"Alexandru Ioan Cuza" University of Iaşi
Faculty of Computer Science

TECHNICAL REPORT

Modular Analysis of Petri Net Models

(Ph.D. Thesis)

Aurora ŢIPLEA

TR 04-03, December 2004

ISSN 1224-9327

Universitatea "Alexandru Ioan Cuza" Iaşi
Facultatea de Informatică
Str. Berthelot 16, Iaşi 700483, Romania
Tel. +40-232-201090, email: bibl@infoiasi.ro

Modular Analysis of Petri Net Models

Aurora Ţiplea

Iaşi County Council
Iaşi, Romania
E-mail: atiplea@icc.ro

Thesis submitted to the "Al. I. Cuza" University of Iaşi
for the degree of Doctor of Philosophy
in Computer Science

"Al.I.Cuza" University of Iaşi
Department of Computer Science
TR04-03, ISSN 1224-9327

Iaşi, September 24, 2004

Aurora Ţiplea

Iaşi County Council
Iaşi, Romania
E-mail: atiplea@icc.ro

Prof.Dr. Dorel Lucanu, Chair ("Al.I.Cuza" University of Iaşi)
Prof.Dr. Toader Jucan, Supervisor ("Al.I.Cuza" University of Iaşi)
Prof.Dr. Adrian Atanasiu (University of Bucharest)
Prof.Dr. Alexandru Cicortaş (West University of Timişoara)
Prof.Dr. Teodor Toadere ("Babeş-Bolyai" University of Cluj–Napoca)

"Al.I.Cuza" University of Iaşi
Department of Computer Science
TR04-03, ISSN 1224-9327

To my family

Preface

In the last decade a lot of progress has been made in the development of methods for formal analysis and verification. In spite of this progress, many realistic systems are still too large to be handled. Thus, it is important to find techniques that can be used in conjunction with these methods to extend the size of the systems that can be verified. Two such techniques, generally recognized as the only methods can ever scale up to handle industrial-size design and verification, are the *abstraction* and *modularization* which break the task of verifying a large system into several smaller tasks of verifying simpler systems. Modularization exploits the modular structure of a complex system composed of multiple processes running in parallel. Then, an obvious strategy is to derive properties (proofs) of the whole system from partial (local) properties involving (abstractions of) its modules (components). Generally speaking, modular design and verification requires:

- an ability to describe and compose modules with different synchrony assumptions, and at different level of abstraction;
- an ability to decompose verification tasks into subtasks of lower complexity.

In Petri net theory, modular analysis and verification has mainly focused on transition or place refinement and on preserving properties like liveness or boundedness [63, 47, 38, 65, 13, 20, 24, 29, 7, 8, 34, 66, 10, 21, 64, 4, 51, 6, 28, 31]. Vogler's book in 1992 on Petri net modular constructions [66] is a notable progress, trying to survey all the results obtained in this area.

Our thesis focuses on modular analysis of Petri nets too. Unlike the other papers, we deal with different topics. We decompose Petri nets along subset of places and then study and analyze each Petri net component as a *reactive system* [34]. This is because from the point of view of a given component the rest of the system can be viewed as an environment that continuously

interacts with the component. Then, we focus on two main topics that can be described as follows:

1. Decomposing Petri nets into components, can their processes be decomposed correspondingly? Composing Petri nets, can processes of the resulting net be obtained by composing processes of the constituting nets? We give positive answers to both questions and a compositional semantics of Petri nets is then obtained (Chapter 2).

2. Modular analysis, in general, should be accompanied by correctness proof techniques. Such techniques should allow to prove transformation correctness (from some given point of view) or to prove properties of the nets obtained by such transformations. We provide two powerful correctness proof techniques. One of them assures that concurrent semantics (process semantics or partial word semantics) is preserved by our transformations (Chapter 3). We use this technique in order to prove, in a very elegant way, correctness of several Petri net structural transformations (Chapter 4 and 5). The second correctness proof technique is based on temporal logic and abstraction. An abstraction of a Petri net is obtained by decomposing the original net and enriching one of the components by the environment induced by the others. A suitable simulation relation is defined which allows to transfer properties from the abstract net to the original one (Chapter 6). This technique is exemplified on workflow nets (Chapter 7).

A few words about the structure of this thesis are in order.

Chapter 1 fixes the main notation and terminology to be used throughout the thesis. We did not go into details due to the fact that many good textbooks on Petri nets are available nowadays. On the other hand, this field is quite mature, many international journals publish papers on Petri nets, and many international conferences dealing with Petri net topics are taken place every year.

Chapter 2 introduces the concept of a Petri net reactive module [56, 58, 62]. Its semantics is fully investigated. Thus, we establish a process decomposition theorem and, correspondingly, a process composition theorem. These two theorems give a complete description of the process semantics of Petri nets via module decomposition, showing that this semantics is compositional.

Chapter 3 begins by a short exposition of refinement techniques in Petri net theory. We do not exhaust all these method because most of them are

very little related to our topic, or almost none. On the contrary, the chapter concentrates on deriving correctness proof techniques for the replacement operation. Our replacement operation is place-driven, and it consists of just replacing a subnet of a net by another net. Under certain circumstances, by such an operation process and partial word semantics are preserved. We identify precisely two such circumstances and correctness proof techniques are then obtained [56, 62].

Chapter 4 exemplifies the techniques developed in the previous chapter [54, 56, 62]. First, we discuss Pelz's normal form of Petri nets and we show that a very simple and elegant correctness proof can be obtained by using our techniques (the original correctness proof takes many journal pages and uses advanced graph coloring techniques). Then, we prove the correctness of a series of Petri net transformations aimed to produce a more simplified normal form, called the super normal form of Petri nets.

Chapter 5 opens the way to a new correctness proof technique that is to be discussed in the next chapter. Splitting a Petri net into two modules, each of these two modules can be viewed as an abstraction of the original Petri net. However, the abstraction is to rough. A better solution is to enrich one of these two modules by the environment induced by the other one. In this way we get environmental modules that act as abstractions of original Petri nets, and they can be used for validation purposes as it is described in this chapter [56].

Chapter 6 introduces a major correctness proof technique for Petri net modules [56, 57]. The technique is based on a substantial fragment of the temporal logic CTL^*, namely $\forall CTL^*$, and a suitable simulation preorder. In broad terms, the abstraction is similar to the one described in the previous chapter. Preservation results are obtained, and step-fairness constraints are introduced.

Chapter 7 does what Chapter 4 did. It describes an application of the results developed in the previous chapter with respect to workflow nets [59]. It is shown that in some circumstances workflow nets can be decomposed into modules and $\forall CTL^*$-properties can be transferred from modules to the original net.

The thesis is self-contained and unitary, and all chapters follow a natural order to develop our modular analysis technique based on place-driven decompositions. Two major topics are constant invariants of the thesis: semantics and correctness proof techniques.

I would not have finished this thesis without the assistance of various people. First of all I would like to thank my supervisor Toader Jucan who did not lose confidence in me at times when I was struggling to make progress. I am deeply grateful for all his effort. Also, I would like to thank Dumitru Todoroi, my former supervisor, and then to all committee members for carefully reading my thesis. My colleague Geanina Macovei deserves special thanks for fruitful conversations we had.

Last and most I would like to thank my family for its continuous support.

Iaşi, September 24, 2004
Aurora Ţiplea

Contents

Chapter 1

Preliminaries

In this chapter we recall briefly basic concepts and notation in Petri net theory that are to be used in our paper. For further details, the reader is referred to standard textbooks such as [40, 41, 25].

1.1 Petri Nets

We use standard notation on sets, relations and functions. We recall only a very few of them. The set of *integers* is $\mathbf{Z}$, and $\mathbf{N}$ is the set of *positive integers* or *natural numbers*. *Set inclusion* is denoted by $\subseteq$, and the *strict inclusion* is $\subset$. $|A|$ stands for the cardinality of the set A.

If $f : A \to B$ is a function, then its *restriction* to $C \subseteq A$ is denoted by $f|_C$. Given a set of functions from A into B, $X|_S$ stands for

$$X|_S = \{f|_S | f \in X\}.$$

The *extension* of X to D, where $A \subseteq D$, is the set

$$X|^D = \{f : D \to B | f|_A \in X\}.$$

If f and g are functions from A into B and a order relation $\leq$ on B is given, then extend it to functions by $f \leq g$ if and only if $f(a) \leq g(a)$ for all $a \in A$.

For Petri net concepts and notation we follow mainly [25]. A (finite) *Petri net* (or simply, *net*) is any 4-tuple $\Sigma = (S, T, F, W)$, where

1. S and T are two finite sets (of *places* and *transitions*, respectively) such that $S \cap T = \emptyset$;

2. $F \subseteq (S \times T) \cup (T \times S)$ is the *flow relation*;

3. $W : (S \times T) \cup (T \times S) \to \mathbf{N}$ is the *weight function* of Σ verifying $W(x, y) = 0$ iff $(x, y) \notin F$.

When $W(x, y) \leq 1$ for all $(x, y) \in F$, we may (and will) simplify the 4-tuple (S, T, F, W) to the 3-tuple (S, T, F).

In our paper we shall suppose that all nets we consider do not have isolated transitions (but they may have isolated places).

The net $(\emptyset, \emptyset, \emptyset, \emptyset)$ is called the *empty net*.

For $x \in S \cup T$ we set

– ${}^\bullet x = \{y | (y, x) \in F\}$,

– $x^\bullet = \{y | (x, y) \in F\}$, and

– ${}^\bullet x^\bullet = {}^\bullet x \cup x^\bullet$,

and extend usually these notations to subsets $X \subseteq S \cup T$.

A *marking* of a net Σ is any function $M : S \to \mathbf{N}$ (when S is empty, M is the empty function); it will sometimes be identified with a vector $M \in \mathbf{N}^{|S|}$. The marking whose components are all zero, called the *zero marking*, will be denoted by $\underline{0}$ (no matter how large is S).

Operations with markings are operations with functions or operations with vectors due to the fact that we have identified markings with vectors. Therefore, they are component wise defined.

A *marked Petri net* is any pair $\gamma = (\Sigma, M_0)$, where Σ is a Petri net and M_0, the *initial marking* of γ, is a marking of Σ. A λ-*labeled marked Petri net* is a 3-tuple $\gamma = (\Sigma, M_0, l)$, where the first two components form a marked Petri net and l, the λ-*labeling function* of γ, assigns to each transition either a letter or the empty word λ. When λ is not in the range of l we will refer to such triples as *labeled marked Petri nets* and to l as *labeling function*; the notation will be modified correspondingly to labeled marked Petri nets.

In the sequel we shall often use the term "Petri net" or "net" whenever we refer to a structure γ as defined above. In all the above definitions Σ is called the *underlying net* of γ. A marking (place, transition, arc, weight, resp.) of a net γ is any marking (place, transition, arc, weight, resp.) of the underlying net of γ. Two nets γ_1 and γ_2 are called *disjoint* if $(S_1 \cup T_1) \cap (S_2 \cup T_2) = \emptyset$.

Pictorially, a net γ is represented by a graph. Places are denoted by circles and transitions by boxes; the flow relation is represented by arcs. The arc $f \in F$ is labeled by $W(f)$ whenever $W(f) > 1$. The initial marking M_0 is presented by putting $M_0(s)$ tokens into the circle representing the place s and the labeling function is denoted by placing letters into the boxes representing transitions.

Let γ be a net and M a marking of it. The *transition* (*firing*) *rule* states that a transition t is *enabled* at M, denoted $M[t\rangle_\gamma$, if $M(s) \geq W(s,t)$ for all $s \in S$. If t is enabled at M then t may *occur* yielding a new marking M' given by $M'(s) = M(s) - W(s,t) + W(t,s)$, for all $s \in S$; we abbreviate this by $M[t\rangle_\gamma M'$. The transition rule is extended to sequences of transitions $w \in T^*$ in the usual way:

- $M[\lambda\rangle M$, for any marking M;
- if $M[u\rangle M'$ and $M'[t\rangle M''$, for some $u \in T^*$ and markings M' and M'', then $M[ut\rangle M''$.

If $M_0[w\rangle_\gamma M$ then M is called *reachable*; $[M_0\rangle_\gamma$ denotes the set of all reachable markings. The notation "$[\cdot\rangle_\gamma$" will be simplified to "$[\cdot\rangle$" whenever γ is understood from context.

Let Σ be a Petri net and M_0 a marking of it. We say that:

- Σ is *m-bounded with respect to M_0*, where $m \geq 0$ is a natural number, if $M(s) \leq m$ for all reachable markings $M \in [M_0\rangle$ and places s. Σ is *bounded with respect to M_0* if it is m-bounded with respect to M_0 for some m. Σ is *safe with respect to M_0* if it is 1-*bounded with respect to M_0*;
- a transition t of Σ is *quasi-live with respect to M_0* if there exists $M \in [M_0\rangle$ such that $M[t\rangle$. If t is not quasi-live we will also say that t is *dead with respect to M_0*;
- a transition t of Σ is *live with respect to M_0* if for all $M \in [M_0\rangle$ there exists $M' \in [M\rangle$ such that $M'[t\rangle$;
- Σ is *quasi-live* (*live*) *with respect to M_0* if all transitions of Σ are quasi-live (live) with respect to M_0;

- Σ is *deadlock-free with respect to* M_0 if for any reachable marking M from M_0 there exists a transition t such that $M[t\rangle$.

All these concepts extend naturally to marked and/or labeled nets.

1.2 Processes of Petri Nets

The concurrent behavior of Petri nets is well-expressed by the notion of a *process*. Generally speaking, processes of Petri nets are obtained by running nets and solving conflicts in an arbitrary fashion as and when they arise. A process of a net is also a net; these nets are called *occurrence nets* and they are classical nets $N = (B, E, R)$ (B is the set of places, E is the set of transitions, and R is the flow relation) satisfying:

(i) $|{}^\bullet b| \leq 1$ and $|b^\bullet| \leq 1$, for all $b \in B$;

(ii) R^+ is acyclic, i.e. for all $x, y \in B \cup E$ if $(x, y) \in R^+$ then $(y, x) \notin R^+$.

Usually the elements of B are called *conditions* whereas the elements of E are called *events*.

The following concepts are closely related to occurrence nets:

- the *partially ordered set induced* by N is $(B \cup E, \prec_N)$, where $\prec_N = R^+$;
- a *B-cut* of N is any maximal subset $C \subseteq B$ of incomparable elements according with the relation $\prec_N$. As we will only use B-cuts we call them shortly *cuts* (see [9] for more details);
- the *initial* (*final*, respectively) *cut* of N is ${}^\circ N = \{b \in B || {}^\bullet b| = 0\}$ ($N^\circ = \{b \in B || b^\bullet| = 0\}$, respectively).

