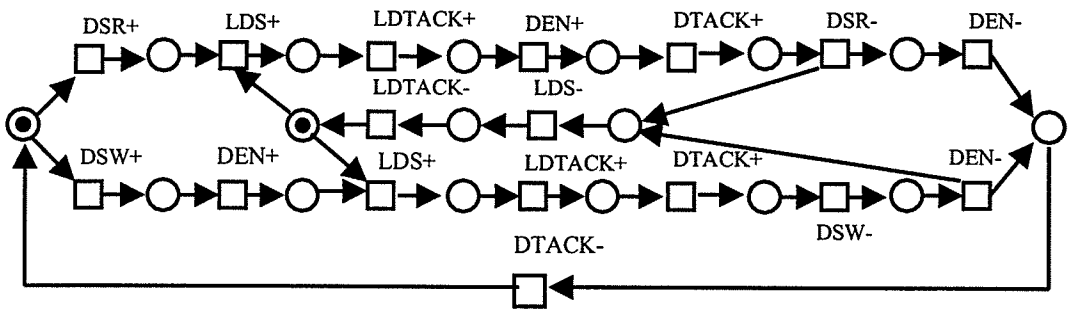

Fig. 9.1 Smaller Version of VME-bus controller.

verification and implementation of circuits.

STGs are bounded Petri nets whose transitions are labeled with the changes of binary signals. The occurrence of a transition with label x^+ *raises* x, while the occurrence of a transition with label x^- *lowers* x.

We use an example to demonstrate the application of the concurrency relation for the hardware design from the STG specifications. In [99] the process of modeling VME-bus Slave-Interrupter controller by STG was described. The original protocol specifications are defined by timing diagrams. Using a technique from [16] the protocol is specified by the STG.

Petri net model of a VME-bus controller is shown in Fig. 9.1.

This device synchronizes two handshake protocols, one at the VME-bus link and other at the link with the device. The first handshake involves bus data strobe signals DSR (read operation) or DSW (write operation) and acknowledgement DTACK. The second handshake involves the local data strobe command LDS and local acknowledgement LDTACK. The process of synchronization includes an additional signal, DEN, to control data bus buffers.

The STG combines both Read and Write operations into a single model. This is due to the ability of Petri nets to model choice using places with several incident output transitions. The STG also captures potential concurrency by allowing some transitions to fire independently.

The signal abbreviations are as follows:

DSR = data strobe read;

DSW = data strobe write;

DTACK = data acknowledgement;

DEN = data buffer enable;

LDS = local data strobe;

LDTACK = local data acknowledgement.

To produce hazard-free implementations, STGs must be *consistently encoded.* In current synthesis tools, the verification of this property is carried out by the construction of the *reachability graph* and therefore is very expensive computationally.

Recently, some synthesis methods were proposed which avoid the reachability graph construction. They have low polynomial complexity, but require knowledge of the *concurrency relation* between transitions. Therefore, an efficient polynomial algorithm for the computation of the concurrency relation between transitions has become very important.

For the above example of an STG, we compute the concurrency relation using our algorithm (Fig. 9.2).

But STGs are not only application of the concurrency relation. Another application is the verification of concurrent control algorithms.

The problem of concurrency relation computation is exponential for arbitrary Petri nets, even for 1-bounded Petri nets [14].

FC- and EFC Petri nets can model conflict and concurrency, they are easy to analyze, but they are too restrictive for some applications. The article [98] contains strong arguments in favor of lifting free-choice limitation for Signal Transition Graphs. So we need to compute the concurrency relation for wider classes of Petri nets.

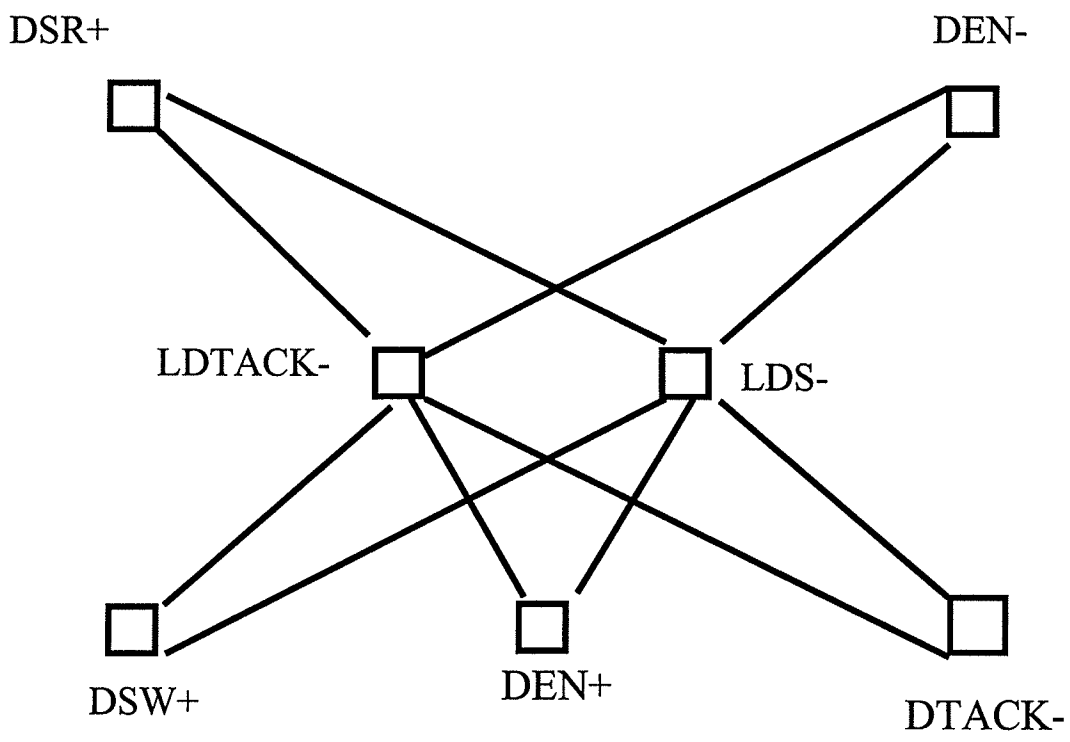

Fig. 9.2 The concurrency relation for the VME-bus controller.

9.2 Summary

In [49] we generalize the previous algorithm of the concurrency relation construction for free-choice Petri nets [55], to regular Petri nets, and therefore to regular STGs. The time complexity of the algorithm is $O(n^4)$, where n is the number of nodes of the net.

The usefulness of our algorithm is raised by conjecture that it is valid for the whole class of state machine decomposable Petri nets. The proof is left for future work. For instance, the Signal Transition Graph from [98] (a smaller version of a model of the VME-bus controller) is SMD, and our algorithm is valid for it.

Our work was motivated by interesting applications of the concurrency relation to the design and verification of asynchronous circuits. Our algorithm can be used to detect inconsistently encoded regular STGs.

Chapter 10

Application to Logical Control

10.1 Logical Control Algorithms

Here we consider logical control in automated production. This production is understood as a number of distributed facilities. The different pieces of this distributed dynamic system are working independently and interacting with each other from time to time. The system can be viewed as a condition-event system. Examples of conditions: a part is ready for processing or some resources are available. Examples of events: closing a valve, begin of processing a part, end of processing a part. The control system is waiting for a condition in the system to satisfy, then it performs some control action, causing another event and other conditions to satisfy.

One way of solving the problems of logical control in automated production is to reflect the concurrency of control objects in control algorithms.

These problems cannot be resolved simply by borrowing appropriate models and methods from the theory of parallel computations because control algorithms basically differ from computational ones and represent open dynamic systems continuously interacting with the control objects. Classical automaton theory is also not sufficient for that purpose because it only deals with strictly sequential discrete processes.

A number of original models have been proposed. These models define the concurrent

control algorithms and are related, to a lesser or greater degree, to Petri nets. The Logical Control Algorithms (LCAs) [101] are also such a model.

It is natural to begin the design of a logical control device with formulating a logical control algorithm deriving it from a notion of the behavior of a system that has to be put under control. LCAs are formal models proposed for the description of such algorithms in terms of input and output Boolean variables of control devices.

The LCA model combines the properties of formality, universality and simplicity. Formality facilitates automation of the development of control algorithms, verification and the synthesis of the structures realizing them. Universality allows for a wide application. Simplicity facilitates the use of the model in engineering practice and also promotes the development of the theory of the model.

LCA is a class of interpreted Petri nets, whose transitions are labeled with pairs of operations. The first element $-k_1$ is said to be the waiting operation and the second $\rightarrow k_2$ is the acting operation. The transitions may fire analogously to the transitions of a Petri net, but the firing rule is slightly modified to take into account the information in the waiting and acting operations. The waiting operation does not change any values but waits until k_1 becomes equal to 1. The acting operation assigns to the variables of k_2 such values that satisfy the equation $k_2 = 1$.

The pair of operations $-k_1 \rightarrow k_2$ assigned to a transition of the algorithm specifies the condition-event relationship between the simple events represented by conjunction terms k_1 and k_2: the event k_1 gives rise to the event k_2.

Let us classify the Boolean variables of the waiting and acting operations into three classes: X - input variables in k_1 only; Y - output variables in k_2 only; Z - internal variables encountered both in k_1 and k_2.

A logical control algorithm is presented as a triple (N, M^0, P), where (N, M^0) is an underlying Petri net and is said to be a skeleton of an algorithm, and P is the set of pairs of operations assigned to transitions.

A logical control algorithm is shown in Fig. 10.1.