In defining processes we need *V-labeled occurrence nets* which are couples $\pi = (N, p)$, where N is an occurrence net and p is a total function from $B \cup E$ into an alphabet V. The above definitions (partial order, cut, initial and final cut) are transferred to labeled occurrence nets π by means of N; the corresponding notations are obtained by changing "N" into "π" (e.g. $\prec_\pi$, ${}^\circ\pi$, π°).

Let $\Sigma = (S, T, F, W)$ be a Petri net, $\pi = (N, p)$ an $(S \cup T)$-labeled occurrence net such that $p(B) \subseteq S$ and $p(E) \subseteq T$, and C a subset of conditions of π. Define the *marking induced by C in Σ* as being

$$M_C(s) = |p^{-1}(s) \cap C|,$$

for all $s \in S$.

There are two alternative definitions of a process, *axiomatic* and *inductive*, and it is well-known that for Petri nets of finite synchronization they yields exactly the same objects [9]. We adopt here the axiomatic definition (the inductive one will be presented in Section 4.1 in terms of Petri net composition).

A *process* of $\gamma = (\Sigma, M_0)$ is any $(S \cup T)$-labeled occurrence net $\pi = (N, \varphi)$ satisfying:

(i) $\varphi(B) \subseteq S$, $\varphi(E) \subseteq T$;

(ii) $M_0(s) = |\varphi^{-1}(s) \cap {}^oN|$ for all $s \in S$;

(iii) $W(s, \varphi(e)) = |\varphi^{-1}(s) \cap {}^\bullet e|$ and $W(\varphi(e), s) = |\varphi^{-1}(s) \cap e^\bullet|$ for all $e \in E$ and $s \in S$.

In order to obtain processes of λ-labeled nets $\gamma = (\Sigma, M_0, l)$ what we have to do is to consider each process $\pi = (N, p)$ of (Σ, M_0) and to replace the function p by p', where

$$p'(x) = \begin{cases} p(x), & \text{for all the conditions } x \\ (l \circ p)(x), & \text{for all the events } x. \end{cases}$$

That is, the events will be labeled by $l \circ p$. From this reason we will sometimes use $l \circ p$ instead of p' (with the above meaning). The set of all processes of a net γ is denoted by $\Pi(\gamma)$.

1.3 Bisimulation

Generally speaking, two systems are called *bisimilar*, if there is a relation between the systems states with the following properties:

- the initial states are related,

- if the systems are in related states, each of them can simulate the other, i.e. if one system performs some actions, the other can perform the same actions in such a way that the systems end up in related states again.

A relation that makes two systems bisimilar is called a *bisimulation*. Such a relation can be quite easily defined in case of interleaving semantics. For example we can say that two nets $\gamma_1 = (\Sigma_1, M_0^1)$ and $\gamma_2 = (\Sigma_2, M_0^2)$ are interleaving bisimilar if there is a relation $\mathcal{B} \subseteq [M_0^1\rangle \times [M_0^2\rangle$ such that:

(i) $(M_0^1, M_0^2) \in \mathcal{B}$;

(ii) if $(M_1, M_2) \in \mathcal{B}$ and $M_1[w_1\rangle M_1'$, with $w_1 \in T_1^*$, then there exists $w_2 \in T_2^*$ and $M_2' \in [M_0^2\rangle$ such that $M_2[w_2\rangle M_2'$ and $(M_1', M_2') \in \mathcal{B}$;

(iii) if $(M_1, M_2) \in \mathcal{B}$ and $M_2[w_2\rangle M_2'$, with $w_2 \in T_2^*$, then there exists $w_1 \in T_1^*$ and $M_1' \in [M_0^1\rangle$ such that $M_1[w_1\rangle M_1'$ and $(M_1', M_2') \in \mathcal{B}$.

In case that we are interested in process behaviors of systems, it is much more difficult to define bisimulations. Usually, in such a case, one defines simulations of a system by another one with respect to some criteria (by preserving some properties). This one will be the policy that we adopt in the next sections.

Chapter 2

Petri Net Reactive Modules

This chapter proposes *Petri net reactive modules* as models of (discrete) reactive systems, and presents basic results on their semantics [56, 58, 62].

2.1 Definitions and Examples

As we have already said, we are going to model discrete reactive systems that may interact with each other by Petri net reactive modules which are defined as follows [56].

Definition 2.1.1 A *Petri net reactive module* (*Petri net module* or *module*, for short) is a couple $\mathcal{M} = (\gamma, S^c)$, where $\gamma = (\Sigma, M_0, l)$ is a net, called the *underlying net* of $\mathcal{M}$, and S^c is a subset of places of γ, called the *set of interface* or *shared places* of $\mathcal{M}$.

For a module $\mathcal{M}$, the set $S^i = S - S^c$ is called the *set of internal places* of $\mathcal{M}$. When $S^c = \emptyset$ we say that $\mathcal{M}$ is *closed*; otherwise, it is called *open*. All the concepts referring to nets (place, transition, marking, process etc.) are transferred to modules by means of their underlying nets.

The interface places are used by a module to interact with an environment. During an execution, their content is updated by the system (module) or by the *environment*. The content of the internal places can be updated only by the module itself. The distinction between internal and interface places is similar to the distinction between controlled and external variables

in the Alur and Henziger's formalism of reactive modules [4], or to the distinction between unobservable owned variables and observable variables in the formalism of fair Kripke structures in [28] [1].

We define now the *asynchronous parallel composition* of modules. Generally speaking, a parallel composition operation on models of distributed systems combines two models into a single one whose behavior captures, in some sense, the interaction between that two models. There are two major ways of forming the parallel composition of two models, *synchronous* and *asynchronous*, and for each of them different variants are known [6, 28]. In synchronous parallel composition, the models run in parallel and synchronize on actions from a given set of actions. The main use of such an operation is for coupling a system with a *tester* which tests for the satisfaction of a given property. Opposite to the synchronous parallel composition is the asynchronous parallel composition, which does not assume any action synchronization but the systems may communicate via a set of shared variables (locations). The execution of such a system can be viewed as the *interleaved execution* of the components. For examples of parallel compositions of Petri nets the reader is referred to [13, 66, 64, 51, 67, 30].

In order to avoid some annoying and totally unessential things for our purpose we assume that two disjoint countable sets $\mathcal{S}$ and $\mathcal{T}$ are given, and all the nets we consider have the sets of places and transitions included in $\mathcal{S}$ and $\mathcal{T}$, respectively. For a finite set $S^c \subset \mathcal{S}$ and a marking M_0^c on S^c (that is, $M_0^c : S^c \to \mathbf{N}$) consider the set $PN(S^c, M_0^c)$ of all modules whose set of places includes S^c and whose initial marking agrees with M_0^c on S^c. Two modules $\mathcal{M}_0$ and $\mathcal{M}_1$ in this set are called *compatible* if $S_0 \cap S_1 = S^c$ and $T_0 \cap T_1 = \emptyset$.

Definition 2.1.2 Let $\mathcal{M}_0, \mathcal{M}_1 \in PN(S^c, M_0^c)$ be two compatible modules. The *asynchronous parallel composition* of $\mathcal{M}_0$ and $\mathcal{M}_1$, denoted by $\mathcal{M}_0 \circ \mathcal{M}_1$, is the component-wise union of $\mathcal{M}_0$ and $\mathcal{M}_1$, that is:

- $\mathcal{M}_0 \circ \mathcal{M}_1 = (\gamma, S^c)$, $\gamma = (\Sigma, M_0, l)$, and $\Sigma = (S, T, F, W)$;

[1] The set of interface places can be partitioned further into two sets, the set S^{in} of *input places* and the set S^{out} of *output places*. In this way we have a full analogy with the formalisms mentioned above. However, for our purposes such a partition is not important and we will not consider it.

- S, T, F, W, M_0 and l are the union of the sets of places, transitions, flow relations, weight functions, markings and labeling functions of $\mathcal{M}_0$ and $\mathcal{M}_1$, respectively.

We note that the unions of functions in Definition 2.1.2 are well-defined. In the case of M_0 we can write $M_0 = M_0^0|_{S_0^i} + M_0^c + M_0^1|_{S_1^i}$, where M_0^0 and M_0^1 are the initial markings of $\mathcal{M}_0$ and $\mathcal{M}_1$, respectively. The module $\mathcal{M}_0 \circ \mathcal{M}_1$ is an element of $PN(S^c, M_0^c)$.

To have a flexible notation we will identify a module $\mathcal{M} = (\gamma, S^c)$ by its underlying net γ whenever S^c is clear from context. Moreover, for $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ we will write $\gamma_0 \circ \gamma_1$ instead of $\mathcal{M}_0 \circ \mathcal{M}_1$ and we call it the *composition of* γ_0 *and* γ_1 *along* S^c or, simply, the *composition of* γ_0 *and* γ_1.

The asynchronous parallel composition of modules in $PN(S^c, M_0^c)$ is a partially defined binary operation. It is commutative and associative whenever it is defined; that is, $x_0 \circ x_1 = x_1 \circ x_0$ and $(x_0 \circ x_1) \circ x_2 = x_0 \circ (x_1 \circ x_2)$, for all pairwise compatible modules x_0, x_1 and x_2. Moreover, the module $\gamma_\emptyset = ((S^c, \emptyset, \emptyset, \emptyset), M_0^c, \emptyset)$ is the unit of this operation [2].

An important and intensively used method for trying to verify a system is to decompose the system, to verify properties of individual components, and to infer from these properties of the system. There are many ways to decompose a system into components. The one we consider is closely related to the asynchronous parallel composition; it is a *decomposition along* a set of places. It is obvious that, given a net γ and a subset of places S^c, the decomposition along S^c is not necessarily unique. Moreover, it is not always possible to decompose nets such that certain properties of the components be obtained. For example, if we consider the net γ in Figure 2.1 and $S^c = \{s_1\}$, then there is no decomposition of γ into two modules each of them having one transition and such that their asynchronous composition along S^c is the original net γ. However, the decomposition based on subnets generated by subsets of transitions is indeed the inverse operation of the asynchronous composition. If $\Sigma = (S, T, F, W)$ is a net and T_1 is a subset of T, by the *subnet generated by* T_1 we understand the net $\Sigma_1 = (S_1, T_1, F_1, W_1)$, where

[2] One may consider the equivalence relation $\equiv$ on $PN(S^c, M_0^c)$ induced by isomorphisms of labeled nets which preserve S^c (that is, their restrictions to S^c is the identity on S^c) and their initial markings, and define the asynchronous parallel composition on equivalence classes by means of any two compatible representatives of those classes. Then, this operation is totally defined on the quotient set $PN(S^c, M_0^c)/\equiv$ and structures it as a monoid.

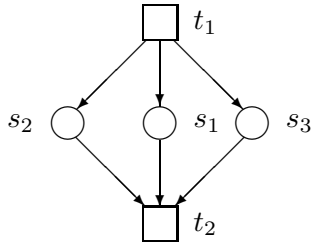


Figure 2.1: A Petri net with no module decomposition along $\{s_1\}$

$S_1 = {}^\bullet T_1^\bullet$, and F_1 and W_1 are the corresponding restrictions of F and W to S_1 and T_1. The subnet generated by $T - T_1$ will be called the *difference* of Σ and Σ_1, and it will be denoted by $\Sigma - \Sigma_1$ (the set ${}^\bullet T_1^\bullet \cap {}^\bullet(T - T_1)^\bullet$ plays the role of interface places between Σ_1 and $\Sigma - \Sigma_1$). These concepts can be naturally extended to (labeled) marked nets. It is clear now that the asynchronous composition of two nets γ_1 and $\gamma - \gamma_1$ as above, along the common set of places, leads to the net γ.

We close the section by two examples of decomposition which will be used in the next section in order to exemplify the process decomposition and process sample generation operations.

Owicki-Lamport's Mutex Algorithm The net γ in Figure 2.2 is a Petri net model of the *Owicki-Lamport's Mutex algorithm*. It consists essentially of two sites: the *writer* and *reader* site, the first one to the left, and the second one to the right, of the dash box in figure. The net uses three flags: the flag *writer detached* (s_2) signals to the reader that the writer is presently not striving to become *writing*, the flag *reader detached* (s_3) likewise signals to the writer that the reader is presently not striving to become *reading*, and the flag *writer involved* (s_1) is just the complement of writer detached (for a detailed discussion about this net model the reader is referred to [42]). These two sites of the net in Figure 2.2 are connected each other by means of s_1, s_2 and s_3. We may separate them into two nets γ_0 and γ_1 by multiplying twice these places together with their initial markings (Figure 2.3). Thus, we obtain two nets γ_0 and γ_1 whose asynchronous composition along $\{s_1, s_2, s_3\}$ is γ.

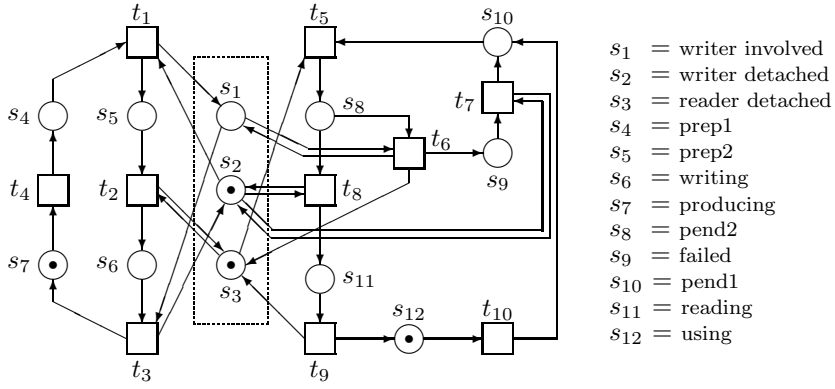


Figure 2.2: Owicki-Lamport's Mutex algorithm

A Vending Machine for Beverages We recall an example from [15] concerning a Petri net model γ of a vending machine for beverages (Figure 2.4).

Initially, the machine is ready for the insertion of a coin. An inserted coin is checked (counterfeit is rejected). When a coin is accepted, a beverage is dispensed and the control is returned to the state *ready*.

The Petri net model in Figure 2.4 may be viewed as composed by two main parts: the *control part* (concerned with accepting/rejecting coins) and the *dispense part* (concerned with dispensing of a beverage). These two parts are connected by means of the places s_1 and s_4. We may separate them into two Petri nets γ_0 and γ_1 by multiplying the places s_1 and s_4 together with their initial markings (Figure 2.5). Thus, we obtain two nets γ_0 and γ_1 whose asynchronous composition along $\{s_1, s_4\}$ is γ.

2.2 Process Decomposition

This section addresses a basic question with respect to the asynchronous parallel composition of two modules $\mathcal{M}_0$ and $\mathcal{M}_1$ with the same set of interface places:

- Which is the structure of processes of $\mathcal{M}_0 \circ \mathcal{M}_1$? Can we decompose them into processes of $\mathcal{M}_0$ and $\mathcal{M}_1$?

In what follows we will answer this question.

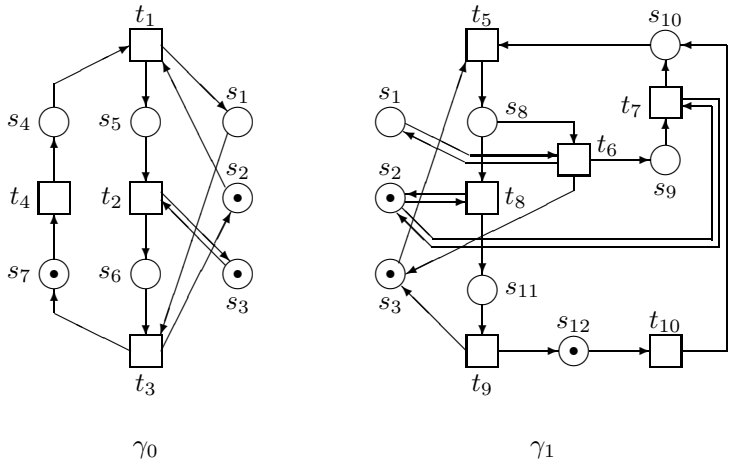


Figure 2.3: A decomposition of the Petri net in Figure 2.2

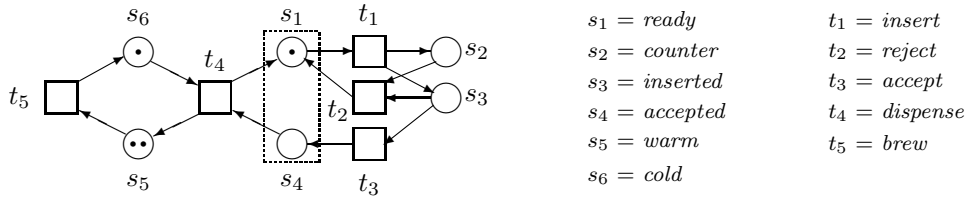


Figure 2.4: A vending machine for beverages

Let us consider the process π of the net γ in Figure 2.2, pictorially represented in Figure 2.6. This process can be split into two parts (occurrence nets) π_0 and π_1 as in Figure 2.7, according to the decomposition of γ (Figure 2.3). The initial cut of π_0 defines a marking of γ_0 with one token in each of s_1, s_2 and s_7, and two tokens in s_3, which is neither reachable in γ nor in γ_0; similarly, the initial cut of π_1 defines a marking of γ_1 with one token in each of s_1, s_2, s_3, and s_{12}, which is neither reachable in γ nor in γ_1. However, π_0 (π_1) can become a process of γ_0 (γ_1) if we increase the initial marking of γ_0 (γ_1) by one token in each of s_1 and s_3 (one token in s_1). This fact is not a fortuitous one, but it is a particular case of the process decomposition theorem as we will see later.

First, we mention that labeled occurrence nets are particular nets whose places are labeled as well. Therefore, we extend the definition of composition along a set S^c to the case of these nets by requiring supplementary $p_1(s) =$

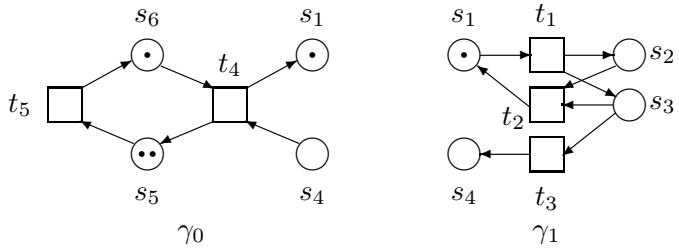


Figure 2.5: A decomposition of the Petri net in Figure 2.4

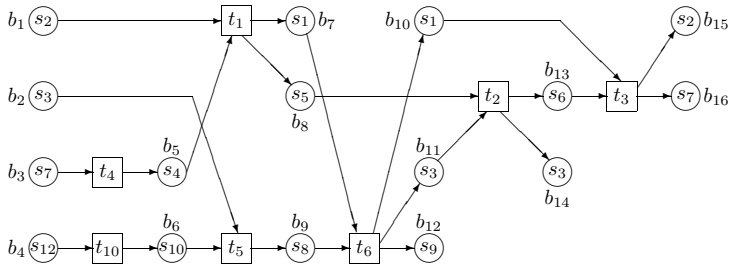


Figure 2.6: A process of the Petri net in Figure 2.2

$p_2(s)$ for all $s \in S^c$ (p_1 and p_2 are the corresponding labeling functions). Then, we adopt one more notation. For a net $\gamma = (\Sigma, M_0, l) \in PN(S^c, M_0^c)$ and a marking $M \in \mathbf{N}^{S^c}$, we denote by $(\gamma + M)$ the net $(\Sigma, M_0 + M, l) \in PN(S^c, M_0^c + M)$. For every two compatible nets $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$ and every marking $M \in \mathbf{N}^{S^c}$ the following property holds:

$$((\gamma_1 \circ \gamma_2) + M) = (\gamma_1 + M) \circ (\gamma_2 + M).$$

Now, we can prove the following important theorem (it has first been proposed in [51] using a different formalism).

Theorem 2.2.1 (Process decomposition theorem [51, 56, 58, 62])
Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets. For each process π of $\gamma_0 \circ \gamma_1$ there are two markings $M', M'' \in \mathbf{N}^{S^c}$ and two processes $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$ such that $\pi = \pi_0 \circ \pi_1$ (the composition of processes is along some set of common conditions).

Proof Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be compatible nets and $\pi = (N, l \circ p')$ be a process of $\gamma_0 \circ \gamma_1$, where $N = (B, E, F')$ (according to the process definition

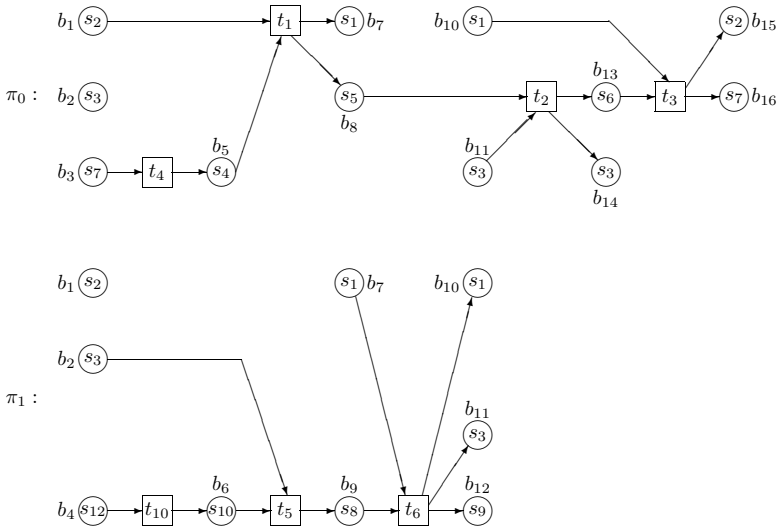


Figure 2.7: A decomposition of the process in Figure 2.6

in Section 1). Let

$$E_k = \{e \in E | p'(e) \in T_k\},$$

for $k = 0, 1$, and let C be the set of conditions of π labeled by places in S^c. This set can be partitioned into the following subsets (not necessarily each of them non-empty):

$$\begin{array}{lcl}
C_{in} & = & C \cap {}^\circ\pi \cap \pi^\circ \\
C_{in,0} & = & \{b \in C \cap {}^\circ\pi | \exists e_0 \in E_0 : \ (b, e_0) \in F'\} \\
C_{in,1} & = & \{b \in C \cap {}^\circ\pi | \exists e_1 \in E_1 : \ (b, e_1) \in F'\} \\
C_{0,1} & = & \{b \in C | \exists e_0 \in E_0, e_1 \in E_1 : \ (e_0, b), (b, e_1) \in F'\} \\
C_{1,0} & = & \{b \in C | \exists e_0 \in E_0, e_1 \in E_1 : \ (e_1, b), (b, e_0) \in F'\} \\
C_0 & = & \{b \in C - C_{0,1} | \exists e_0 \in E_0 : \ (e_0, b) \in F'\} \\
C_1 & = & \{b \in C - C_{1,0} | \exists e_1 \in E_1 : \ (e_1, b) \in F'\}.
\end{array}$$

Conditions from each of these sets (in this order) are pictorially represented in Figure 2.8(a).

Consider now

- $D_k = \{b \in {}^\circ\pi \cap \pi^\circ | p'(b) \in S_k - S^c\}$, for $k = 0, 1$;
- π_0, the subnet of π generated by E_0 together with the set of conditions $D_0 \cup C_{in} \cup C_{in,1}$ (the lower part of Figure 2.8(b));

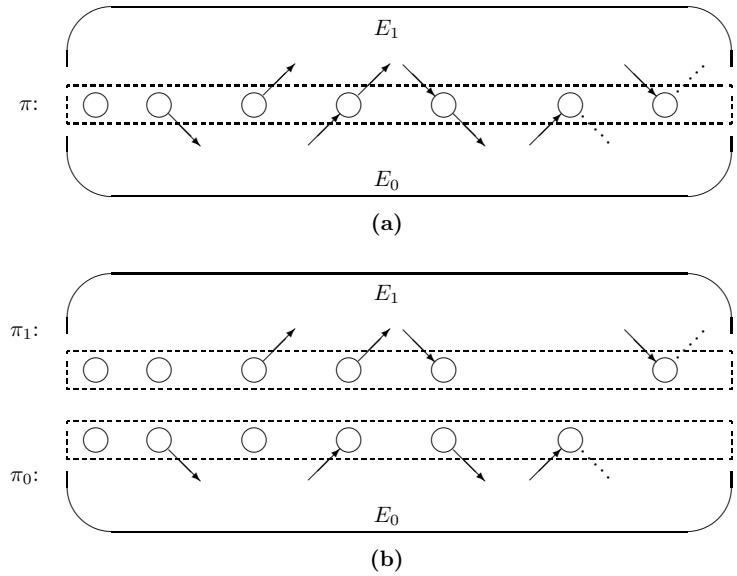


Figure 2.8: Process decomposition

- π_1, the subnet of π generated by E_1 together with the set of conditions $D_1 \cup C_{in} \cup C_{in,0}$ (the upper part of Figure 2.8(b));

- $M' = M_{C_{1,0}}$ and $M'' = M_{C_{0,1}}$.

One can easily verify that $\pi_0 \in \Pi(\gamma_0 + M')$, $\pi_1 \in \Pi(\gamma_1 + M'')$, and $\pi = \pi_0 \circ \pi_1$, where the composition is along the set $B^c = C_{in} \cup C_{in,0} \cup C_{in,1} \cup C_{0,1} \cup C_{1,0}$ of conditions. □

The occurrence net π_0 (π_1) in the theorem above will be called a *process sample of* π *w.r.t.* γ_0 (γ_1).

Example 2.2.1 For the process π in Figure 2.6 we have (with the same notation as in the proof of Theorem 2.2.1):

$$
\begin{array}{lcl}
C & = & \{b_1, b_2, b_7, b_{10}, b_{11}, b_{14}, b_{15}\} \\
C_{in} & = & \emptyset \\
C_{in,0} & = & \{b_1\} \\
C_{in,1} & = & \{b_2\} \\
C_{0,1} & = & \{b_7\} \\
C_{1,0} & = & \{b_{10}, b_{11}\} \\
C_0 & = & \{b_{14}, b_{15}\} \\
C_1 & = & \emptyset \\
D_0 & = & \emptyset \\
D_1 & = & \emptyset.
\end{array}
$$

π_0 and π_1 in Figure 2.7 are processes of $(\gamma_0 + (1,0,1))$ and $(\gamma_1 + (1,0,0))$, respectively (the order on the interface places is s_1, s_2, s_3). Therefore, π_0 is a process sample of π w.r.t. γ_0, and π_1 is a process sample of π w.r.t. γ_1.

Example 2.2.2 A process of the Petri net in Figure 2.4 is pictorially represented in Figure 2.9, and a decomposition of it is given in Figure 2.10 (according to the decomposition in Figure 2.5). In this case, $D_0 = \{b_2, b_3\}$

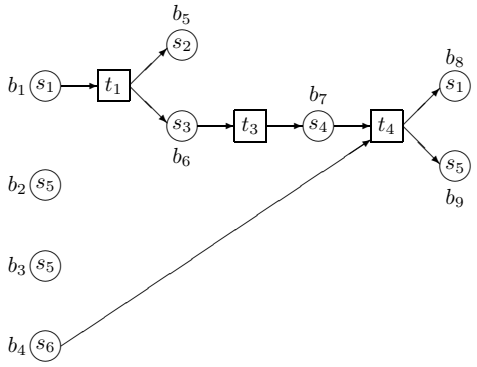


Figure 2.9: A process of the Petri net in Figure 2.4

and $D_1 = \emptyset$.

2.3 Process Composition

Closely related to the question in Section 2.2 is the following important question:

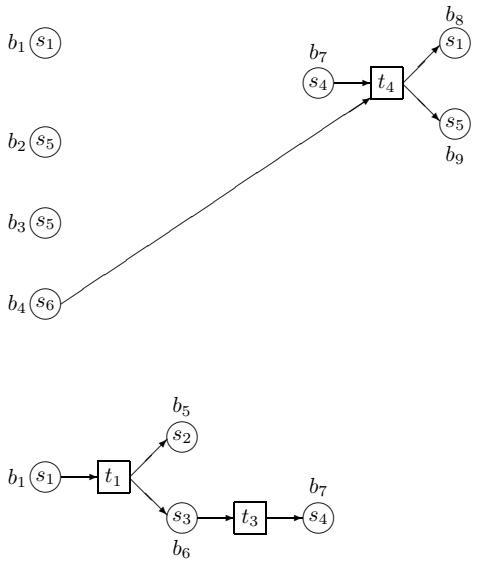


Figure 2.10: A decomposition of the process in Figure 2.9

- Do we obtain processes of $\mathcal{M}_0 \circ \mathcal{M}_1$ if we compose arbitrary processes of $\mathcal{M}_0$ and $\mathcal{M}_1$?