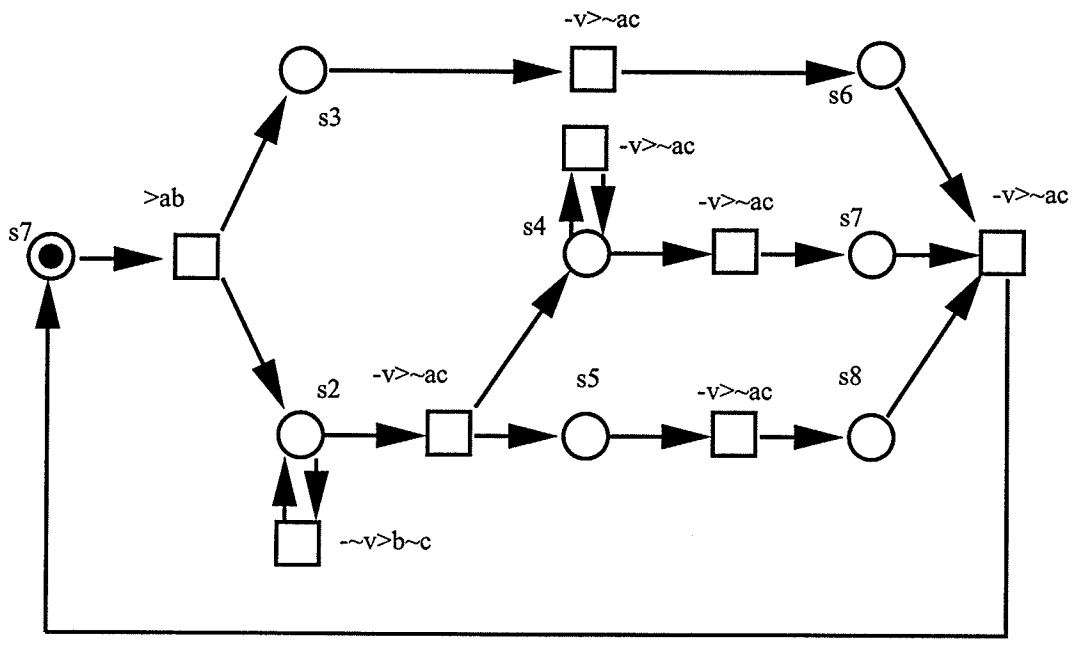

Fig. 10.1 A Logical Control Algorithm.

For ensuring the determinism of an algorithm's execution, the following constraints are introduced into the model: for two conflict transitions t_i and t_j the terms k_1^i and k_1^j should be orthogonal.

So called two-terminal algorithms may be used as blocks in hierarchical algorithms.

One mechanism for transitions to interact is contained in the skeleton of an algorithm. The remaining information is contained in waiting and acting operations, which ensure informational interactions between transitions.

10.2 Logical Control Algorithm Correctness

Now we consider the verification of algorithm correctness. The following proposition was given in [101].

An algorithm is correct iff it is live, safe, persistent and consistent.

The algorithm is consistent iff any of its concurrent transitions t_i and t_j satisfy the condition: $k_2^i \wedge k_2^j \neq 0$. The algorithm is persistent iff the completion of one of the

parallel chains being performed does not destroy conditions for executing the others ($k_1^i \wedge k_2^j \neq 0$ and $k_1^j \wedge k_2^i \neq 0$ for the concurrent chains).

The verification of each of these properties represents a nontrivial combinatorial problem. Let us consider some of them.

Abstracting of informational interaction we propose the next criterion of correctness: The algorithm A is correct iff its skeleton (N, M^0) is live and safe.

A skeleton of a two terminal algorithm has only one marked place in the initial marking and is called *α-net.*

To take into account informational interaction of transitions we first model the algorithm by a Petri net (N_1, M_1^0): we replace each internal variable by corresponding pairs of places v and $\overline{v}$, and connect the places to the corresponding transitions by arcs.

We prove the following proposition in [69].

The LCA (N, M^0, P) is correct iff (N_1, M_1^0) is a live safe and persistent Petri net.

10.3 Summary

In [71, 72, 73, 64] we develop a set of simple local reduction rules for α-nets and EFC-nets, prove the hypothesis on complete reducibility of live and safe α-nets, and give another hypothesis on complete reducibility of live and bounded extended free-choice systems. In [49] we give a complete reduction method for regular Petri nets and estimate its complexity as $O(|S| \times |T|)$. In [70] we give a decomposition method for Logical Control Algorithms.

In [54, 59, 60] we develop an $O(|S| \times |T|)$-algorithm to decide if a given EFC-system is live and bounded, which is a reduction by one order of magnitude, compared to the algorithm in [43] ($O(|S|^2 \times |T|)$). The algorithm [43] is based on the Rank Theorem. Two main steps of the algorithm [43] are: to find a cover of S-components

and to compute the rank of incidence matrix. Finding a cover of S-components has been done in [41] in $O(|S| \times |T|)$-time. Since the calculation of a matrix rank requires $O(|S|^2 \times |T|)$, the Rank Theorem can be checked in $O(|S|^2 \times |T|)$. Hence the calculation of the matrix rank dominates the complexity of the algorithm. To reduce the complexity we need other criterion of well-formedness. In [24] such a criterion has been given. In order to reduce the complexity of the decision algorithm we combine two approaches above. Since well-formedness is only a necessary condition for a system to be live and bounded, we need to decide if a given initial marking of a well-formed net is live and bounded. An $O(|S| \times |T|)$-algorithm to decide this problem is given.

In [60] an algorithm is given to decide if an EFC-system is live, and if it is not, to obtain the maximal live subnet or to answer that it is impossible.

In [57] we present an $O(|S| \times |T|)$-algorithm to decide if a given system is regular. To prove the soundness of our algorithm we give a theorem on the covering of strongly connected EFC-nets by minimal siphons.

In [56, 60, 51] new formulations of the Rank Theorem for extended free choice systems and for regular Petri nets are given. These formulations strengthen the known ones from [22, 13] in the sense that the new necessary and sufficient conditions of well-formedness are less strict than the known ones. A necessary condition of structural liveness for EFC-nets is given. A simple and unified proof for covering live and bounded EFC-systems by S- and T-components is proposed. This proof is based on structural properties of this class of nets. A theorem on the coverability of strongly connected EFC-nets by minimal siphons is given.

In [51] we generalize the Rank Theorems to general class of Petri nets.

In [61, 68] we solve the so called problem of diagnostics, i.e. not only checking liveness and boundedness properties of free choice Petri nets, but also finding the source of the error, i.e. Unbounded place and/or non-live transitions.

To verify the algorithm with informational interaction between transition in [62, 58]

we develop a method to decide whether a Petri net is safe and persistent. This problem has application at the verification of systems. To solve this problem we propose a method of simulation of ordinary Petri nets by conflict-free ones and prove a proposition about behavior equivalence of two classes of safe Petri nets: persistent systems and labeled conflict-free systems, which are the result of simulation of persistent systems. The behavior equivalence proves to be a necessary and sufficient condition for an ordinary system to be safe and persistent.

Chapter 11

Application to Workflow Management

11.1 Workflow Procedures

Workflow management provides us a new solution to an old problem: controlling, monitoring, optimizing and supporting business processes. The new feature is the formal representation of the business process logic which facilitates computerized support. Petri nets are a formal model which can be used as a formal design language for the specification of workflow processes and also as verification tool. In [1] such possibilities are considered.

A Petri net model of a *workflow procedure* is shown in Fig. 11.1. A procedure specifies the set of tasks and the partial order in which these tasks have to be executed.

Each task represents an elementary activity, for example, sending a message or printing a report. A Petri net model of a workflow procedure is called a *workflow net.* Tasks are modeled by transitions and precedence relations are modeled by places and arcs.

In [1] the definition of a workflow net is given:

A net N is a *workflow net* iff

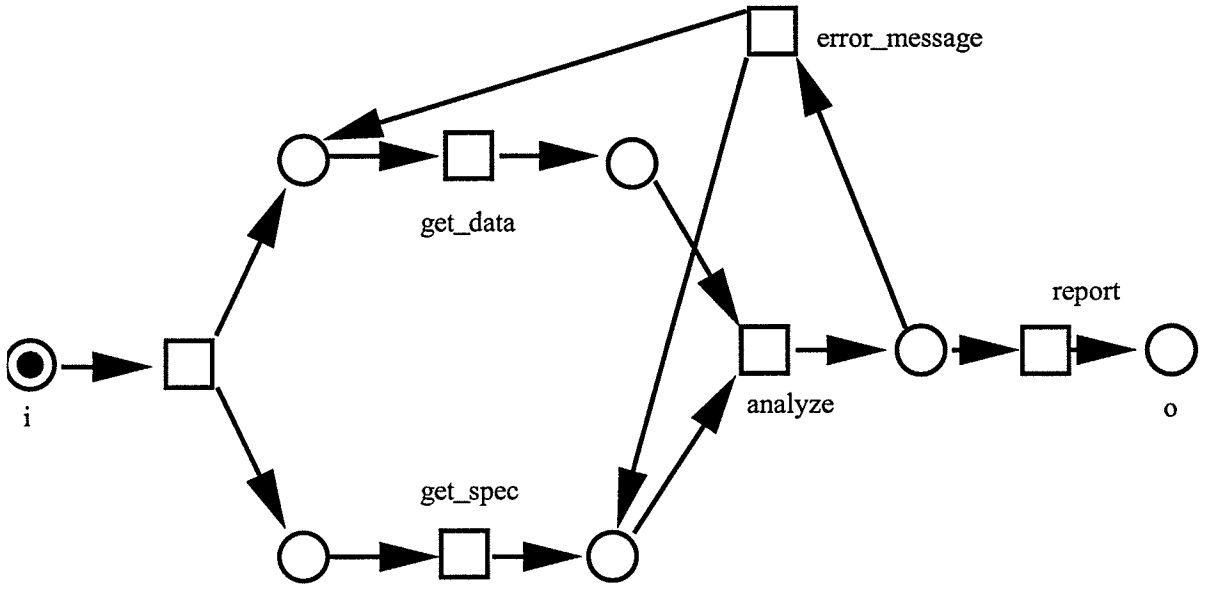

Fig. 11.1 A Workflow Procedure.