In order to answer this question let us analyze again the proof of Theorem 2.2.1 and see what we want and how we want to do it:

- given two processes $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$, where $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ and $M', M'' \in \mathbf{N}^{S^c}$, we want to compose π_0 and π_1 in order to obtain a process of $\gamma_0 \circ \gamma_1$;

- the initial cut of π_0 corresponds to $M_0^c + M'$, and the initial cut of π_1 corresponds to $M_0^c + M''$;

- the marking M' should be regarded as the marking corresponding to the set of conditions given by π_1 to π_0, and similarly for M'' (see the proof of Theorem 2.2.1);

- due to the above requirement, a subset X_1 of conditions of π_1 and a subset X_0 of conditions of π_0 should be identified such that conditions in these two sets be one-to-one pairable [3]. Moreover, X_1 should define M',

[3] A conditions $b_1 \in X_1$ is pairable with a condition $b_0 \in X_0$ if they have the same label, $|{}^\bullet b_0 \cup {}^\bullet b_1| \leq 1$ and $|b_0^\bullet \cup b_1^\bullet| \leq 1$.

each condition in X_1 should have in input arc but no output arc, and each condition in X_0 should have no output arc. Therefore, $X_1 \cap {}^\circ\pi_1 = \emptyset$ and $X_0 \subseteq {}^\circ\pi_0$. Similar reasons work for M'' and, therefore, two subsets $Y_0 \subseteq B_0$ and $Y_1 \subseteq B_1$ should be identified satisfying similar properties to those above;

- finally, ${}^\circ\pi_0 - X_0$ and ${}^\circ\pi_1 - Y_1$ should be pairable.

As a conclusion, we can say that B_0 should be partitioned into ${}^\circ\pi_0 \cup Y_0$ and $B_0 - ({}^\circ\pi_0 \cup Y_0)$, and B_1 should be partitioned into ${}^\circ\pi_1 \cup X_1$ and $B_1 - ({}^\circ\pi_1 \cup X_1)$ such that the sets ${}^\circ\pi_0 \cup Y_0$ and ${}^\circ\pi_1 \cup X_1$ be pairable in a well-defined sense.

Definition 2.3.1 Let $\gamma \in PN(S^c, M_0^c)$, π a process of γ, and let $C(\pi)$ be the set of all conditions of π labeled by places in S^c. A *p-partition* of $C(\pi)$ is any couple of sets $(A, C(\pi) - A)$ such that:

- $A \subseteq C(\pi)$;
- $(\forall b \in C(\pi) - A)(|{}^\bullet b| = 1)$ and $(\forall b \in C(\pi))(|{}^\bullet b \cup b^\bullet| = 2 \;\Rightarrow\; b \in C(\pi) - A)$.

We remark that ${}^\circ\pi \subseteq A$. Intuitively, $C(\pi) - A$ plays the role of C_0 or C_1 in the proof of Theorem 2.2.1.

Definition 2.3.2 Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$, $\pi_0 \in \Pi(\gamma_0)$ and $\pi_1 \in \Pi(\gamma_1)$. A *p-composition function* from π_0 to π_1 is any labeled-preserving bijection f from A_0 into A_1 such that:

1. $(A_0, C(\pi_0) - A_0)$ is a p-partition of $C(\pi_0)$, and $(A_1, C(\pi_1) - A_1)$ is a p-partition of $C(\pi_1)$;
2. (*f is non-branching*)
   $(\forall b_0, b_1)(f(b_0) = b_1 \;\Rightarrow\; |{}^\bullet b_0 \cup {}^\bullet b_1| \leq 1 \;\wedge\; |b_0^\bullet \cup b_1^\bullet| \leq 1)$;
3. (*f is in-out*)
   $(\forall b_0, b_1)(f(b_0) = b_1 \;\wedge\; |{}^\bullet b_0 \cup {}^\bullet b_1| = 1 \;\Rightarrow\; |b_0^\bullet \cup b_1^\bullet| = 1)$;
4. (*the triple (π_0, π_1, f) is cycle-free*)
   $(\nexists b_0, b_1, b_0', b_1')(f(b_0) = b_1 \;\wedge\; f(b_0') = b_1' \;\wedge\; b_1 \prec_{\pi_1} b_1' \;\wedge\; b_0' \prec_{\pi_0} b_0)$.

It is easy to see that for $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$ in the proof of Theorem 2.2.1, the identity function f on

$$B^c = C_{in} \cup C_{in,0} \cup C_{in,1} \cup C_{0,1} \cup C_{1,0}$$

is a p-composition function from π_0 to π_1. Moreover, for these processes we have:

(5) $M' = M_{\{b \in Cod(f) || {}^\bullet b| = 1\}}$ and $M'' = M_{\{b \in Dom(f) || {}^\bullet b| = 1\}}$, where $Dom(f)$ and $Cod(f)$ denote the domain and the codomain of f, respectively.

When a p-composition function satisfies (5) we say that it is *compatible with* M' and M''.

Theorem 2.3.1 (Process composition theorem [56, 62])
Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets, M' and M'' two markings on S^c, $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$. Then, for every p-composition function $f : A_0 \to A_1$ from π_0 to π_1 compatible with M' and M'', there is a process $\pi_1' \in \Pi(\gamma_1 + M'')$ such that:

- π_1' is obtained from π_1 by renaming its elements (but not the labels) [4];
- $\pi_0 \circ \pi_1' \in \Pi(\gamma_0 \circ \gamma_1)$ (the composition of processes is along A_0, whereas the composition of nets is along S^c).

Proof Let $f : A_0 \to A_1$ be a function as in the theorem's hypothesis. Rename the elements of the process π_1 such that:

- the elements $x \notin A_1$ are renamed in a distinct way from all the elements of π_0;
- each element $b_1 \in A_1$ is renamed b_0, where $f(b_0) = b_1$.

[4] Formally, there is a bijection $\varphi : B_1 \cup E_1 \to B_1' \cup E_1'$ such that:

- $p_1(x) = p_1'(\varphi(x))$, for all $x \in B_1 \cup E_1$;
- $x \prec_{\pi_1} y$ iff $\varphi(x) \prec_{\pi_1'} \varphi(y)$, for all $x \in B_1 \cup E_1$

(p_1 and p_1' are the labeling functions of π_1 and π_1', respectively). In fact, this is the classical concept of *isomorphism* of processes. We did not consider it yet because in Section 3.2 we will introduce a more general concept of isomorphism.

Let π_1' be the process such obtained.

The second property of f, together with the fact that π_0 and π_1 are processes, assures that $|{}^\bullet b| \leq 1$ and $|b^\bullet| \leq 1$, for all conditions b. The fourth property of f assures that $\prec_{\pi_0 \circ \pi_1'}$ is acyclic. Therefore, $\pi_0 \circ \pi_1'$ is an occurrence net.

We shall prove that $\pi_0 \circ \pi_1'$ is a process of $\gamma_0 \circ \gamma_1$ by using the axiomatic definition of processes. Clearly, all conditions of $\pi_0 \circ \pi_1'$ are labeled by places, and all events by transitions.

Checking the property "$M_0(s) = |p^{-1}(s) \cap {}^\circ\pi|$ for all $s \in S$" can be reduced to checking "$M_0^c(s) = |p^{-1}(s) \cap {}^\circ\pi|$ for all $s \in S^c$" by the remark that $M_0 = M_0^0|_{S_0^i} + M_0^c + M_0^0|_{S_1^i}$.

Let $s \in S^c$. The initial cut of π_0 contains exactly $(M_0^c + M')(s)$ conditions labeled by s. Due to the fact that f is compatible with M', A_1 contains exactly $M'(s)$ conditions labeled by s which have an input arc but no output arc. Now, the third property of f leads to the fact that these conditions are paired (by f) with $M'(s)$ conditions labeled by s in the initial cut of π_0. A similar reasoning works for M'' too. Therefore, $M_0^c(s)$ conditions in $A_0 \cap {}^\circ\pi_0$ labeled by s are paired by f with $M_0^c(s)$ conditions in $A_1 \cap {}^\circ\pi_1$ labeled by s. Moreover, these are the only possible pairings between conditions in the initial cuts of these two processes. Hence,

$$M_0^c(s) = |p^{-1}(s) \cap {}^\circ\pi|$$

It is straightforward to see that the last property in the axiomatic definition of processes holds true for $\pi_0 \circ \pi_1'$. Therefore, $\pi_0 \circ \pi_1' \in \Pi(\gamma_0 \circ \gamma_1)$. □

2.4 Compositional Semantics

Theorems 2.2.1 and 2.3.1 lead to a compositional semantics of modules.

Roughly speaking, a semantics is *compositional* if the semantics of a whole is a function of the semantics of its parts. Formally, in order to define a compositional semantics for a class of formal systems, two things are necessary:

- to define a semantics of the systems in that class;
- to define two operations: one on systems (denoted $\circ_1$), and one on their semantics (denoted $\circ_2$).

Then, the semantics is compositional if the following property holds

$$\text{Semantics}(\mathcal{S}_1 \circ_1 \mathcal{S}_2) = \text{Semantics}(\mathcal{S}_1) \circ_2 \text{Semantics}(\mathcal{S}_2),$$

for all systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in the class.

Petri net modules can be associated with a compositional semantics as it has been shown in [56, 62]. First, we define the semantics Π_m of Petri net modules by

$$\Pi_m(\gamma) = \bigcup_{M \in \mathbf{N}^{S^c}} \Pi(\gamma + M),$$

for any $\gamma \in PN(S^c, M_0^c)$. Consider then the process composition operator $\circ_{M_0^c}$ as suggested by the composition theorem. Formally, let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets. Then, for every $M', M'' \in \mathbf{N}^{S^c}$, $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$, consider the set $\pi_0 \circ_{M_0^c} \pi_1$ of all processes $\pi_0 \circ \pi_1$, where the composition, whenever it is possible, is along some set A_0 of conditions such that there is a p-composition function from π_0 to π_1 compatible with M' and M'' and whose domain is A_0. Extend then this operation, by union, to sets of processes.

Lemma 2.4.1 Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets. Then,

$$\Pi(\gamma_0 \circ \gamma_1) = \Pi_m(\gamma_0) \circ_{M_0^c} \Pi_m(\gamma_1).$$

Proof Directly from definitions, Theorem 2.2.1 and Theorem 2.3.1. □

Define now the composition operator $\circ_{\geq M_0^c}$ on processes of compatible nets $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$, as follows. For every $M', M'' \in \mathbf{N}^{S^c}$, $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$, consider the set $\pi_0 \circ_{\geq M_0^c} \pi_1$ of all processes $\pi_0 \circ \pi_1$, where the composition, whenever it is possible, is along some set A_0 of conditions such that there is a p-composition function from π_0 to π_1 compatible with $M' - M$ and $M'' - M$ and whose domain is A_0, for some marking M smaller than both M' and M'' (in other words, $\pi_0 \circ \pi_1$ is a process of $(\gamma_0 \circ \gamma_1 + M)$). Extend then this operation, by union, to sets of processes.

We are able to prove now that Π_m is a compositional semantics for Petri net modules.

Theorem 2.4.1 (Compositional semantics [56, 62])
Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets. Then,

$$\Pi_m(\gamma_0 \circ \gamma_1) = \Pi_m(\gamma_0) \circ_{\geq M_0^c} \Pi_m(\gamma_1).$$

Proof For every marking M on S^c we have:

$$\begin{array}{rcl} \Pi((\gamma_0 \circ \gamma_1) + M) & = & \Pi((\gamma_0 + M) \circ (\gamma_1 + M)) \\ & = & \Pi_m(\gamma_0 + M) \circ_{M_0^c + M} \Pi_m(\gamma_1 + M) \\ & \subseteq & \Pi_m(\gamma_0) \circ_{\geq M_0^c} \Pi_m(\gamma_1) \end{array}$$

(the second equality is based on Lemma 2.4.1, and the inclusion on the definition of Π_m and $\circ_{\geq M_0^c}$). Thus, $\Pi_m(\gamma_0 \circ \gamma_1) \subseteq \Pi_m(\gamma_0) \circ_{\geq M_0^c} \Pi_m(\gamma_1)$.

Conversely, for all markings M, M' and M'' on S^c, and all processes $\pi_0 \in \Pi((\gamma_0 + M) + M')$ and $\pi_1 \in \Pi((\gamma_1 + M) + M'')$, if there is a p-composition function from π_0 to π_1 compatible with M' and M'' and whose domain is A_0, then the composition of π_0 and π_1 along A_0, whenever it is possible, is a process of $(\gamma_0 \circ \gamma_1) + M$. That is, $\Pi_m(\gamma_0) \circ_{\geq M_0^c} \Pi_m(\gamma_1) \subseteq \Pi_m(\gamma_0 \circ \gamma_1)$. □

Chapter 3

Replacement Techniques

Refinement and *abstraction* are two important operations in Petri net theory, both of them being particular cases of *replacement*. By this operation, a subnet of a given net is replaced by another net. The main goal of a replacement operation is to transform the original net by preserving certain properties of it. Usually, these properties are: boundedness, liveness, interleaving behavior, process behavior, trace behavior, partial order behavior etc.

In this chapter we will survey some of the most important refinement and abstraction techniques found in the literature. Then, a very general replacement technique is discussed [51, 56, 58, 62], and semantics preservation results are provided.

3.1 Refinement and Abstraction

When a subnet of a given net is replaced by a more detailed net, the operation is usually called *refinement*. On the contrary, when the replacement is by a less detailed net, the operation is called *abstraction*. In literature, mostly refinement was studied. Various techniques and a large number of behavior and equivalence relations preserving refinement have been proposed [63, 47, 38, 65, 20, 29, 7, 8, 66, 10, 51]. In this section we are going to briefly review a few methods just to provide the reader with an intuitive image about refinement. Of course, some methods improve over the others and, therefore, we will present the most important ones in their classes. We emphasis again that our thesis focuses on semantics preservation results and correctness proof

techniques which, in some sense, is a different goal than the one in most of the papers mentioned above (which focus on boundedness, liveness, deadlock etc, as you will see in what follows).

Valette's Refinement Technique A first important refinement technique has been proposed by Valette in [63]. By this technique, a transition is refined by a *block* which is a Petri net with two distinguished transitions. Let assume that we want to refine the transition t of the net in Figure 3.1. We may view

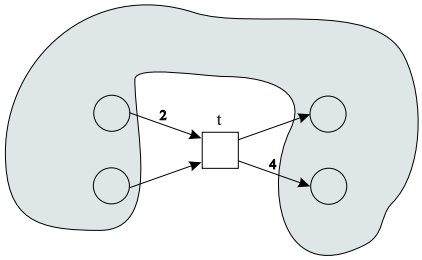


Figure 3.1: A Petri net and a transition t of it

t as being composed by two transitions, t^- and t^+, as in Figure 3.2. Now, in

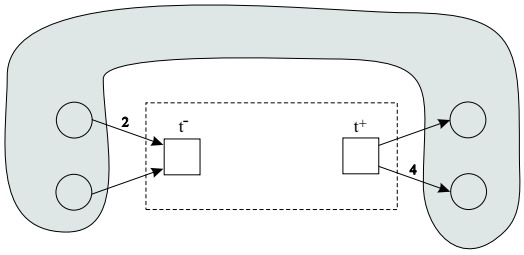


Figure 3.2: t is composed by two transitions, t^- and t^+

order to refine t we may insert any Petri net in between t^- and t^+. The net which is to be inserted should have two distinguished transitions t^- and t^+ as in Figure 3.3, in order to assure a match for the insertion. The result is the net pictorially represented in Figure 3.4.

Let us formalize a little bit this operation.

Definition 3.1.1 ([63]) A *block* is any (labeled) marked net γ together with two distinguished and distinct transitions t^-, called the *initial transition*, and t^+, called the *final transition*, of γ.

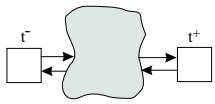


Figure 3.3: The net to be inserted

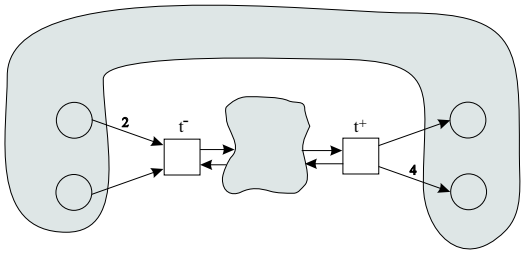


Figure 3.4: The result of t's refinement

We will denote blocks by (γ, t^-, t^+). The net in Figure 3.3 can be regarded as a block (having t^- and t^+ as distinguished transitions).

Definition 3.1.2 ([63]) Let γ_1 be a net, $t \in T_1$, and (γ, t^-, t^+) be a block such that γ_1 and γ are disjoint. Refining t by (γ, t^-, t^+) yields the *refined net* $\gamma_2 = \gamma_1[t/\gamma]$, which is defined as follows:

- $S_2 = S_1 \cup S$;
- $T_2 = (T_1 - \{t\}) \cup T$;
- W_2 is given by

$$W_2(x,y) = \begin{cases} W_1(x,y), & \text{if } x,y \in S_1 \cup (T_1 - \{t\}) \\ W(x,y), & \text{if } x,y \in S \cup T \\ W_1(x,t), & \text{if } x \in S_1 \text{ and } y = t^- \\ W_1(t,y), & \text{if } y \in S_1 \text{ and } x = t^+ \\ 0, & \text{otherwise} \end{cases}$$

- the initial marking M_0^2 of γ_2 is

$$M_0^2(s) = \begin{cases} M_0^1(s), & \text{if } \ s \in S_1 \\ M_0(s), & \text{if } \ s \in S \end{cases}$$

- the labeling function l_2 of γ_2 is

$$l_2(x) = \begin{cases} l_1(x), & \text{if } x \in T_1 - \{t\} \\ l(x), & \text{if } x \in T \end{cases}$$

We are interested in refining transitions by blocks, and by preserving certain properties such as boundedness, liveness etc. It is very clear that the refining block should satisfy certain properties. This properties will be formulated by means of a new construction associated to the block, called the *closer* of that block.

The *closer* of a block (γ, t^-, t^+) is the 4-tuple $(\bar{\gamma}, t^-, t^+, io)$, where $\bar{\gamma}$ is the net obtained from γ by adding a new place io which will be an input place of t^- and an output place of t^+. These two arcs have weight one, and no other arc is added. The initial marking of γ is extended to $\bar{\gamma}$ by marking io by one token. This marking is denoted by $\bar{M}_0$. The closer of a block looks like in Figure 3.5.

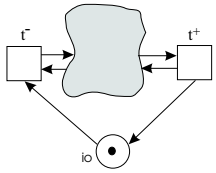


Figure 3.5: The closer of a block

Definition 3.1.3 ([63]) A block (γ, t^-, t^+) is called *well-formed* if its closer satisfies the following properties:

1. $\bar{\gamma}$ is live;
2. $(\forall M \in [\bar{M}_0\rangle_{\bar{\gamma}})(M(io) > 0 \quad \Rightarrow \quad M = \bar{M}_0)$;
3. $(\forall t \in T)(\bar{M}_0[t\rangle_{\bar{\gamma}} \quad \Rightarrow \quad t = t^-)$.

A well-formed block (γ, t^-, t^+) can simulate a transition t as follows. The only transition that can activate $\bar{\gamma}$ is the initial transition t^-. Thus a simulation can only start if the environment, which is represented by the place io, allows this. If $\bar{\gamma}$ is activated, always a transition sequence can fire that enables the final transition, since $\bar{\gamma}$ is live. Hence every simulation can be finished. The firing of the final transition marks the place io, i.e. produces the

output for the environment. Now we have reached the initial marking again, i.e. a well-formed block does not have a "memory", just like a procedure call does not depend on earlier calls. During the simulation of a transition t by $\bar{\gamma}$, the place io is empty. Therefore it is not possible to start a second simulation of t before the first is finished. Only when the simulation has come to an end, a new simulation can be initiated by the initial transition. The environment of t in a host net γ_1 has to ensure this property, therefore the transition that is to be refined must not be 2-*enabled* by any reachable marking of γ_1 (a transition $t \in T_1$ is k-*enabled* by a marking M if $k \cdot W_1(s,t) \leq M(s)$, for all $s \in S_1$).

Then the following results concerning behavior preservation are obtained.

Theorem 3.1.1 ([63]) Let γ_1 be a net, $t \in T_1$ a transition that is not 2-enabled by any reachable marking of γ_1, and (γ, t^-, t^+) a well-formed block. Then, the following properties hold:

1. γ_1 is bounded iff $\gamma_1[t/\gamma]$ is bounded.
2. γ_1 and $(\bar{\gamma}, t^-, t^+, io)$ are safe iff $\gamma_1[t/\gamma]$ is safe.
3. γ_1 is live iff $\gamma_1[t/\gamma]$ is live.

Suzuki and Murata's Extension The results obtained by Valette have been extended later by Suzuki and Murata [47].

Define first the k-closer of a block (γ, t^-, t^+), where $k \geq 1$, as the closer of that block but with the difference that the place io is marked by k tokens. Denote this structure (net) by $(\bar{\gamma}, t^-, t^+, k \cdot io)$.

By $\#(w, x)$ we will denote the number of occurrences of the element x in the sequence w.

Definition 3.1.4 ([47]) A block (γ, t^-, t^+) is called *well-behaved* if its k-closer satisfies the following properties:

1. t^- is live in $(\bar{\gamma}, t^-, t^+, k \cdot io)$;
2. for any transition sequence w in $(\bar{\gamma}, t^-, t^+, k \cdot io)$, $\#(w, t^-) \geq \#(w, t^+)$;
3. if $\#(w, t^-) > \#(w, t^+)$ for some transition sequence w in the k-closer $(\bar{\gamma}, t^-, t^+, k \cdot io)$, then there exists $w' \in (T - \{t^-\})^+$ such that ww' is a transition sequence in $(\bar{\gamma}, t^-, t^+, k \cdot io)$ and $\#(w, t^-) = \#(ww', t^+)$.

Let us examine, how a k-well-behaved block (γ, t^-, t^+) simulates a transition t. Since we have k tokens on the idle place, at most k simulations can be started simultaneously. Liveness of the initial transition ensures that at least one simulation is possible. Because of the third condition every run of $(\bar{\gamma}, t^-, t^+, k \cdot io)$ can be finished (by the firing of the final transition). The second condition says that io is k-bounded, since every firing of the final transition corresponds to a firing of the initial transition. When all simulations are finished, there are k tokens on io, but the reached marking may differ from the initial marking on the inner places.

The following result is analogous to Theorem 3.1.1.

Theorem 3.1.2 ([47]) Let γ_1 be a net, $t \in T$ a transition that is not $(k+1)$-enabled by any reachable marking of γ_1, and (γ, t^-, t^+) a k-well-behaved block. Then, the following properties hold:

1. If $\gamma_1[t/\gamma]$ is m-bounded then γ_1 is m-bounded.
2. If $\gamma_1[t/\gamma]$ is safe then γ_1 is safe.
3. If γ_1 is m-bounded and $(\bar{\gamma}, t^-, t^+, k \cdot io)$ is m'-bounded, then $\gamma_1[t/\gamma]$ is $max(m, m')$-bounded.
4. If γ_1 and $(\bar{\gamma}, t^-, t^+, 1 \cdot io)$ are safe, then $\gamma_1[t/\gamma]$ is safe.
5. If $\gamma_1[t/\gamma]$ is live then γ_1 is live.
6. If γ_1 and $(\bar{\gamma}, t^-, t^+, k \cdot io)$ are live and

   $$(*) \quad (\forall M \in [M_0^1\rangle_{\gamma_1})(\exists M' \in [M\rangle_{\gamma_1})(t \text{ is } k\text{-enabled by } M'),$$

   then $\gamma_1[t/\gamma]$ is live.

Note that we have to require that $(\bar{\gamma}, t^-, t^+, k \cdot io)$ is m'-bounded in (3) and that $(\bar{\gamma}, t^-, t^+, k \cdot io)$ is live in (6) since k-well-behavedness does not ensures boundedness of (γ, t^-, t^+) but only ensures liveness of t^-. On the contrary, in the definition of well-formedness liveness of (γ, t^-, t^+) is explicitly stated and, therefore, boundedness follows from liveness of t^+ and the fact that the firing of t^+ reproduces the initial marking.

For practical reasons we are interested in deciding whether a refinement is behavior preserving, i.e. whether the conditions of Theorem 3.1.2 hold.

Theorem 3.1.3 ([47])

1. It is decidable whether a given transition of a net is $(k+1)$-enabled at a given marking M, for a given $k \geq 1$.

2. It is decidable whether a given block is k-well-behaved, for a given $k \geq 1$.

3. It is decidable whether a given net satisfies the condition $(*)$ in Theorem 3.1.2(6), for a given $k \geq 1$.

In [47] it is suggested to use the same refinement technique also for refinement of places. The place s that is to be refined is split into two places and a transition is added in between. Then, this transition is refined as above. The results gained for the refinement of transitions can be transfered to this case too.

Refinement by Daughter Nets Let look again at the refinement techniques defined so far. Let γ_1 be a net and t a transition in γ_1 that is to be refined by some refinement net (γ, t^-, t^+). First of all, we may allow more than one transition like t^- and more than one transition like t^+. Therefore, we may assume that t is refined by (γ, T^-, T^+), where T^- and T^+ are non-empty subsets of T (we may assume they are disjoint as well). Then, the refinement can be simply defined by connecting every place s in the pre-set of t with every *initial transition* of γ (i.e., transition in T^-) having the same weight as the arc between s and t. Symmetrically, *final transitions* (i.e., transitions in T^+) and places in the postset of t are connected. We have obtained a natural generalization of the refinement technique presented above. We can do a little bit more. First, let us remark that firings of initial transitions absorb the same number of tokens from each place in the preset of t as firings of t. These tokens *cannot be distributed* between two initial transitions or between an initial transition and a transition not belonging to γ. Symmetrically, firings of final transitions produce the same number of tokens on each place in the postset of t as firings of t. Especially it is not possible that same output is produced and used by the environment before the rest is produced. We say that *distributed input* and *distributed output* is not allowed.

In [65], a refinement technique is proposed, where distributed input and distributed output is possible. A transition t is removed and a *daughter-net* γ

is inserted by identifying all places in the preset of t with some special places of γ, called *input places*, and all places in the postset of t with some other places of γ, called *output places*. Thus, several initial transitions (postsets of input places) and several final transitions (presets of output places) are allowed.

Hence the set of places of a daughter-net γ should be a disjoint union of a non-empty set S_i of *input places*, a set S of *inner places*, and a non-empty set S_o of *output places*. Furthermore, we assume that ${}^\bullet S_i \neq \emptyset \neq S_o^\bullet$ and that input and output places are not marked under the initial marking. For simplicity we assume that the weights on arcs to or from the transition t that is to be refined are 1. We emphasize that the set of places of a daughter net is $S_i \cup S \cup S_o$ and the set of transitions is T.

Since distributed input and output is possible, this refinement technique is more general than the basic refinement technique. But it can only be applied to a transition t if ${}^\bullet t = S_i$ and $t^\bullet = S_o$. Otherwise the refinement is not possible. One has to remark that if ${}^\bullet t \cap t^\bullet = \emptyset$, $|{}^\bullet t| = |S_i|$ and $|t^\bullet| = |S_o|$, then we can rename the places in S_i and S_o such that ${}^\bullet t = S_i$ and $t^\bullet = S_o$ and, therefore, the refinement technique can be applied. However, the result of the refinement might be different for different renamings.

Definition 3.1.5 Let γ_1 be a net and $t \in T_1$ such that $W_1(s,t) \leq 1$ for all $s \in S_1$. Let γ be a daughter-net as defined above such that ${}^\bullet t = S_i$ and $t^\bullet = S_o$. We assume that $S_1 \cup T_1$ and $S \cup T$ are disjoint. Refining t by γ yields the refined net $\gamma_2 = \gamma_1[t/\gamma]$ which is defined as follows:

- $S_2 = S_1 \cup S$;
- $T_2 = (T_1 - \{t\}) \cup T$;
- W_2 is given by

$$W_2(x,y) = \begin{cases} W_1(x,y), & \text{if } x,y \in S_1 \cup (T_1 - \{t\}) \\ W(x,y), & \text{if } x,y \in S \cup S_i \cup S_o \cup T \\ 0, & \text{otherwise.} \end{cases}$$

- the initial marking M_0^2 and the labeling function l_2 of γ_2 are given as in Definition 3.1.2.

In the previous approach, blocks were designed such that they simulate a transition. As a result, the interleaving semantics is preserved. Now we

look at things the other way round. We give the desired property, namely γ_1 and $\gamma_1[t/\gamma]$ should have equivalent behavior, and try to determine which daughter-nets ensures this. These daughter-nets will be called called *fair daughter-nets*. The main problem still remain: to define an appropriate notion of behavior. The following definition gives an answer to this problem.

Definition 3.1.6 ([65]) Let γ be a daughter-net. γ is called a *fair daughter-net* if for all nets γ_1 and all $t \in T_1$ satisfying

- $W_1(s,t) \leq 1$ and $W_1(t,s) \leq 1$ for all $s \in S_1$, and
- ${}^\bullet t = S_i$ and $t^\bullet = S_o$,

the nets γ_1 and $\gamma_2 = \gamma_1[t/\gamma]$ have the same behavior in the sense

- for any $M \in [M_0^2\rangle$ there exists $M' \in [M_1\rangle$ and $w \in t^*$ such that $M[w\rangle M''$ and $M''|_{S_1} = M'$. Moreover, if $M'[t\rangle M_1$ then there exists $w'' \in T^*$ such that $M''[w\rangle M_2$ and $M_2|_{S_1} = M_1$.