(a) N has two special places: i and o. Place i is a source place: ${}^\bullet i = \emptyset$. Place o is a sink place: $o^\bullet = \emptyset$.

(b) If we add a transition $\bar{t}$ to N which connects place o with i, then the resulting net is strongly connected.

By M_i (M_o) we denote a marking with the only token in a place i (o).

11.2 Strongly Connected Free-Choice Systems Having Nondead Home Markings are Live and Bounded

A home marking is a marking which is reachable from any other reachable marking. In [7] it is shown that live and safe extended free-choice systems have home markings. In [97] it is observed that the result could be easily generalized to live and bounded extended free-choice systems. In the present chapter we prove the reverse. Combining the results we get another necessary and sufficient condition of liveness

and boundedness expressed in terms of home markings: for an extended free-choice system to be live and bounded it is necessary and sufficient to be strongly connected and to have a nondead home marking. Strong connectedness can not be removed from the proposition because it is a necessary condition for liveness and boundedness but not for the existence of nondead home markings. The latter can be easily shown by a simple counter example.

The idea of the proof came while considering a problem stated in [1]: to prove that a live extended free-choice system is bounded if it has a positive T-invariant and is strongly connected.

Let (N, M^0) be a system. A marking M of the net N is a *home marking* of (N, M^0) iff $\forall M' \in [M^0\rangle \quad M \in [M'\rangle$. A marking M of the net N is *dead* iff there are no transitions fireable at M. A transition t is *dead* at a marking M iff $\forall M' \in [M\rangle \quad t$ is not fireable at M'.

Theorem 11.2.1 [7, 97] Live and bounded EFC-systems have home markings.

■

Lemma 11.2.2 [52] Let (N, M^0) be a live EFC-system. Then (N, M^0) is bounded iff it is strongly connected and satisfies 3.2.

Proof:

$\Rightarrow$. By Lemma 3.3.1e, N is strongly connected and has a positive T-invariant. Hence N satisfies 3.2.

$\Leftarrow$. Since (N, M^0) is live, by Theorem 5.1.1a (Hack's theorem), every siphon contains a trap. By Lemma 3.3.5a, every minimal siphon is an S-component. Since N is strongly connected, by Theorem 3.2.7, it is covered by minimal siphons and hence by S-components. Hence it has a positive S-invariant and is structurally bounded and bounded for each marking.

■

Theorem 11.2.3 Let (N, M^0) be an EFC-system. Then (N, M^0) is live and bounded iff it is strongly connected and has a nondead home marking.

Proof:

$\Rightarrow$. Lemma 3.3.1c and Theorem 11.2.1.

$\Leftarrow$. Let $M \in [M^0\rangle$ be a marking such that every transition is live or dead. Such a marking always exists by definition of liveness. Let $T_l \subseteq T$ be the set of all the live transitions at M. Since (N, M^0) has a nondead home marking, $T_l \neq \emptyset$. Let $M' \in [M\rangle$ be a home marking with $M'(s) = n$.

We prove now that $T_l = T$. (Indirect.) Suppose that $T_l \neq T$. Since N is strongly connected, there exists a place s and two transitions t_1 and t_2 such that $(t_1, s) \in F$, $(s, t_2) \in F$, t_1 is live, and t_2 is dead at M. By the EFC-property, all the transitions $s^\bullet$ are dead at M. Since t_1 is live at M, we can get $M'' \in [M\rangle$ with $M''(s) > n$, and the number can not be reduced after the firing of any transition. Hence $M' \notin [M''\rangle$, and M' is not a home marking. A contradiction to our supposition. Hence $T_l = T$, and (N, M^0) is live.

Let σ be a firing sequence from a home marking to itself which fires every transition of N. Such a sequence exists because (N, M^0) is live and has a home marking. Then $\#(\sigma)$ is a positive T-invariant (where $\#(\sigma)$ denotes a vector whose entries are the numbers of firings of each transition in σ). Hence N has a positive T-invariant and satisfies 3.2. Then by Lemma 11.2.2, (N, M^0) is bounded.

■

11.3 Soundness of Workflow Procedures

For verification of correctness of workflow nets the soundness property is introduced in [1].

A workflow net N is *sound* iff

(i) $\forall M \in [M_i\rangle \quad M_o \in [M\rangle$,

(ii) $\forall M \in [M_i\rangle \quad M \geq M_o \rightarrow M = M_o$,

(iii) $\forall t \in T \quad \exists M \in [M_i\rangle \quad M[t\rangle$.

For the verification of soundness property, the extended net $\overline{N}$ was defined by adding an extra transition $\bar{t}$ which connects o and i:

$\overline{P} = P$

$\overline{T} = T \cup \{\bar{t}\}$

$\overline{F} = F \cup \{(o, \bar{t}), (\bar{t}, i)\}$

Theorem 11.3.1 [1] (a) A workflow net N is sound iff $(\overline{N}, M_i)$ is live and bounded.

(b) For free-choice workflow net N, (i) and (ii) imply (iii).

■

Theorem 11.3.1a allows us to apply the Rank Theorems and our other techniques for verification of soundness.

Theorem 11.3.1b says that for free-choice workflow net N, (i) and (ii) imply (iii). That means that (iii) is redundant in the definition of soundness for free-choice workflow nets.

Using Theorem 11.2.3, we now prove stronger result than Theorem 11.3.1b.

Theorem 11.3.2 For free-choice workflow net N, (i) is sufficient for soundness and hence implies (ii) and (iii).

Proof:

The proof easily follows from Theorem 11.2.3 and the definition of a workflow net.

■

11.4 Summary

Here we prove a proposition which is stronger than the previous proposition from [1]. To prove it we used our theorem, that for extended free-choice systems: strong connectedness and the existence of home state is necessary and sufficient for liveness and boundedness.

Applying the theorem to extended free-choice workflow nets we get that (i) is necessary and sufficient for soundness and hence (i) implies (ii) and (iii). So (ii) and (iii) are redundant in the definition of soundness for extended free-choice workflow nets.

Chapter 12

Application to the Performance Evaluation of Communication Networks

After model development, performance evaluation is usually required. The developed universal methods face the state explosion problem. Therefore our objectives are the creation of efficient algorithms for performance evaluation of communication networks. Petri nets are an analytical model in performance evaluation. The advantages of analytical models in performance evaluation are:

1. the use of probabilistic approach with macroscopic assumptions, when the details are not known;
2. sufficient accuracy and generality;
3. a higher level of abstraction.

In general, exact performance results are obtained from the numerical solution of a Markov chain, whose dimension is given by the size of the state space of the model. Constructing models of complex systems directly at the Continues Time Markov Chain level is generally difficult, mainly due to the need of choosing an appropriate

state definition, and enumerating all states in the evaluation process. For this reason, more abstract probabilistic modeling tools were proposed.

Product form queuing networks are used for the performance evaluation of traditional communication systems. The shortcomings of queueing-based models are mainly due to their lack of descriptive power in presence of phenomena such as simultaneous possession of resources, synchronization, blocking and splitting of customers. Moreover these features, quite common in distributed and concurrent systems, generally destroy the product form solution.

More powerful model, which can model the above features is stochastic Petri nets.

Analysis algorithms for Petri nets are mostly non-efficient in general case because of the state explosion problem [37]. One of the techniques to cope with the problem is to identify a subclass of systems for which polynomial algorithms can be proposed. In [13] the polynomial algorithms for the performance evaluation of live and bounded free-choice Petri nets have been proposed. free-choice and extended free-choice Petri nets can model conflict and concurrency, they are easy to analyze, but they are too restricted for some applications. Therefore we need to generalize the method of [13] to wider classes of Petri nets. Here we generalize this method to regular Petri nets and propose polynomial algorithms for performance evaluation of communication systems using stochastic Petri nets. Quick computation of performance bounds is a complementary approach to the exact analysis, especially useful in the preliminary phases of the design. We evaluate the upper and lower bounds for the throughput, the mean queue length, the Probability Mass function of the number of customers in queues and the mean response time at steady-state.

12.1 Stochastic Petri Nets

Since transitions represent activities that change the state (marking) of the net, it is natural to associate a duration with transitions.

In order to solve conflicts among transitions, two alternatives have been proposed: either a "timed firing" of transitions in three phases (which changes the firing rule of Petri nets introducing a timed phase in which the transition is "working" after having removed tokens from the input places and before adding tokens to the output places) or "timed enabling" followed by an atomic firing (which does not affect the usual Petri net firing rule). A more detailed discussion of the timing and firing process can be found in [79]. These different timing interpretations have different implications on the resolution of conflicts.

In [13], free-choice systems were considered. Therefore any conflict can be resolved in a local way by specifying the routing rates of tokens at clusters; thus we are not forced to choose a particular firing mechanism. Now we show that in regular systems any conflict can also be resolved locally.

Lemma 12.1.1 Let (N, M^0) be a regular system, $s \in S$, $|s^\bullet| > 1$, $M \in [M^0\rangle$, $M(s) > 0$. Then for every $t \in s^\bullet$ $\exists \sigma$ such that $M[\sigma\rangle M_1[t\rangle$ and $\forall t_1 \in s^\bullet$ $\#(\sigma|t_1) = 0$.

Proof:

Let N' be an EFC-representation of N. By Lemma 3.1.9a, (N', M) is live and bounded EFC-net. Then by liveness, $\exists \sigma$ such that $M[\sigma\rangle M_1[t\rangle$. Let σ be the firing sequence of minimal length such that $M[\sigma\rangle M_1$. Then $\forall t_1 \in s^\bullet$ $\#(\sigma|t) = 0$ by minimality of σ and the EFC-property.

■

We consider both timed and immediate transitions. For simplicity it is assumed that there do not exist circuits containing only immediate transitions.

For each cluster with more than one transition, we assume that these transitions are immediate (i.e. they fire in zero time). The constants $r_1, ..., r_k \in \mathcal{N}^+$ are explicitly defined in the net interpretation in such a way that when $t_1, ..., t_k$ are enabled,

transition t_i fires with probability

$$r / \sum_{j=1}^{k} r_j$$

Taking into consideration all structural conflict transition sets, the system of equations for any set of transitions in a structural conflict describing the ratio of firing frequences between the transitions using associated *routing rates* can be represented straightforward in a matrix form as: $R \cdot \nu = 0$. We will call R the *Routing Rates matrix.*

A *stochastic Petri net* is a sixtuple $(S, T, F, M^0, \Theta, R)$, where $\Theta : T \to \mathcal{T}$ is a function assigning the delays to transitions. Replacing the triple (S, T, F) by N, we get shorter description: (N, M^0, Θ, R).

In the case of stochastic Petri nets, the ergodicity condition can be considered for the two most important associated stochastic processes: the marking process and the firing process.

Theorem 12.1.2 [19] Let (N, M^0) be a regular system. Then it has a home marking.

■

Using the previous result we prove the next theorem.

Theorem 12.1.3 Let (N, M^0, Θ, R) be a stochastic regular system. Then, both the marking and the firing processes of (N, M^0) are weakly ergodic.

Proof:

For regular systems, the existence of home marking is assured (Theorem 12.1.2). Then after a possible transient phase, the system state is always trapped in a unique strongly connected finite subset of the state space. Thus, the marking and firing processes are weakly ergodic.

■

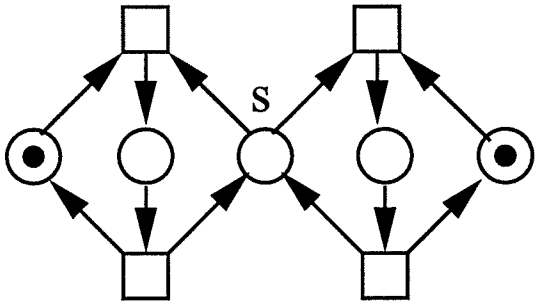

Fig. 12.1 A Regular Petri net for which the Visit Ratios Depend on Initial Marking.

In other words, for regular systems it makes sense to speak of a unique steady-state behavior and to compute bounds for the performance of this steady-state.

In [13] it was proved that for live and bounded FC-systems the visit ratios depend only on the net structure and matrix R, and do not depend on initial marking and actual delays associated with each transition. We will prove in this section that it is valid for safe regular systems. For general class of regular systems the above conclusions are not valid. An example from [13] is given in Fig. 12.1. If we put two tokens to the place s, this system will behave as two separate systems.

A place $s \in S$ is *implicit* iff $\forall t \in s^{\bullet}\ \forall M \in [M^0\rangle$ if $\forall s_1 \in {}^{\bullet}t \setminus \{s\}\ M(s_1) > 0$ then $M(s) > 0$. That is, the deletion of an implicit place does not affect the language of the system.

Let N be a net with $|S| > 1$. We define the net $N^{-s} = (S \setminus \{s\}, T, F \cap ((S \setminus \{s\}) \times T) \cup (T \times (S \setminus \{s\}))$.

A place $s \in S$ is *bridge place* iff $|\mathcal{A}^{-s}| > |\mathcal{A}|$. That is, after the deletion of a bridge place, the cluster splits into two or more clusters.

Lemma 12.1.4 Let (N, M^0) be a safe regular system and s be a bridge place. Then s is not implicit.

Proof:

By definition of a bridge place there are two transitions t, t_1 such that $\{(s,t),(s,t_1),(s_1,t_1)\} \subset F \cap (S \times T)$ and $(s_1,t) \notin F \cap (S \times T)$.

Since (N, M^0) is live and safe, and there exists the corresponding live and safe EFC-net, there is a marking M when all the places of the cluster are marked. Let $M[t\rangle M'$. Then $M'(s) = 0$ and $M'(s_1) = 1$. Therefore s can not be implicit.

■

12.2 Bounds for the Throughput of Transitions

Throughput ν_i of a transition t_i is the average frequency of firings of t_i. The constant vector $\overline{\sigma}$ is the *limit firing flow vector*, with components $\sigma_i, i = 1, ..., m$. The *mean cycle time* $\overline{\tau}$ has the components $\tau_i = \frac{1}{\sigma_i}$. The vector of *visit ratios*, normalized for transition t^Δ (for having the t^Δ component equal to 1), is a vector $\overline{v}^\Delta$ with components:

$$\nu_i^\Delta = \frac{\sigma_i}{\sigma^\Delta}$$

From the definitions it follows that $\overline{v}^\Delta = \tau^\Delta \cdot \overline{\sigma}$.

Theorem 12.2.1 [13] Let (N, M^0, Θ, R) be a stochastic Petri net. Then a lower bound for the mean interfiring time τ^Δ of transition t^Δ (or its inverse an upper bound for the throughput) can be computed by solving the following linear programming problem:

$\tau^\Delta \geq$ maximum $Y^T \cdot PRE \cdot \overline{\Theta} \cdot \nu^\Delta$

subject to $Y^T \cdot C = 0$, $Y^T \cdot M^0 = 1$, $Y \geq 0$, where $\overline{\Theta}$ is a diagonal matrix with elements of Θ.

■

The throughput upper bound derived from Theorem 12.2.1 is not reachable in general for live and bounded free-choice systems. Several improvements and a reachable bound for the case of live and safe free-choice systems can be found in [12]. Those improvements are also applicable to regular systems.

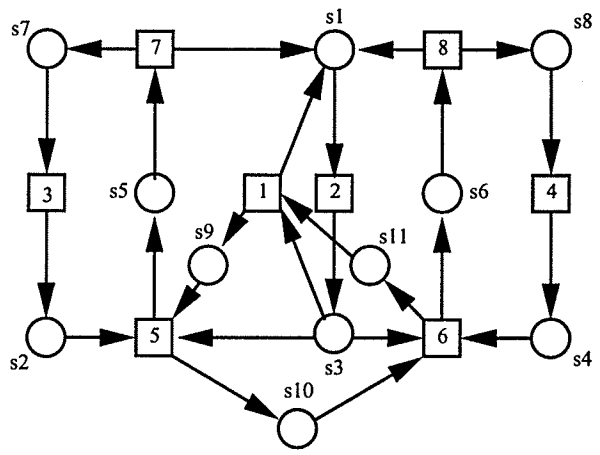

Fig. 12.2 The Net N_R For the System in Fig. 2.1.

To calculate the vector of visit ratios for a stochastic regular Petri net (N, M^0, Θ, R), we define the corresponding non-stochastic Petri net (N_R, M_R^0) which have the same visit ratios.

Let $Q = \{t_1, ..., t_m\} \subseteq T$ be the set of transitions of a cluster (where $t_1, .., t_m$ is an arbitrary fixed order). For every transition t_i of Q, we define a new place s_i. The net $N_Q = (\{s_1, ..., s_m\}, Q, \{(s_1, t_1), (t_1, s_2), ..., (s_m, t_m), (t_m, s_1)\})$ is called a *regulation circuit* of Q.

We denote by

$$N_R = \cup_i N_{Q_i} \cup N$$

the net obtained from N by componentwise union of places, transitions and arcs of N and N_{Q_i} for every cluster i. Note that, after the addition of all regulation circuits to every cluster the resulting net N_R is not regular any more.

Lemma 12.2.2 Let (N, M^0, Θ, R) be a stochastic regular Petri net, and N_R be a net defined above. Then

$$C_R = \begin{pmatrix} C \\ R \end{pmatrix}$$

Example. Consider the system in Fig. 2.1. For simplicity we assume that all routing rates at the conflict are equal to 1. Then the net N_R is shown in Fig. 12.2, and the matrix

$$C_R = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Lemma 12.2.3 [22] Let N be a net and N_R be a net defined above. Given a marking M' of N', define M as the projection of M' on the places of N. Then

(a) If $M_1'[\sigma\rangle M_2'$ is an occurrence sequence of N_R, then $M_1[\sigma\rangle M_2$ is an occurrence sequence of N.

(b) The regulation circuit N_Q is an S-component of N_R.

■

Lemma 12.2.4 [82] Let X be a minimal support of an invariant. Then there exists a unique minimal invariant J such that $\langle J\rangle = X$, and for every invariant J_1 such that $\langle J_1\rangle = X$, there exists an integer k such that $J_1 = kJ$.

■

Lemma 12.2.5 [22] Let (N, M^0) be a bounded system and let $M^0[\sigma\rangle$ be and infinite occurrence sequence.

(a) There exists sequences σ_1, σ_2, σ_3 such that $\sigma = \sigma_1\sigma_2\sigma_3$, σ_2 is not empty sequence and $M^0[\sigma_1\rangle M[\sigma_2\rangle M[\sigma_3\rangle$ for some marking M.

(b) There exists a semi-positive T-invariant J such that $\langle J\rangle \subseteq Pk(\sigma)$.