The behaviour notion in Definition 3.1.6 means the following. Each reachable marking M in the refined net corresponds to a reachable marking M' in γ_1 after normalization of M by a firing sequence of γ, and if t is enabled under the corresponding marking M', then firing of t can be simulated by a transition sequence w'' in γ, i.e. the marking reached after the firing of t corresponds to the marking reached after the firing of w''.

The next theorem follows from Definition 3.1.6 and Theorem 1(iv), Corollary 1, Definition 3, and Proposition 1 in [65].

Theorem 3.1.4 ([7]) Let γ_1 be a net, $t \in T_1$ such that $W_1(s,t) \leq 1$ and $W_1(t,s) \leq 1$ for all $s \in S_1$, and let γ be a fair daughter-net such that ${}^\bullet t = S_i$ and $t^\bullet = S_o$. If the following conditions hold true then $\gamma_1[t/\gamma]$ is bounded:

1. If s is an m-bounded place of γ_1, then s is m-bounded in $\gamma_1[t/\gamma]$;
2. A transition $t' \neq t$ of γ_1 is live in γ_1 if and only if it is live in $\gamma_1[t/\gamma]$;
3. γ_1 and γ' are bounded, where γ' is obtained from γ by removing the input and output places.

The main result in [65] is a characterization of fair daughter-nets that does not involves the (test) nets γ_1 but only the daughter-net that is in question.

Theorem 3.1.5 ([65]) A daughter-net γ is fair if and only if

(1) for any transition $t \in T$ the following property holds true:

$$((\exists s_1, s_2 \in S_i)(W(s_1,t) \neq W(s_2,t)) \quad \Rightarrow \quad t \text{ is dead in } \gamma'),$$

where γ' is constructed from γ by removing the input places;

(2) for any natural number $m > 0$ the following property holds true:

$$(\forall M \in [M_{\gamma_m}\rangle)(\exists w \in T^*)(M[w\rangle M' \ \wedge \ (\forall s \in S_o)(M'(s) = m)),$$

where γ_m is constructed from γ by placing m tokens on each input place, keeping the current initial marking on inner places, and setting on zero each output place.

We have the following decidability result, whose proof is by reduction to the liveness problem.

Theorem 3.1.6 ([65]) It is decidable whether a given daughter-net is fair.

Replacement by Open Interface Nets The transition refinement idea presented above can be generalized in a very natural way: replace a subnet which is connected to the host net by means of a subset of places, by another net. Of course, this is a very general operation and the main problem is not to define this operation (which is straightforward) but to study its properties. In 1993, Chehaibar has shown that this operation preserves interleaving behavior but, unfortunately, this is in a very restrictive way as you will see a little bit later [10].

Definition 3.1.7 An *open interface net*, abbreviated *OI-net*, is a 4-tuple $\psi = (\Sigma, S_o, \mathcal{M}, M_0)$ where:

- $\Sigma = (S, T, F, W)$ is a net;
- $S_o \subseteq S$ is the set of *interface places* ($S_i = S - S_o$ is the set of *inner places*);
- $\mathcal{M} \subseteq \mathbf{N}^{S_i}$ is the set of *stable states* (*markings*);
- $M_0 \in \mathcal{M}$ is the *initial marking* of ψ.

The domain of open interface nets is denoted $\mathcal{OIN}$, and $\mathcal{OIN}_{S_o}$ is the domain of OI-nets whose set of interface places is S_o.

Given a Petri net $\Sigma = (S, T, F, W)$ define the following functions:

- $C : S \times T^* \to \mathbf{Z}$ given by
  - $C(s, \lambda) = 0$;
  - $C(s, t) = W(t, s) - W(s, t)$;
  - $C(s, \sigma t) = C(s, \sigma) + C(s, t)$;
- $V : (S \times T^*) \cup (T^* \times S) \to \mathbf{Z}$ given by
  - $V(s, \lambda) = 0 = V(\lambda, s)$;
  - $V(s, \sigma t) = max\{V(s, \sigma), W(s, t) - C(s, \sigma)\}$;
  - $V(\sigma, s) = C(s, \sigma) + V(s, \sigma)$,

for all $s \in S$, $t \in T$ and $\sigma \in T^*$.

These functions will be indexed in correspondence with the net they are referring to. For example, C_1 and V_1 refer to Σ_1 or ψ_1

Definition 3.1.8 Let $\psi_1, \psi_2 \in \mathcal{OIN}_{S_o}$, $\sigma_1 \in T_1^*$ and $\sigma_2 \in T_2^*$. We say that σ_2 *simulates* σ_1 *with respect to* S_o, written $\sigma_1 \leq_{S_o} \sigma_2$, if the following properties hold true:

1. $C_2(s, \sigma_2) = C_1(s, \sigma_1)$, for all $s \in S_o$;
2. if $\sigma_1 = t_1 \cdots t_n \in T_1^n$ for some n, then there exists $w_1, \ldots, w_n \in (T_2^*)^n$ such that:
   (a) $\sigma_2 = w_1 \cdots w_n$;
   (b) $(\forall s \in S_o)(\forall 1 \leq k \leq n)(C_2(s, w_1 \cdots w_k) \geq C_1(s, t_1 \cdots t_k))$;
   (c) $(\forall s \in S_o)(\forall 1 \leq k \leq n)(V_2(s, w_k) \leq V_1(s, t_k))$.

Definition 3.1.8(1) says that σ_1 and σ_2 carry out the same transformation on the interface places, while Definition 3.1.8(2) expresses the difference of atomicity and the interaction with an environment: there is a decomposition of σ_2 such that after the k-th step σ_2 has produced at least as much as σ_1, and for each step it consumes at most as much as σ_1. Notice that w_i may be the empty word for some i.

The net obtained from Σ by removing a subset S' of places (and the corresponding arcs too) will be denoted by $\Sigma - S'$.

Definition 3.1.9 Let $\psi_1, \psi_2 \in \mathcal{OIN}_{S_o}$, $\longrightarrow'_1$ be the firing relation induced by $(\Sigma_1 - S_o, M_{01})$, and $\longrightarrow'_2$ be the firing relation induced by $(\Sigma_2 - S_o, M_{02})$. A binary relation $\mathcal{R}_{SST} \subseteq \mathcal{M}_1 \times \mathcal{M}_2$ is called a *stable state transformation bisimulation*, abbreviated *SST-bisimulation*, if:

1. $M_{01}\, \mathcal{R}_{SST}\, M_{02}$;
2. if $M_1\, \mathcal{R}_{SST}\, M_2$ then
   (a) if $M_1 \stackrel{\sigma_1}{\longrightarrow'_1} M'_1 \in \mathcal{M}_1$ then there exists a transition sequence σ_2 such that $M_2 \stackrel{\sigma_2}{\longrightarrow'_2} M'_2 \in \mathcal{M}_2$, $\sigma_1 \leq_{S_o} \sigma_2$, and $M'_1\, \mathcal{R}_{SST}\, M'_2$;
   (b) as (a) but with the roles of 1 and 2 reversed.

This is an equivalence because $\leq_{S_o}$ is transitive. Only sequences leading from a stable state to another one are considered. Notice that we remove the interface places of OI-nets to define SST-bisimulation on $\mathcal{OIN}$ but S_o is present through $\leq_{S_o}$. This implies that two SST-bisimilar OI-nets behave the same in any environment.

Now, we have to introduce the replacement operation. First, if Σ is a Petri net and Σ_1 is a subnet of Σ, then the *border of* Σ_1 *in* Σ is defined by

$$bd_\Sigma(\Sigma_1) = S_1 \cap {}^\bullet(T - T_1)^\bullet.$$

Let Σ be a net, Σ_1 a subnet of Σ generated by a subset T_1 of transitions (as in Section 2.1), and Σ_2 a net such that $T_2 \cap T = \emptyset$ and $S_2 \cap S = S_2 \cap S_1 = bd_\Sigma(\Sigma_1)$. Define the net $\Sigma[\Sigma_1/\Sigma_2]$ by

$$\Sigma[\Sigma_1/\Sigma_2] = ((S - S_1) \cup S_2, (T - T_1) \cup T_2, (F - F_1) \cup F_2, (W - W_1) \cup W_2)$$

and call it the *c-replacement of* Σ_1 *by* Σ_2 *into* Σ.

Let $\psi = (\Sigma, S_o, \mathcal{M}, M_0) \in \mathcal{OIN}_{S_o}$ and Σ_1 be a subnet of Σ generated by a subset of transitions $T_1 \subseteq T$ and such that $bd_\Sigma(\Sigma_1) \subseteq S_o$. Denote $S_{i0} = S_i - S_{i1}$. The *OI-net generated by* T_1, as a subnet of ψ, is defined by $\psi(T_1) = (\Sigma_1, S_{o1}, \mathcal{M}_1, M_{01})$ where:

- $S_{o1} = S_o \cap S_1$ and $S_{i1} = S_1 - S_{o1}$;
- $\mathcal{M}_1 = \mathcal{M}|_{S_{i1}}$;
- $M_{01} = M_0|_{S_{i1}}$;

- $\mathcal{M} = ((\mathcal{M}|_{S_{i0}})|^{S_i}) \cap (\mathcal{M}_1|^{S_i})$.

We will denote by $SN(\psi)$ the set of all such subnets of ψ.

Definition 3.1.10 Let $\psi \in \mathcal{OIN}$, $\psi_1 \in SN(\psi)$ and $\psi_2 \in \mathcal{OIN}$ such that $\Sigma[\Sigma_1/\Sigma_2]$ is defined. Then the *c-replacement of* ψ_1 *by* ψ_2 *in* ψ yields an OI-net $\psi' = \psi[\psi_1/\psi_2] = (\Sigma', S'_o, \mathcal{M}', M'_0)$ defined by:

- $\Sigma' = \Sigma[\Sigma_1 \leftarrow \Sigma_2]$;
- $S'_o = S_o$ (and $S'_i = S_{i0} \cup S_{i2}$);
- $\mathcal{M}' = ((\mathcal{M}|_{S_{i0}})|^{S'_i} \cap (\mathcal{M}_2|^{S'_i})$;
- $M'_0(s) = \begin{cases} M_0(s), & \text{if } s \in S_{i0} \\ M_{02}(s), & \text{if } s \in S_{i2}. \end{cases}$

The following theorem is the fundamental result in [10].

Theorem 3.1.7 ([10]) If $\psi_2 \equiv_{SST} \psi_1$ then $\psi[\psi_1/\psi_2] \equiv_{SST} \psi$.

This is an important result, but it has a main drawback: the SST-bisimulation is a very strong one. It relies on $\leq_{S_o}$ and it is not clear whether or not this bisimulation is decidable.

Conclusions We think that what we have presented so far in this section have provided the reader with a good view about refinement techniques in Petri net theory. As we will see in the next section, all these refinement techniques are particular cases of the replacement operation we are going to define there. One of the main difference between our study and the results presented above consists of the fact that we are not interested in preserving boundedness or liveness but process semantics and partial word semantics.

3.2 Concurrent Behavior and Replacements

As we already saw in Section 3.1, many transformations of Petri nets can be simply described by "replace the subnet γ_1 of γ by the net γ_2"; this means the subnet γ_1 will be removed from γ and the net γ_2 is inserted in its place (denote the result by $\gamma[\gamma_1 \leftarrow \gamma_2]$). When γ_2 is "more detailed" than γ_1, this operation is usually called a *refinement*; otherwise it is called an *abstraction*. Both of them are particular cases of *replacement*.

One of the main problem in connection with replacement is the following:

- find some equivalence relations on nets, $\approx_1$ and $\approx_2$, such that $\gamma_1 \approx_1 \gamma_2$ implies $\gamma \approx_2 \gamma[\gamma_1 \leftarrow \gamma_2]$.

In literature, mostly refinement was studied. Various techniques and a large number of behavior and equivalence relations preserving refinement (as above) have been proposed [63, 47, 38, 65, 20, 29, 7, 8, 66, 10, 51].

In this section we develop a technique of replacement based on subnets generated by subsets of transitions. The technique was sketched in [51], where a slightly different formalism has been used and no proof has been provided. The details have been completed in [56, 58, 62], where more new results have been also added.

A fundamental requirement of this technique consists in the fact that only subnets generated by subsets of transitions will be replaced. As a conclusion, the replacement operation can be defined by

$$\gamma[\gamma_1 \leftarrow \gamma_2] = (\gamma - \gamma_1) \circ \gamma_2,$$

where $(\gamma - \gamma_1), \gamma_2 \in PN(S^c, M_0^c)$ are compatible nets, and S^c is the set of interface places between $\gamma - \gamma_1$ and γ_1 (that is, $\gamma = (\gamma - \gamma_1) \circ \gamma_1$).

The equivalence relations we consider are based on two main concepts: *isomorphism of processes* and *partial words*.

Definition 3.2.1 Let $\pi_1 = (N_1, p_1)$ and $\pi_2 = (N_2, p_2)$ be processes (not necessary of the same net). π_1 and π_2 are called *isomorphic*, abbreviated $\pi_1 \cong \pi_2$, if there is a bijection $\varphi : B_1 \cup E_1 \rightarrow B_2 \cup E_2$ such that:

(1) $p_1(x) = p_2(\varphi(x))$, for all $x \in E_1 \cup \{x \in B_1 | p_1(x) \in S_1 \cap S_2 \vee p_2(\varphi(x)) \in S_1 \cap S_2\}$;

(2) $x \prec_{\pi_1} y$ iff $\varphi(x) \prec_{\pi_2} \varphi(y)$, for all $x, y \in B_1 \cup E_1$.

As we can see our notion of isomorphism of processes is a generalization of the classical one (see the footnote in Section 2.3): it is an isomorphism of occurrence nets preserving all the condition-labels that the underlying nets have in common, and all the event-labels. It does not yield in general an equivalence relation because transitivity is not assured. For example, the processes in Figure 3.6 satisfy: $\pi_1 \cong \pi_2$, $\pi_2 \cong \pi_3$, but π_1 and π_3 are not isomorphic (assuming that $S_1 = \{s_1, s_2\}$, $S_2 = \{s_2, s_3\}$, and $S_3 = \{s_1, s_3\}$). This lack of transitivity should not be regarded as a weakness because our method will be applied to classes of nets where transitivity is satisfied.

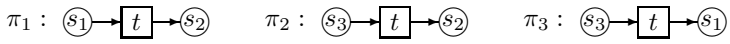


Figure 3.6: The isomorphism relation in Definition 3.2.1 is not transitive

The reason for choosing such a notion of isomorphism can be explained briefly as follows (it will become very clear in Chapter 4 where several Petri net transformations are discussed). Generally speaking, we are interested in replacing a subnet γ_1 of a net γ by a net γ_2 which has some structural properties that γ_1 does not have. Moreover, we want to preserve the behavior of the net γ (that is, we want to have $\gamma \approx \gamma[\gamma_1 \leftarrow \gamma_2]$, for some equivalence relation $\approx$). In many cases, γ_2 is obtained from γ_1 by introducing new places or transitions which are copies of some places and transitions of γ_1. Usually, copies of a transition are labeled as the transition is; places are not labeled but it is not important at all whether we use a place s or a copy of it. From this point of view, isomorphisms which associate to conditions labeled by places s conditions labeled by copies of s, prove to be suitable enough in order to deal with such transformations.

In order to define the partial word associated to a process we have to derive from processes another structure by recording only the events which are not labeled by λ.

Definition 3.2.2 Let $\pi = (N, p)$ be a process of a net γ.

1. A labeled partially ordered set [1] (E', A, p') is called an *abstraction* of π if the following properties hold:
   - $E' = \{e \in E | p(e) \neq \lambda\}$;
   - $(e, e') \in A$ iff there is a path in π leading from e to e';
   - $p' = p|_{E'}$.
2. The equivalence class with respect to isomorphism (of labeled partially ordered sets [2]) induced by (E', A, p'), denoted by $PW(\pi)$, is called the

[1] A labeled partially ordered set is a triple $(X, \leq, p)$, where $(X, \leq)$ is a partially ordered set and p is a mapping from X into a set V.

[2] Two labeled partially ordered sets $(X_i, \leq_i, p_i)$, $i = 1, 2$, are called isomorphic if there is a bijection $f : X_1 \rightarrow X_2$ such that:

- $(\forall x, y \in X_1)(x \leq_1 y \quad \Leftrightarrow \quad f(x) \leq_2 f(y))$;
- $(\forall x \in X_1)(p_1(x) = p_2(f(x)))$.

partial word associated to π.

The set of all partial words of γ is denoted by $PW(\gamma)$.

Definition 3.2.3 Let γ_1 and γ_2 be two nets.

1. γ_1 and γ_2 are called *process equivalent*, abbreviated $\gamma_1 \approx_P \gamma_2$, if for each process π_1 of γ_1 there is a process π_2 of γ_2 such that $\pi_1 \cong \pi_2$, and vice versa.

2. γ_1 and γ_2 are called *partial word equivalent*, abbreviated $\gamma_1 \approx_{PW} \gamma_2$, if $PW(\gamma_1) = PW(\gamma_2)$.

Thus, we have obtained two binary relations on nets, $\approx_P$ (process equivalence) and $\approx_{PW}$ (partial word equivalence). The relation $\approx_{PW}$ is always an equivalence relation, but not necessarily $\approx_P$; however, on sets $A \subseteq PN(S^c, M_0^c)$ of pairwise compatible nets it is an equivalence too. From this reason we will refer to both relations as equivalence relation; this one does not induce "unwanted" consequences due to the fact that, whenever it is necessary, we will assume that the nets to be considered are pairwise compatible.

The relations $\approx_P$ and $\approx_{PW}$ will play the role of $\approx_2$ from the beginning of this section; the substitutes for $\approx_1$ are just to be defined.

From an intuitive point of view, partial words of $\gamma[\gamma_1 \leftarrow \gamma_2]$ can be obtained from abstractions of processes of γ by removing from them abstractions of processes of γ_1, and "inserting" back isomorphic abstractions of processes of γ_2. However, the insertion operation needs some "sockets" and these will be modeled by conditions labeled by interface places. Thus, we introduce the concept of an S^c-abstraction of a process as follows.

Definition 3.2.4 Let γ be a net, $S^c \subseteq S$, and π a process of γ.

1. A labeled partially ordered set $(B' \cup E', A, p')$ is called an *S^c-abstraction* of π if the following properties hold:

   - $B' \subseteq \{b \in B | p(b) \in S^c \ \wedge \ (|{}^\bullet b| = 0 \ \vee \ |b^\bullet| = 0)\}$, and $E' = \{e \in E | p(e) \neq \lambda\}$;
   - $(x, y) \in A$ iff there is a path in π leading from x to y;
   - $p' = p|_{B' \cup E'}$.

2. The equivalence class with respect to isomorphism induced by $(B' \cup E', A, p')$, denoted by S^c-$PW(\pi)$, is called the S^c-*partial word* associated to π.

The class of all S^c-partial words of γ is denoted by S^c-$PW(\gamma)$.

Now, we are in a position to introduce two new relations which will play the role of $\approx_1$.

Definition 3.2.5 Let $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$.

1. γ_1 and γ_2 are called *m-process equivalent*, abbreviated $\gamma_1 \approx_{mP} \gamma_2$, if $(\gamma_1 + M) \approx_P (\gamma_2 + M)$, for any marking M on S^c.

2. The nets γ_1 and γ_2 are called *m-partial word equivalent*, abbreviated $\gamma_1 \approx_{mPW} \gamma_2$, if S^c-$PW(\gamma_1 + M)$=S^c-$PW(\gamma_2 + M)$, for any marking M on S^c.

This definition says that no matter how the initial marking on S^c is increased these two nets have the same processes (up to an isomorphism) or the same S^c-partial words. It is easy to see that $\approx_{mPW}$ is an equivalence relation on nets, but not necessarily $\approx_{mP}$; the same remark as for process equivalence holds true in this case as well.

In order to simplify the notation we will use in what follows $\approx_m$ to denote one of the relations $\approx_{mP}$ or $\approx_{mPW}$, and $\approx$ to denote $\approx_P$ or $\approx_{PW}$. Moreover, whenever we use both $\approx_m$ and $\approx$, and $\approx_m$ denotes $\approx_{mP}$ ($\approx_{mPW}$), then $\approx$ will denote $\approx_P$ ($\approx_{PW}$).

Let $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$, $M \in \mathbf{N}^{S^c}$ and $\overset{M}{\approx}_m$ be the binary relation

$$\gamma_1 \overset{M}{\approx}_m \gamma_2 \quad \text{iff} \quad (\gamma_1 + M) \approx_m (\gamma_2 + M).$$

That is, $\gamma_1 \overset{M}{\approx}_m \gamma_2$ iff $(\gamma_1 + M') \approx (\gamma_2 + M')$ for all $M' \in \mathbf{N}^{S^c}$ with $M \leq M'$.

Directly from definitions we obtain:

Proposition 3.2.1 ([51, 56, 58, 62]) $\approx_m \subseteq \overset{M}{\approx}_m \subseteq \overset{M'}{\approx}_m$, for all $M, M' \in \mathbf{N}^{S^c}$ with $M \leq M'$.

When M is the zero-marking, the proposition above leads to $\approx_m \subseteq \approx$. It is also worth to mention that, for every two nets $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$, every $s \in S^c$ and $s_1, s_2 \in \mathcal{S} - (S_1 \cup S_2)$, $\gamma_1 \approx_m \gamma_2$ implies $\gamma_1' \approx_m \gamma_2'$, where γ_1' (γ_2') is obtained from γ_1 (γ_2) replacing s by s_1 (s_2).

The next theorem represents the main result of this section.

Theorem 3.2.1 ([51, 56, 58, 62]) Let $\gamma_0, \gamma_1, \gamma_2 \in PN(S^c, M_0^c)$. If γ_0 is compatible with both γ_1 and γ_2, then $\gamma_1 \approx_m \gamma_2$ implies $\gamma_0 \circ \gamma_1 \approx \gamma_0 \circ \gamma_2$.

Proof Let γ_0, γ_1 and γ_2 as in theorem. Consider the next two cases.

Case 1: $\gamma_1 \approx_{mP} \gamma_2$. We will show that for each process π of $\gamma = \gamma_0 \circ \gamma_1$ there is a process π' of $\gamma' = \gamma_0 \circ \gamma_2$ such that π and π' are isomorphic. Let π be a process of γ. From the process decomposition theorem it follows that there are two markings $M', M'' \in \mathbf{N}^{S^c}$ and two processes $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$ such that $\pi = \pi_0 \circ \pi_1$ (the composition of processes is along a set B^c of common conditions – see the proof of the process decomposition theorem).

By hypothesis, there is a process $\pi_2 \in \Pi(\gamma_2 + M'')$ such that $\pi_1 \cong \pi_2$. Let f be an isomorphism from π_1 to π_2. The identity function ι_{B^c} on B^c is a p-composition function from π_0 to π_1 compatible with M' and M''. Clearly, $f|_{B^c} \circ \iota_{B^c}$ is a p-composition function from π_0 to π_2 compatible with M' and M''. Then, by the process composition theorem it follows that there is a process $\pi_2' \in \Pi(\gamma_2 + M'')$, isomorphic to π_2 and such that $\pi' = \pi_0 \circ \pi_2'$ is a process of $\gamma_0 \circ \gamma_2$. Moreover, it is easily seen that $\pi = \pi_0 \circ \pi_1 \cong \pi_0 \circ \pi_2' = \pi'$.

Case 2: $\gamma_1 \approx_{mPW} \gamma_2$. This case is quite similar to the previous one and so we will sketch only the main idea. Let π be a process of $\gamma_0 \circ \gamma_1$. Split π as in the first case and consider α_0 and α_1 two S^c-abstractions associated to π_0 and π_1, respectively, such that the only conditions these abstractions contain are exactly those from B^c. By hypothesis, there is an S^c-abstraction α_2 of γ_2, isomorphic to α_1. Hence, there is a process π_2 of γ_2 such that α_2 is an S^c-abstraction of it. Compose π_0 and π_2 along B^c as in the first case and let π' be the result (clearly, we may assume that π_0 and π_2 have in common only the conditions in B^c). One can easily prove that $\pi' \in \Pi(\gamma_1 \circ \gamma_2)$ and $PW(\pi) = PW(\pi')$. □

Corollary 3.2.1 ([51, 56, 58, 62]) Let $\gamma_0, \gamma_1, \gamma_2 \in PN(S^c, M_0^c)$. If γ_0 is compatible with both γ_1 and γ_2, then $\gamma_1 \approx_m \gamma_2$ implies $\gamma_0 \circ \gamma_1 \approx_m \gamma_0 \circ \gamma_2$. Therefore, $\approx_{mP}$ and $\approx_{mPW}$ are congruences on sets $A \subseteq PN(S^c, M_0^c)$ of pairwise compatible nets, w.r.t. the composition along S^c.

Proof Assume $\gamma_1 \approx_m \gamma_2$. From Proposition 3.2.1 it follows that

$$(\gamma_1 + M) \approx_m (\gamma_2 + M),$$

and by Theorem 3.2.1 we have

$$(\gamma_0 + M) \circ (\gamma_1 + M) \approx (\gamma_0 + M) \circ (\gamma_2 + M),$$

for all $M \in \mathbf{N}^{S^c}$. Hence,

$$((\gamma_0 \circ \gamma_1) + M) \approx ((\gamma_0 \circ \gamma_2) + M)$$

for all $M \in \mathbf{N}^{S^c}$, which shows that $\gamma_0 \circ \gamma_1 \approx_m \gamma_0 \circ \gamma_2$.

The relations $\approx_{mP}$ and $\approx_{mPW}$ are equivalences on sets $A \subseteq PN(S^c, M_0^c)$ of pairwise compatible nets. In the view of commutativity and of the first part of this corollary we obtain that these relations are congruences on A. □

Corollary 3.2.2 ([51, 56, 58, 62]) Let γ_1 be a subnet of a net γ, S^c be the set of interface places between $\gamma - \gamma_1$ and γ_1, and let M_0^c be the restriction of the initial marking of γ to the set S^c. Then, for every net $\gamma_2 \in PN(S^c, M_0^c)$ compatible with $\gamma - \gamma_1$, $\gamma_1 \approx_m \gamma_2$ implies $\gamma \approx \gamma[\gamma_1 \leftarrow \gamma_2]$.

Proof Let $\gamma_0 = \gamma - \gamma_1$. Then, $\gamma_0, \gamma_1, \gamma_2 \in PN(S^c, M_0^c)$, γ_0 is compatible with both γ_1 and γ_2, $\gamma = \gamma_0 \circ \gamma_1$, and $\gamma[\gamma_1 \leftarrow \gamma_2] = \gamma_0 \circ \gamma_2$. The corollary follows now from Theorem 3.2.1. □

3.3 Process Stable Petri Nets

From a practical point of view we are interested in Corollary 3.2.2. The difficulty in using this corollary consists in the fact that we have to decide whether or not $\gamma_1 \approx_m \gamma_2$; that is, we have to decide whether or not $(\gamma_1 + M) \approx (\gamma_2 + M)$ for all $M \in \mathbf{N}^{S^c}$. A favorable particular case would be when a marking $M \in \mathbf{N}^{S^c}$ exists such that all processes of $(\gamma + M')$, where $M' \in \mathbf{N}^{S^c}$ and $\neg(M' \leq M)$, can be reduced to processes of $(\gamma + M)$. We will discuss such a case in what follows, but first let us introduce a few concepts.

If $\pi = (N, p)$ is a labeled occurrence net and C is a subset of ${}^\circ\pi^\circ$, where ${}^\circ\pi^\circ = {}^\circ\pi \cap \pi^\circ$, then we will denote by $(\pi - C)$ the labeled occurrence net obtained from π by removing all the conditions in C.

Definition 3.3.1 Let $\gamma \in PN(S^c, M_0^c)$, M_1 and M_2 markings on S^c, $\pi_1 \in \Pi(\gamma+M_1)$, and $\pi_2 \in \Pi(\gamma+M_2)$. We say that π_1 and π_2 are *almost isomorphic*, abbreviated $\pi_1 \cong_a \pi_2$, if there are $C_1 \subseteq p_1^{-1}(S^c) \cap {}^\circ\pi_1^\circ$ and $C_2 \subseteq p_2^{-1}(S^c) \cap {}^\circ\pi_2^\circ$ such that $(\pi_1 - C_1) \cong (\pi_2 - C_2)$.

This definition wants to say that if we remove from π_1 and π_2 some conditions without predecessors and successors and labeled by places in S^c, then we get two isomorphic labeled occurrence nets.

Lemma 3.3.1 Let $\gamma \in PN(S^c, M_0^c)$, M_1 and M_2 markings on S^c, $\pi_1 \in \Pi(\gamma + M_1)$, and $\pi_2 \in \Pi(\gamma + M_2)$. Then, $\pi_1 \cong_a \pi_2$ iff $(\pi_1 - (p_1^{-1}(S^c) \cap {}^\circ\pi_1^\circ)) \cong (\pi_2 - (p_2^{-1}(S^c) \cap {}^\circ\pi_2^\circ))$.