■

Lemma 12.2.6 Let N be a regular net and N_R be a net defined above. Then N_R is has the only minimal T-invariant J, and J is positive.

Proof:

Let M^0 be a live and bounded marking of N. Choose a marking M_R^0 of N_R which coincides with M^0 on all places of N and marks one place of each regulation circuit.

Claim 1. (N_R, M_R^0) is bounded. By Lemma 3.5.6, N is covered by S-components. Since both the S-components of N and the added regulation circuits are S-components of N_R, the net N_R is also covered by S-components.

Claim 2. (N_R, M_R^0) is deadlock-free. Let M_R be a reachable marking of (N_R, M_R^0), and let M be its restriction to the places of N. By Lemma 12.2.3, M is reachable marking of (N, M^0). We prove that M_R enables some transition.

Since (N, M^0) is live and M is reachable, M enables a transition t. Since N is regular, there exists a marking M' reachable from M which enables every transition of the cluster $[t]$. Since the regulation circuit of the transitions of $[t]$ is an S-component of N_R, the total number of tokens in its set of places remains constant, and so M'_R marks one of its places. This place belongs to the preset of some transition in $[t]$, say t_k. Then M'_R enables t_k, and the claim is proved.

Claim 3. N_R has a semi-positive T-invariant. By Claim 2, there exists an infinite occurrence sequence $M_R^0[\sigma\rangle$. By Lemma 12.2.5, N_R has a semi-positive T-invariant J. J is also a T-invariant of N because the pre- and post-set of a place of N coincides with its pre- and post-set in N_R.

Claim 4. For every semi-positive T-invariant: $[t] = [t_k]$ implies $J(t) = J(t_k)$.

Since $[t] = [t_k]$, there exists a path $ts_1t_1...t_{k-1}s_kt_k$ inside the regulation circuit of the transitions of $[t]$ leading from t to t_k. Since J is a T-invariant of N_R, and the places $s_1, ..., s_k$ have exactly one input and one output transition, we have

$J(t) = J(t_1) = ... = J(t_{k-1}) = J(t_k)$

Claim 5. Every semi-positive T-invariant of N_R is positive. Let t be a transition of $\langle J \rangle$. Since N_R is strongly connected, t has an output place s in N_R. Since J is a semi-positive T-invariant, we have ${}^\bullet\langle J \rangle = \langle J \rangle^\bullet$, and therefore $s^\bullet$ contains a transition t_k of $\langle J \rangle$. By Claim 4, $\langle J \rangle$ includes every transition of the cluster $[t_k]$. Since N is regular, this set is $[s]$. Hence, $(\langle J \rangle^\bullet)^\bullet \subseteq \langle J \rangle$. Since $\langle J \rangle$ is non-empty and N_R is strongly connected, the set $\langle J \rangle$ contains all transitions of N_R, which implies that J is positive.

By Claim 3, there is a semi-positive T-invariant of N_R. By Claim 5, it is positive. By Lemma 12.2.4, the result follows.

■

To calculate the vector of visit ratios for a stochastic regular Petri net we use the following theorem.

Theorem 12.2.7 Let (N, M^0, Θ, R) be a stochastic safe regular Petri net. Then, for every live and safe marking of N, the vector of visit ratios $\overline{v}^\triangle$ depends only on N and R, is positive, and is the one and only one solution of the system of linear equations:

$$\begin{pmatrix} C \\ R \end{pmatrix} \overline{v}^\triangle = \overline{0}, \qquad \nu^\triangle = 1 \tag{12.1}$$

Proof:

By Lemma 12.1.4, the system (N, M^0) does not have implicit bridge places and we can use the net N_R for calculation of $\overline{v}^\triangle$. Let (N_R, M_R^0) be the net defined above. By

Lemma 12.2.6, N_R has the only minimal T-invariant J, and J is positive. By Lemma 12.2.2, $C_R = \begin{pmatrix} C \\ R \end{pmatrix}$, and equation $C_R \cdot J = \overline{0}$ is equivalent to:

i) $C \cdot J = \overline{0}$ (i.e. J is a T-invariant of C)

ii) $R \cdot J = \overline{0}$ (i.e. the routing rates are respected)

Let $\overline{\sigma}$ be a Parikh map of a reproductive firing sequence of (N_R, M_R^0). Obviously $\overline{\sigma}$ is a semi-positive T-invariant of N_R. By Lemma 12.2.4, $\overline{\sigma} = kJ$. Then $\frac{\sigma_i}{\sigma^\Delta} = \frac{J_i}{J^\Delta}$ and $\overline{\nu}^\Delta = \frac{J}{J^\Delta}$.

Since every place of the regulation circuit is linearly independent from the places of N, we have $rank(C_R) = rank(C) + rank(R) = |\mathcal{A}| - 1 + |T| - |\mathcal{A}| = |T| - 1$. So the system 12.1 has one and only one solution.

■

Example. For the system above, and for $t^\Delta = t_1$, $\overline{\nu}^\Delta = (1,3,1,1,1,1,1,1)^T$. Then the lower bound of the mean cycle time $\tau_l^\Delta = max(3\theta_2 + \theta_7 + \theta_8, \theta_3 + \theta_7, \theta_4 + \theta_8)$, and the upper bound of the throughput $\overline{\sigma}_u = (\frac{1}{\tau_l^\Delta}, \frac{3}{\tau_l^\Delta}, \frac{1}{\tau_l^\Delta}, \frac{1}{\tau_l^\Delta}, \frac{1}{\tau_l^\Delta}, \frac{1}{\tau_l^\Delta}, \frac{1}{\tau_l^\Delta}, \frac{1}{\tau_l^\Delta})^T$. Note that $\theta_1 = \theta_5 = \theta_6 = 0$, because transitions in a conflict should be immediate as agreed above.

Theorem 12.2.7 was first proved in [13] for the class of free-choice nets. In [12] this result was generalized to *FRT-nets* (*nets with freely related T-semiflows*). The class of regular nets is not included into the class of FRT-nets (the regular net in Fig. 2.1 is not an FRT-net). To check this statement, and for the complete definition of FRT-nets (quite involved), we direct the reader to [12].

Let (N, M^0) be a system and $t \in T, E(t) = max\{k | \exists M \in R(N, M^0) : M \geq kPre(t)\}$ is an *Enabling bound.*

$L(t) = max\{k | \forall M_1 \in R(N, M^0), \exists M \in R(N, M^1) : M \geq kPre(t)\}$ is a *liveness bound.*

$SE(t) =$ maximize k subject to $M = M^0 + C \times \sigma \geq kPre(t), \sigma \geq 0$ is a *Structural bound.*

Lemma 12.2.8 [13] Let (N, M^0) be a system. Then

(a) $\forall t \in T, SE(t) \geq E(t) \geq L(t)$

(b) if (N, M^0) is reversible, then $E(t) = L(t)$

■

Lemma 12.2.9 If (N, M^0) is a live and bounded system, such that for every transition t_i there is an S-component S_i with $M^0(S_i) = 1$ then $SE(t) = E(t) = L(t) = 1$.

Proof:

Obvious.

■

For T-systems, it makes no sense to speak about R matrix, because there are no conflicts there. So the definition of stochastic T-systems is shorter: (N, M^0, Θ).

Theorem 12.2.10 [13] Let (N, M^0, Θ) be a stochastic live and bounded T-system, and let the mean firing times θ_i for each transition be t_i. It is not possible to assign PDF's to the transition firing times such that the average cycle time is greater than

$$\tau^u = \sum_{j=1}^{m} \frac{\theta_j}{SE(t_j)}$$

independently of the net structure. Moreover this upper bound is reachable for any T-system structure and for any assignment of PDF's to the firing delay of transitions (i.e. the bound cannot be improved).

■

Theorem 12.2.11 Let (N, M^0, Θ) be a stochastic live and bounded T-system such that for every transition there is a loop with the only token on it. It is not

possible to assign PDF's to the transition firing times such that the average cycle time is greater than

$$\tau^u = \sum_{j=1}^{m} \theta_j$$

independently of the net structure. Moreover this upper bound is reachable for any T-system structure and for any assignment of PDF's to the firing delay of transitions (i.e. the bound cannot be improved).

Proof:

The proof follows from Lemma 12.2.9 and Theorem 12.2.10.

■

To calculate the lower bound for the throughput we use the following theorem.

Theorem 12.2.12 Let (N, M^0, Θ, R) be a stochastic safe regular system. Then the upper bound for the average cycle time of transition $t^\triangle$ is

$$\tau_u^\triangle = \sum_{j=1}^{m} \nu^\triangle(t_j)\theta_j$$

■

Example. For the system above the upper bound of mean cycle time $\tau_u^\triangle = 3\theta_2 + \theta_3 + \theta_4 + \theta_7 + \theta_8$, and the lower bound of the throughput is $\overline{\sigma}_l = (\frac{1}{\tau_u^\triangle}, \frac{3}{\tau_u^\triangle}, \frac{1}{\tau_u^\triangle}, \frac{1}{\tau_u^\triangle}, \frac{1}{\tau_u^\triangle}, \frac{1}{\tau_u^\triangle}, \frac{1}{\tau_u^\triangle}, \frac{1}{\tau_u^\triangle})^T$.

12.3 Bounds for the Mean Length of Queues

To compute the lower bound for the mean length of queues we generalize the results from [12]: $\overline{M}^l = PRE \cdot \overline{\Theta} \cdot \overline{\sigma}^l$, where $\overline{\sigma}^l$ is a lower bound for the throughput vector (i.e., $\overline{\sigma}^l(t_i) = 1/\tau_u(t_i), i = 1, ..., m$, with $\tau_u(t_i)$ being the upper bound for the mean cycle time of t_i computed in Theorem 12.2.12).

For the computation of an upper bound for the mean marking of a given place s let us consider an S-invariant I, whose support includes this place.

We have $Y^T \cdot M^0 = Y^T \cdot \overline{M}$.

Hence $\overline{M}(s) \leq \overline{M}^l(s) + \frac{1}{I(s)} \cdot I^T \cdot (M^0 - \overline{M}^l)$ and the same condition holds for each S-invariant including place s. Then, the computation of an upper bound for the mean marking of places can be formulated in terms of an LPP as follows:

$\overline{M}^u(s) = min\ \overline{M}^l(s) + Y^T \cdot (M^0 - \overline{M}^l)$ subject to $Y^T \cdot C = 0$; $Y^T \cdot e_s = 1$; $Y \geq 0$, where e_s is the characteristic vector of s (i.e. $e_s(s) = 1$, and for any other place $s' \neq s$: $e_s(s') = 0$) the restriction $Y^T \cdot e_s = 1$ allows us to omit the denominator $I(s)$ which is assumed to be non-zero value.

Now the computation of the pmf of the number of tokens at steady-state in a place s is straightforward. Since the system is safe, $p(M(s) = 1) = \overline{M}(s)$ and $p(M(s) = 0) = 1 - \overline{M}(s)$.

12.4 Bounds for the Mean Response Time at Places

The mean response time $\overline{R}(s)$ at a place s is the mean value of the sojourn time of a token in this place (i.e., sum of waiting plus service time). From the knowledge of upper and lower bounds for the throughput of transitions and for the mean marking of places, and applying Little's law, upper and lower bounds for the mean response time at places can deduced as follows:

$\overline{R}^u(s) = \frac{\overline{M}^u(s)}{PRE(s) \cdot \overline{\sigma}^l}$ and

$\overline{R}^l(s) = \frac{PRE(s) \cdot \overline{\Theta} \cdot \nu^{\Delta}}{PRE(s) \cdot \nu^{\Delta}}$.

To confirm the theoretical results given in the chapter we give the unfolding of our system from Fig. 2.1. The unfolding is presented in Fig. 12.3.

12.5 Summary

In [46] we propose polynomial algorithms for performance evaluation of communication networks using stochastic Petri nets. We used theoretical results and polynomial

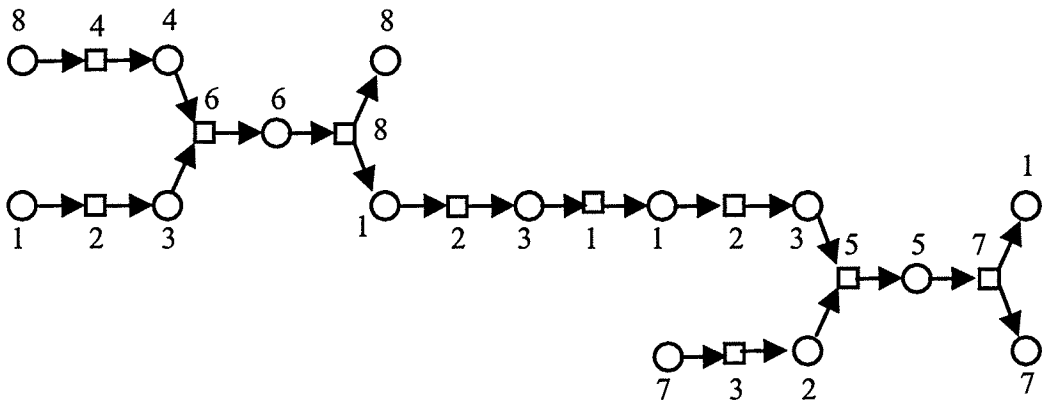

Fig. 12.3 The unfolding of the system from Fig. 2.1.

algorithms from [13, 19, 20, 57] for quantitative analysis of stochastic Petri nets.

Our major contribution is a generalization of results from [13] obtained for free-choice Petri nets to regular Petri nets. There was already a generalization to FRT-nets [12]. It can be shown that regular Petri nets are not a subclass of FRT-nets. We proved that in regular Petri nets, conflicts can be resolved locally, similar to free-choice Petri nets and extended free-choice Petri nets.

Chapter 13

Conclusions and Future Research

13.1 Contributions of this Thesis

In this work we described the solution of some problems of verification and performance evaluation of concurrent systems. We studied Petri nets and their derivatives: Signal Transition Graphs, Logic Control Algorithms, stochastic Petri nets. Communication and interaction in concurrent systems are complex and difficult to specify, implement and test. Therefore formal approaches are needed for the design, synthesis, verification and implementation of concurrent systems.

We applied our results in four fields:

1) Asynchronous Circuits Analysis and Synthesis.

Signal Transition Graphs is a model of speed-independent circuits. Several synthesis techniques for Signal Transition Graphs have been proposed which require knowledge of the concurrency relation for the corresponding Petri net. We use some results on Petri nets to derive an efficient polynomial algorithm for the computation of the concurrency relation on free-choice Signal Transition Graphs. This result can be generalized to a broader class of Signal Transition Graphs.

2) Verification of Logical Control Algorithms.

In the recent years a lot of investigation have been made in the field of asynchronous interactions of concurrent processes. One of the reasons is the problems of logical control in automated production. One way to solve the problems is to reflect the concurrency of control objects in control algorithms.

We develop a number of techniques to verify the algorithm correctness: reduction, graph-theoretic analysis, Rank Theorem and unfoldings.

3) Verification of workflow procedures. A workflow Petri net is a model of workflow procedure. For verification of correctness of workflow nets the soundness property is introduced. The soundness is a set of behavioral properties of Petri nets. Using the Petri net theory, linear algebra and graph theory, new results have been obtained which facilitate the analysis of workflow nets. For some analysis problems polynomial algorithms have been obtained.

4) Performance evaluation of communication networks. In [46] we propose polynomial algorithms for performance evaluation of communication networks using stochastic Petri nets. We used theoretical results and polynomial algorithms from [13, 19, 20, 57] for quantitative analysis of stochastic Petri nets. Our major contribution is a generalization of results from [13] obtained for free-choice Petri nets to regular Petri nets.

13.2 Future Work

- We are going to develop the reduction method (based on simple local rules) of analysis and refinement (synthesis) of EFC-nets. Such rules are necessary for the construction of more complex systems with a correct behaviour. Furthermore, they are interesting for theoretical considerations. Together with completeness results, which state that each system with some behaviour properties can be reduced, they provide the possibility of proving new results using inductive arguments.

- We are going to generalize the method of concurrency relation computation to the class of state machine decomposable Petri nets.

- We are going to find a polynomial algorithm for the reachability problem in live and bounded (live and safe) EFC-nets. There exists already a polynomial algorithm to decide reachability problem for cyclic live and bounded extended free-choice nets. But the problem for all live and bounded extended free-choice nets is still open.

- We are going to generalize Siphon/Trap analysis for different subclasses of Petri Nets.

- We are going to generalize the methods of modular synthesis to the class of state machine decomposable Petri nets.

- We are going to generalize the Performance Evaluation method to the class of state machine decomposable Petri nets.

Bibliography

[1] W.M.P. van der Aalst, "Verification of Workflow Nets", In P.Azema and G.Balbo, editors, *Proc. 18th Int. Conf. on Appl. and Theory of Petri Nets*, Lecture Notes in Computer Science, Vol. 1248, Springer-Verlag, 1997, pp. 407-426.

[2] M.Ajmone Marsan, "Stochastic Petri nets: an elementary introduction", In G.Rozenberg, editor, *Advances in Petri nets 1989.* Lecture Notes in Computer Science, Vol. 424, Springer-Verlag, 1990, pp. 1-29.

[3] P.Alimonti, E.Feuerstein, U.Nanni, "Linear time algorithms for Liveness and Boundedness in Conflict-free Petri nets". *Proc. of 1-st Latin American Symposium on Theoretical Informatics*, Lecture Notes in Computer Science, Vol. 583, Springer-Verlag, 1992.

[4] M.-L. Benalycherif, C.Girault: "Behavioral and Structural Composition Rules Preserving Liveness by Synchronization for Coloured FIFO Nets". *Proc. 17th Int. Conf. on Application and Theory of Petri nets, Osaka, Japan*, Lecture Notes in Computer Science, Vol. 1091, Springer-Verlag, 1996, pp. 73-92.

[5] E.Best, J.Desel, "Partial order behavior and structure of Petri nets". *Formal Aspects of Computing*, Vol.2 No.2, 1990, pp. 123-138.

[6] E.Best, R.Devillers: "Concurrent Behaviour: Sequences, Processes and Programming Languages". GMD-Studien Nr.99, 1985, Sankt Augustin.

[7] E.Best, K.Voss: "Free Choice Systems Have Home States". *Acta Informatica 21*, 1984, pp. 89-100.

[8] E.Best: "Structure Theory of Petri Nets: the Free Choice Hiatus", *Advances in Petri Nets 1986*, Lecture Notes in Computer Science, Vol. 254, Springer-Verlag, pp. 168-205.

[9] E.Best, C.Fernández: "Non-sequential processes. A Petri net view". EATCS Monographs on Theoretical Computer Science, 13, 1988.

[10] G.Berthelot, G.Roucairol, R.Valk: "Reduction of Nets and Parallel Programs". *Net Theory and Applications*, Lecture Notes in Computer Science, Vol. 84, Springer-Verlag, 1980, pp. 277-290.

[11] G.Berthelot: "Checking Properties of Nets Using Transformations". Lecture Notes in Computer Science, Vol. 222, Springer-Verlag, 1986, pp. 19-40.

[12] J.Campos, "Performance Bounds for Synchronized Queuing Networks", PhD thesis, Departamento de Ingenieria Electrica e Informatica, Universidad de Zaragoza, Spain, December 1990. Research Report GISI-RR-90-20.

[13] J.Campos, G.Chiola, M.Silva, "Properties and Performance Bounds for Closed Free Choice Synchronized Monoclass Queuing Networks", *IEEE Trans. on Automatic Control*, Vol. AC-36 No. 12, 1991, pp. 1368-1382.

[14] A. Cheng, J. Esparza, and J. Palsberg. "Complexity Results for 1-safe Nets". *Theoretical Computer Science*, Vol. 147, No.1-2, 1995, pp. 117-136.

[15] T.A. Chu. "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications". Phd thesis, MIT, 1987.

[16] P.Civera, G.Conte, D.Del Corso, F.Maddaleno, "Petri net models for the Description and verification of parallel bus protocol", *Computer Hardware Descrip-*

tion Languages and their Applications, M.R.Barbacci and C.J.Koomen (Eds.), Elsevier (North-Holland), 1987, pp. 309-326.

[17] F.Commoner: "Deadlocks in Petri nets". Wakefield. Applied Data Rsearch, Inc., Report ♯CA-7206-2311, 1972.

[18] F.Commoner, A.W.Holt, S.Even, A.Pnueli: "Marked Directed Graphs." *J.Comput. Syst. Sci.*, Vol.5, 1971, pp. 511-523.

[19] J.Desel: "Regular marked Petri nets". *Proc. of the 19th Int. Workshop on Graph-Theoretic Concepts in Computer Science/* Jan van Leeuween (ed.) Lecture Notes in Computer Science, Vol. 790, Springer-Verlag, 1993, pp. 264-275.

[20] J.Desel, "A proof of the Rank Theorem for extended free choice nets", *Proc. 13th Int. Conf. on Appl. and Theory of Petri nets*, Lecture Notes in Computer Science, Vol. 616, Springer-Verlag, 1992, pp. 299-309.

[21] J.Desel: "Reduction and Design of Well-behaved Concurrent Systems", in CONCUR'90, Baeten, Klop eds., Lecture Notes in Computer Science, Vol. 458, Springer-Verlag, 1990, pp. 166-181.

[22] J.Desel, J.Esparza, *Free Choice Petri Nets*, Cambridge University Press, 1995.

[23] J.Desel,J.Esparza: "Reachability in cyclic extended free choice systems". *Theoretical Computer Science* Vol. 114, 1993, pp. 93-118.

[24] J.Esparza, "Synthesis rules for Petri nets, and how they lead to new results", *Proc. CONCUR'90*, Lecture Notes in Computer Science, Vol. 458, Springer-Verlag, 1990, pp. 182-198.

[25] J. Esparza. "Reduction and Synthesis of Live and Bounded Free Choice Petri Nets". *Information and Computation.* Vol.114. No.1. 1994, pp. 50-87.

[26] J. Esparza. "A Solution to the Covering Problem for 1-Bounded Conflict-Free Petri Nets Using Linear Programming". *Information Processing Letters*, Vol. 41, 1992, pp. 313-319.

[27] J. Esparza. "Model checking using net unfoldings". *Science of Computer programming*, 23, 1994. pp. 151-195.

[28] J.Esparza, E.Best, M.Silva, "Minimal deadlocks in free choice nets", Hildesheimer Informatik-Berichte. Universität Hildesheim. No 1. 1989.

[29] J.Esparza, M.Silva: "Modular Synthesis of free-choice nets". Departamento de Ingenieria Electrica e Informatica, Universidad de Zaragoza, Research Report GISI 90.06, March (29 pages).

[30] J.Esparza, M.Silva: "Circuits, Handles, Bridges and Nets". *Advances in Petri Nets.* 1990. G.Rosenberg (Ed.). Lecture Notes in Computer Science, Vol. 483. Springer-Verlag, 1991, pp. 210-242.

[31] J.Esparza, M.Silva "On the Analysis and Synthesis of Free Choice Systems". *Advances in Petri Nets 1990.* G.Rosenberg (Ed.). Lecture Notes in Computer Science, Vol. 483, Springer-Verlag, 1991, pp. 243-286.

[32] O.Fergemand, A.Olsen. *Introduction to SDL-92.* Computer Networks and ISDN Systems, Vol. 26, 1994, pp. 1143-1167.

[33] H.J.Genrich, K.Lautenbach. "System modeling with high-level Petri nets". *Theoretical Computer Science* Vol. 13, 1981, pp. 109-136.

[34] M.T.Hack, *Analysis of production schemata by Petri nets*, TR-94, MIT, Cambridge, MA, 1972, Corrections 1974.

[35] C.A.R.Hoare: "Communicating Sequential Processing", Prentice/Hall, London, 1985.

[36] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Volumes 1-3. Monographs in Theoretical Computer Science, Springer-Verlag, 2nd corrected printing, 1997.

[37] N.D.Jones, L.H.Landweber, Y.E.Lien. "Complexity of some problem in Petri nets". *Theoretical Computer Science* Vol. 4, 1977, pp. 277-299.

[38] R.Johnsonbaugh, T.Murata: "Additional Methods for Reduction and Expansion of Marked Graphs". *IEEE Trans. Circuits and Systems*, Vol.CAS-28, 1981, no.10, pp. 1009-1014.

[39] R.M.Keller: "A Fundamental Theorem of Asynchronous Parallel Computation". *Parallel Processing*, Lecture Notes in Computer Science, Vol. 24, Springer-Verlag, 1975, pp. 102-112.

[40] P.Kemper: "On Finding a cover of minimal siphons in extended free choice nets". *Algorithmen und Werkzeuge für Petrinetze.* Workshop der GI-Fachgruppe 0.0.1 "Petrinetze und verwandte Systemmodelle". Berlin, 10-11, Oktober 1994.

[41] P.Kemper: "$O(|P| \times |T|)$-algorithm to compute a cover of S-components in EFC-nets". Forschungsbericht 543, Universitaet Dortmund, 1994.

[42] P.Kemper: "Linear time algorithm to find a minimal deadlock in a strongly connected free-choice nets". *Proc. of the 14th Int. Conf. on Appl. and Theory of Petri nets.* Chicago, Lecture Notes in Computer Science, Vol. 691, Springer-Verlag, 1993, pp. 319-338.

[43] P.Kemper, F.Bause: "An efficient polynomial-time algorithm to decide Liveness and Boundedness of free-choice nets". *Proc. of the 13th Int. Conf. on Appl. and Theory of Petri nets.* Sheffield, Lecture Notes in Computer Science, Vol. 616. Springer-Verlag, 1992, pp. 263-278.

[44] Pastor, E., Cortadella, J., Kondratyev, A., Roig, O. " Structural methods for the synthesis of speed-independent circuits. ". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 11, 1998, pages 1108-1129.

[45] A.Kovalyov, "Efficient Methods for Verification of Distributed and Concurrent Systems by Petri Nets", *Formal Methods for Open Object-Based Distributed Systems, Stanford University*, Stanford, California, USA. Sept. 2000.

[46] A.Kovalyov, R.McLeod, O.Kovalyov, "Performance Evaluation of Communication Networks by Stochastic Regular Petri Nets", *The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000)*, USA. 2000, Vol. 4, pp. 1991-1998.

[47] A.Kovalyov,O.Kovalyov, "An Efficient Modular Synthesis of Regular Petri Nets by Simple Composition Rules", *the 4th World Multiconference on Systemics, Cybernetics and Informatics SCI'2000.* USA. 2000, Vol. 8, pp. 655-660.

[48] A.Kovalyov, "A Polynomial Algorithm to Compute the Concurrency Relation of Regular Signal Transition Graphs", in: Hardware Design and Petri Nets (A.Yakovlev, L.Gomes, L.Lavagno eds.), Kluwer Academic Publishers, March 2000. pp. 107-126.

[49] A.Kovalyov, "A Polynomial Algorithm to Compute the Concurrency Relation of Regular Signal Transition Graphs", *20th International Conference on Application and Theory of Petri Nets (ICATPN'99)*, Williamsburg, Virginia, USA. 1999, pp. 15-34.

[50] A.Kovalyov, "A Concise Proof of the Coverability Theorem for Live and Bounded Extended Free Choice Nets", *Petri Nets Newsletter.* No.56, 1999, pp. 1-6.

[51] A.Kovalyov and R.McLeod, "New Rank Theorems for Petri Nets and their Application to Workflow Management", *1998 IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, California, USA. 1998, pp. 226-231.

[52] A.Kovalyov and R.McLeod, "Strongly Connected Free-Choice Systems Having Nondead Home Markings are Live and Bounded", *Petri Nets Newsletter*. No.54. 1998, pp. 16-18.

[53] A.Kovalyov, R.McLeod and B.Rahardjo, "A Simple Co-design Approach for Embedded Systems Using SDL", *Proc. Micronet Annual Workshop*, Ottawa, Ontario, p.67, 1997.

[54] A.Kovalyov, "An $O(|S| \times |T|)$-Algorithm to Verify Liveness and Boundedness in Extended Free Choice Nets", *Int. J. of Intelligent Control and Systems*, Vol.1, No.3, 1997, pp. 401-406.

[55] A.Kovalyov and J.Esparza, "A polynomial algorithm to compute the concurrency relation of Signal Transition Graphs", *Proc. 3rd Workshop on Discrete Event Systems (WODES'96), Edinburgh, Scotland, UK*, August 1996.

[56] A.Kovalyov, "New Formulations of the Rank Theorem for Live and Bounded Extended Free Choice Nets". *International Conference on Robotics and Automation*. 1996. USA.

[57] A.Kovalyov, "An $O(|S| \times |T|)$-Algorithm to Verify if a Net is Regular", *Proc. 17th Int. Conf. on Application and Theory of Petri nets, Osaka, Japan*, Lecture Notes in Computer Science, Vol. 1091, Springer-Verlag, 1996, pp. 366-379.

[58] A.Kovalyov, "A Simulation Method of Petri nets by Conflict-Free Ones", *Proc. 1995 INRIA/IEEE Symp. on Emerging Technologies and Factory Automation, Paris, France*, Vol.1, October 10-13, 1995, pp. 583-591.

[59] A.Kovalyov, "An $O(|S| \times |T|)$-Algorithm to Verify Liveness and Boundedness in Extended Free Choice Nets", *Proc. 10-th IEEE Int. Symp. On Intelligent Control, Monterey, California, USA*, August 27-29, 1995, pp. 597-601.

[60] A.Kovalyov, "Good Behaviour in Extended Free Choice Nets", Technical Report FBI-HH-B-176/95, Fachbereich Informatik, Universität Hamburg, Germany", May 1995. 22p.

[61] A.Kovalyov, "A polynomial-time algorithm for Diagnostics of Extended Free Choice Nets", *Proc. 3rd Int. Conf. on Automation, Robotics and Computer Vision (ICARCV'94), Republic of Singapore*, November 1994, pp. 2149-2153.

[62] A.Kovalyov, "On behaviour equivalence of two classes of safe Petri Nets: Persistent and Conflict-free Nets", *Proc. 3rd Int. Conf. on Automation, Robotics and Computer Vision (ICARCV'94), Republic of Singapore*, November 1994, pp. 2135-2138.

[63] A.Kovalyov, "Concurrency Relations and the Safety Problem for Petri Nets", *Proc. 13th Int. Conf. on Application and Theory of Petri nets, Sheffield, UK*, Lecture Notes in Computer Science, Vol. 616, Springer-Verlag, 1992, pp. 299-309.

[64] A.Kovalyov, "On Complete Reducibility of Some Classes of Petri Nets", *Proc. 11th Int. Conf. on Application and Theory of Petri nets, Paris*, June 1990, pp. 352-366.

[65] A.Kovalyov, "On Finding the Parallelism Relation on the Set of Places for Some Subclasses of Petri Nets" *Problems of theoretical Cybernetics. Abstracts of papers of the VIII USSR Conference*, July 1988. Gorkiy, USSR, Part 1. pp. 156-157, in Russian.

[66] A.Kovalyov, "On Finding the Parallelism Relation on the Set of Places for some Subclasses of Petri Nets" Preprint. The Institute of Engineering Cybernetics of Belarus Academy of Sciences. 1987, in Russian.

[67] A.Kovalyov, "On Finding the Parallelism Relation on the Set of Places for a Subclass of Petri Nets", *Vesci Akad. Navuk BSSR*, Ser. F.-M. Navuk, 1989, No.2, pp. 106-110, in Russian.

[68] A.Kovalyov, "Diagnostics of One Class of Algorithms for Logical Control", *Vesci Akad. Navuk BSSR*, Ser. F.-M. Navuk, 1989. No.5, 96-101, in Russian.

[69] A.Kovalyov, "Diagnostics and Checking of Correctness of Concurrent Algorithms for Logical Control", *Design of Discrete Systems*, Minsk, 1989, pp. 12-21, in Russian.

[70] A.Kovalyov, U.Pottosin, "To Decomposition of Concurrent Algorithms for Logical Control", *Automatics and Computers*, 1988. No 1. pp. 8-13, in Russian.

[71] A.Kovalyov, "On Verification of Algorithms for Logical Control", Abstracts of Papers of USSR Scientific Conference *Theoretical and Applied Problems of Design of Technological Processes Control Systems.* Chelyabinsk, 1990, p.110, in Russian.

[72] A.Zakrevskii, Y.Novikov, M.Gurskaya, V.Vasilenok, A.Kovalyov, "Design of Algorithms for Logical Control", *Modeling of Information Systems. Abstracts of papers of USSR Conference*, Novosibirsk, Computer centre of Siberian Department of Sciences Academy of the USSR, 1988. P.248-250, in Russian.

[73] A.Kovalyov, "On Verification of Algorithms for Logical Control by means of Petri Nets". Preprint. The Institute of Engineering Cybernetics of Belarus Academy of Sciences. 1990, in Russian.

[74] A.Kovalyov, "Concurrency Relations and the Safety Problem for Petri Nets". Automation of Logical Design of Discrete System. Minsk. 1991, pp. 28-37, in Russian.

[75] L. Lavagno and A. Sangiovanni-Vincentelli. Algorithms for synthesis and testing of asynchronous circuits. Kluwer Academic Publishers, 1993.

[76] L.H. Landweber, E.L.Robertson: "Properties of conflict free and Persistent Petri nets". *J.ACM*, Vol.25, No.3, 1978, pp. 352-364.

[77] P.E. Lauer, P.R.Torrigiani and M.W. Shields. "COSY - a System Specification Language Based on Paths and Processes." *Acta Informatica*, 12, 1979, pp. 109-158.

[78] K. Lautenbach and H. Wedde. "Generating Mechanisms by Restrictions". Lecture Notes in Computer Science, Vol. 45, Springer-Verlag, 1976, pp. 416-422.

[79] M. A. Marsan, G. Balbo, A.Bobbio, G.Chiola, G.Conte, and Cumani. "The effect of execution policies on the semantics and analysis of Stochastic Petri Nets". *IEEE Transactions on Software Engineering*, Vol. 15 No. 7, July 1989, pp. 832-846.

[80] T.H.Y. Meng, editor. "Synchronization Design for Digital Systems". Kluwer Academic Publishers, 1991.

[81] R. Milner. "A Calculus of Communicating Systems". Lecture Notes in Computer Science, Vol. 92, Springer-Verlag, 1980.

[82] G.Memmi, G.Roucairol, "Linear Algebra in Net Theory", *Net Theory and Applications*, Lecture Notes in Computer Science, Vol. 84, Springer-Verlag, 1980, pp. 213-223.

[83] T.Murata. Petri Nets: "Properties, Analysis and Applications". *Proceedings of the IEEE.* Vol.77, No.4, April 1989, pp. 541-580.

[84] J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

[85] C.A. Petri. "Kommunikation mit Automaten". Bonn: Institut f'ur instrumentelle Mathematik, 1962.

[86] E. Pastor and J. Cortadella. "An efficient unique state coding algorithm for signal transition graphs". *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 1993, pp. 174-177.

[87] C.V.Ramamoorthy, G.S. Ho: "Performance evaluation of asynchronous concurrent systems using Petri nets". *IEEE Trans. Software Eng.*, Vol.SE-6/5, 1980, pp. 440-449.

[88] C.Ramchandani: "Analysis of Asynchronous Concurrent Systemsby Timed Petri nets". $TR-120$, MIT-MAC, 1974.

[89] W.Reisig: *Petri Nets.* Springer-Verlag, 1985.

[90] W.Reisig: "Petri nets with individual tokens", *Informatik-Fachberichte 66*, pp.229-249, 1983.

[91] P.H.Starke: "Analyse von Petri-Netz-Modellen". B.G.Teubner Stuttgart 1990 (in German).

[92] J.Sifakis. "Structural Properties of Petri Nets". *Mathematical foundations of computer science 1978.* Springer-Verlag. Lecture Notes in Computer Science, Vol. 64, Springer-Verlag, 1978, pp. 474-483.

[93] I.Suzuki, T.Murata: "A Method for Stepwise Refinements and Abstractions of Petri Nets". *J. Comput. Syst. Sci.*, Vol.27, 1983, No.1, pp. 51-76.

[94] R.Tarjan. "Depth-first search and linear graph algorithms". *SIAM J. Comp.*, Vol.1, p.p.146-160.

[95] R.Valette. "Analysis of Petri nets be stepwise refinements", *Journal of Computer and System Science*, Vol. 18, 1979, pp. 35-46.

[96] R.Valk. "Petri Nets as Token Objects - An Introduction to Elementary Object Nets". *Proc. 19th Int. Conf. on Application and Theory of Petri nets, Lisbon, Portugal*, Lecture Notes in Computer Science, Vol. 1420, Springer-Verlag, 1998, pp. 1-25.

[97] W.Vogler: "Live and Bounded Free Choice Nets have Home States". *Petri Net Newsletter 32*, published by the Gesellschaft für Informatik, Bonn, Germany, 1989, pp. 18-21.

[98] A.V.Yakovlev. "On limitations and extensions of signal transition graph model for designing asynchronous control circuits". *Proc. Int. Conf. on Computer Design (ICCD'92)*, Cambridge, MA, October 1992, IEEE Comp. Society Press, N.Y., pp. 396-400.

[99] A.V.Yakovlev and A.M.Koelmans. "Petri Nets and Digital Hardware Design". *Lectures on Petri Nets II: Applications. Advances in Petri Nets, Lecture Notes in Computer Science*, Vol. 1492, Springer-Verlag, 1998, pp. 154-236.

[100] H. C. Yen. "A Polynomial Time Algorithm to Decide Pairwise Concurrency of Transitions for 1-Bounded Conflict Free Petri Nets". *Information Processing Letters*, Vol. 38, 1991, pp. 71-76.

[101] A.D.Zakrevski. "Concurrent Algorithms for Logical Control, Reports of Sciences Academy of Belarus", 1982, Vol.26, No.12, pp. 1088-1091.