Proof The "if" part follows directly from definitions. As for the "only if" part let us suppose that $\pi_1 \cong_a \pi_2$. Then, there is $C_1 \subseteq p_1^{-1}(S^c) \cap {}^\circ\pi_1^\circ$ and $C_2 \subseteq p_2^{-1}(S^c) \cap {}^\circ\pi_2^\circ$ such that $(\pi_1 - C_1) \cong (\pi_2 - C_2)$; let φ be an isomorphism between these two structures. The definition of isomorphism leads to the fact that

$$b \in p_1^{-1}(S^c) \cap {}^\circ(\pi_1 - C_1)^\circ \text{ iff } \varphi(b) \in p_2^{-1}(S^c) \cap {}^\circ(\pi_2 - C_2)^\circ.$$

Moreover, $p_1(b) = p_2(\varphi(b))$. Therefore,

$$(\pi_1 - C_1) - (p_1^{-1}(S^c) \cap {}^\circ(\pi_1 - C_1)^\circ) \cong (\pi_2 - C_2) - (p_2^{-1}(S^c) \cap {}^\circ(\pi_2 - C_2)^\circ)$$

by the corresponding restriction of φ, let it φ'. Hence,

$$\pi_1 - (C_1 \cup (p_1^{-1}(S^c) \cap {}^\circ(\pi_1 - C_1)^\circ)) \cong \pi_2 - (C_2 \cup (p_2^{-1}(S^c) \cap {}^\circ(\pi_2 - C_2)^\circ))$$

by φ'. As to accomplish the proof we have to remark that

$$C_i \cup (p_i^{-1}(S^c) \cap {}^\circ(\pi_i - C_i)^\circ) = p_i^{-1}(S^c) \cap {}^\circ\pi_i^\circ,$$

for $i = 1, 2$. □

Definition 3.3.2 Let $\gamma \in PN(S^c, M_0^c)$ and $M \in \mathbf{N}^{S^c}$. We say that γ is *process stable w.r.t. M* if for every marking $M' \in \mathbf{N}^{S^c}$ with $\neg(M' \leq M)$ we have:

- for any process π' of $(\gamma + M')$ there is a process π of $(\gamma + M)$ such that $\pi \cong_a \pi'$;

- vice versa.

As an example, the nets in Figure 4.3(a)(b) are process stable w.r.t. the zero-marking $M = (0, \ldots, 0)$, where $S^c = \{s_1, \ldots, s_k\}$. We have:

Theorem 3.3.1 ([56, 58, 62]) Let $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$ be two process stable nets w.r.t. a marking M on S^c. If $(\gamma_1 + M) \approx (\gamma_2 + M)$ then $\gamma_1 \approx_m \gamma_2$.

Proof We will prove the proposition only in the case $\approx = \approx_P$; the other one can be easily obtained from this one. Let $M' \in \mathbf{N}^{S^c}$. We have to show that $(\gamma_1 + M') \approx (\gamma_2 + M')$. Consider the next two cases.

Case 1: $M' \leq M$. For each process π_1' of $(\gamma_1 + M')$ there is a process π_1 of $(\gamma_1 + M)$ and $C_1 \subseteq p_1^{-1}(S^c) \cap {}^\circ\pi_1^\circ$ such that $\pi_1' = \pi_1 - C_1$. From hypothesis it follows that there is a process π_2 of $(\gamma_2 + M)$ such that $\pi_1 \cong \pi_2$. Let φ be the isomorphism between these processes and let $C_2 = \varphi(C_1)$ and $\varphi' = \varphi|_{(B_1 - C_1) \cup E_1}$. It is not difficult to see that $\pi_2' = \pi_2 - C_2$ is a process of $(\gamma_2 + M')$, and π_1' and π_2' are isomorphic by φ'.

Case 2: $\neg(M' \leq M)$. Let π_1' a process of $(\gamma_1 + M')$. The net γ_1 is process stable w.r.t. M and therefore there is a process π_1 of $(\gamma_1 + M)$ such that $\pi_1' \cong_a \pi_1$. Moreover, by Lemma 3.3.1 we have

$$(\pi_1' - ((p_1')^{-1}(S^c) \cap {}^\circ(\pi_1')^\circ)) \cong (\pi_1 - (p_1^{-1}(S^c) \cap {}^\circ\pi_1^\circ));$$

let ψ_1 be an isomorphism between these two structures. From hypothesis it follows that there is a process π_2 of $(\gamma_2 + M)$ such that $\pi_1 \cong \pi_2$, and let φ be an isomorphism between these processes. It is clear that

$$p_2^{-1}(S^c) \cap {}^\circ\pi_2^\circ = \varphi(p_1^{-1}(S^c) \cap {}^\circ\pi_1^\circ),$$

and

$$(\pi_1 - (p_1^{-1}(S^c) \cap {}^\circ\pi_1^\circ)) \cong (\pi_2 - (p_2^{-1}(S^c) \cap {}^\circ\pi_2^\circ))$$

by φ', which is the corresponding restriction of φ. Using the fact that γ_2 is process stable w.r.t. M we get a process π_2' of $(\gamma_2 + M')$ such that $\pi_2 \cong_a \pi_2'$. Lemma 3.3.1 leads to

$$(\pi_2 - (p_2^{-1}(S^c) \cap {}^\circ\pi_2^\circ)) \cong (\pi_2' - ((p_2')^{-1}(S^c) \cap {}^\circ(\pi_2')^\circ)),$$

and let ψ_2 be an isomorphism between these structures. Then, $\psi_2 \circ \varphi' \circ \psi_1$ is an isomorphism between $(\pi_1' - ((p_1')^{-1}(S^c) \cap {}^\circ(\pi_1')^\circ))$ and $(\pi_2' - ((p_2')^{-1}(S^c) \cap {}^\circ(\pi_2')^\circ))$, and it is straightforward to see that this isomorphism can be extended to an isomorphism between π_1' and π_2'. Therefore, $\pi_1' \cong \pi_2'$. □

Chapter 4

Proving Correctness of Petri Net Structural Transformations

By a structural transformation one may want to obtain a special form of a given Petri net, such as a normal form, by preserving some properties of the original net. Structural transformations are usually based on iterative replacement of various subnets of the original net. Therefore, one may expect that the correctness of such transformations be proved by Corollary 3.2.2 and Theorem 3.3.1.

We will show that this is the case indeed, by applying Corollary 3.2.2 and Theorem 3.3.1 to prove the correctness of four Petri net structural transformations aimed to produce normal forms of Petri nets. In what follows we will exemplify this by considering 4 transformations aimed to produce normal forms of Petri nets. As we will see, our proofs are very short and elegant in comparison with the original proofs of correctness mainly based on ad hoc methods (compare for example the proofs in [39], which takes many journal pages and uses techniques of graph coloring, with the proofs of Theorem 4.1.1 and 4.1.2 in Section 4.1).

4.1 The Normal Form of Petri Nets

We begin this section by showing how processes of Petri nets can be defined *inductively* in terms of composition of initial occurrence nets and elementary occurrence nets associated to transitions.

Let γ be a marked Petri net and t a transition of it. Then:

- an *elementary occurrence net* associated to t is any labeled occurrence net $\pi = (N, p)$ with the properties: π contains only one event e which is labeled by t, $W(s,t)$ preconditions and $W(t,s)$ postconditions of e labeled by s, for all $s \in S$ (and no other element);

- an *initial occurrence net* of γ is any occurrence net (N, p) which does not contain any event and, for each $s \in S$, it contains exactly $M_0(s)$ conditions labeled by s.

Now, it can be shown that the *set of processes* of γ is the smallest set $\Pi(\gamma)$ with the properties:

1. $\Pi(\gamma)$ contains all the initial occurrence nets associated to γ;

2. if $\pi_1 \in \Pi(\gamma)$ and π_2 is an elementary occurrence net associated to a transition t such that ${}^\circ\pi_2 \subseteq \pi_1^\circ$, then the composition of π_1 and π_2 along ${}^\circ\pi_2$, whenever it is defined, is in $\Pi(\gamma)$.

In case (2) we say that π_1 is *extended* (to the right) by π_2.

We want to place stress on the fact that a necessary requirement for the composition of π_1 and π_2 along ${}^\circ\pi_2$, in the definition above, is that π_1 and π_2 have in common only the conditions in ${}^\circ\pi_2$. This requirement is supplied by the words "whenever it is defined".

It is clear that for every $\pi \in \Pi(\gamma)$ there is a sequence

$$\pi_0, \pi_1, \ldots, \pi_m = \pi,$$

where π_0 is an initial occurrence net and π_{i+1} can be constructed from π_i as described above, for all $0 \leq i \leq m-1$. Processes of labeled nets are obtained by relabeling the events associated to transitions.

According to [39], a labeled marked Petri net is called *normalized* if its weight function and initial marking take values into $\{0, 1\}$. In [39] it was shown that every λ-free labeled marked Petri net (that is, λ cannot be a label) is partial word equivalent to a normalized one. Moreover an algorithm to transform such a net into an equivalent normalized one, was proposed. The algorithm works in two main steps, called **Transformation-A** and **Transformation-B**. In the first one the weight function, and in the second the initial marking, is processed. The initial marking needs to be processed in the first step too. The solution proposed for processing the initial marking was to add new places and transitions in order to "simulate" it.

This fact led to an increasing almost double in the size of the produced net (in terms of places, transitions, and arcs). In [50] has been pointed out that the normalization algorithm can also be applied to labeled nets and, further, in [51] has been noted that the normalization preserves the processes as well if one consider the notion of isomorphism we already adopted. Moreover, in [54] another solution for processing the initial marking was proposed. It consists of a distribution of the initial marking into the old places. The size of the produced net is to the half reduced. Therefore, we will describe the normalization algorithm taking into account the solution proposed in [54] for processing the initial marking.

Definition 4.1.1 ([54]) Let $\gamma = (\Sigma, M_0, l)$ be a net, $S_1 \subseteq S$, and M be a marking of γ. We say that M is *uniformly distributed over* S_1 if $|M(s_1) - M(s_2)| \leq 1$ for all $s_1, s_2 \in S_1$. Now, using the replacement operation we can describe the normalization algorithm as follows.

Transformation-A: Let $\gamma = (\Sigma, M_0, l)$ be a net and

$$n_s = max\{W(s,t), W(t,s) | t \in T\},$$

for all $s \in S$. Replace recursively the subnets γ_s generated by $T_s = {}^\bullet s^\bullet$, where s is a place with $n_s > 1$, by γ'_s defined as follows:

- $\gamma'_s = (\Sigma'_s, M'_s, l'_s)$, $\Sigma'_s = (S'_s, T'_s, F'_s, W'_s)$;
- $S'_s = (S_s - \{s\}) \cup C(s)$, where $C(s) = \{s^1, \ldots, s^{n_s}\}$ is a set of n_s new places (copies of the place s);
- $T'_s := \bigcup_{t \in {}^\bullet s^\bullet} C(t)$, where

  $$C(t) = \{t_{A,B} | A, B \subseteq C(s) \ \wedge \ |A| = W(s,t) \ \wedge \ |B| = W(t,s)\}$$

  is a set of new transitions (copies of the transition t), for each $t \in {}^\bullet s^\bullet$;
- $F' := F_1 \cup F_2$, where:

$$\begin{array}{rcl} F_1 & = & \{(s', t_{A,B}) | t_{A,B} \in C(t) \ \wedge \ t \in {}^\bullet s^\bullet \ \wedge \ s' \neq s \ \wedge \ (s', t) \in F\} \cup \\ & & \{(t_{A,B}, s') | t_{A,B} \in C(t) \ \wedge \ t \in {}^\bullet s^\bullet \ \wedge \ s' \neq s \ \wedge \ (t, s') \in F\} \\ F_2 & = & \{(s', t_{A,B}) | t_{A,B} \in C({}^\bullet s^\bullet) \ \wedge \ s' \in A\} \cup \\ & & \{(t_{A,B}, s') | t_{A,B} \in C({}^\bullet s^\bullet) \ \wedge \ s' \in B\}, \end{array}$$

and $C(^\bullet s^\bullet)$ is the union-extension of $C(\cdot)$ to the set $^\bullet s^\bullet$;

- W'_s is given by:

$$\begin{array}{lcl} W'_s(s', t_{A,B}) & = & W(s', t) \text{ for all } (s', t_{A,B}) \in F_1, \\ W'_s(t_{A,B}, s') & = & W(t, s') \text{ for all } (t_{A,B}, s') \in F_1, \\ W'_s(f) & = & 1 \text{ for all } f \in F_2; \end{array}$$

- $M'_s|_{C(s)}$ is an arbitrary but fixed uniformly distributed marking over $C(s)$ such that $M_0(s) = \sum_{s' \in C(s)} M'_s(s')$, and $M'_s(s') := M_0(s')$ for all $s' \in S_s - \{s\}$;

- $l'_s(t_{A,B}) = l(t)$ for all $t_{A,B} \in T'_s$.

This transformation is exemplified in Figure 4.1.

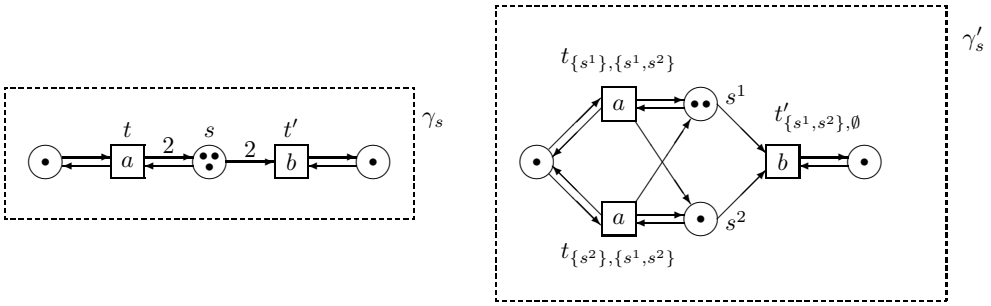


Figure 4.1: Example of Transformation-A

The following theorem has been proved in [54], but the proof given here is certainly more elegant.

Theorem 4.1.1 ([54]) The net γ' yielded by Transformation-A on the input γ satisfies $\gamma \approx_P \gamma'$.

Proof In the view of Corollary 3.2.2 we have to prove that $\gamma_s \approx_{mP} \gamma'_s$, for all s with $n_s > 1$. By the inductive definition of process it suffices to show that, for every marking M on the interface places, the following properties hold true:

- each initial occurrence net of $(\gamma_s + M)$ is isomorphic with each initial occurrence net of $(\gamma'_s + M)$;

- each elementary occurrence net of $(\gamma_s + M)$ associated to a transition t is isomorphic with each elementary occurrence net of $(\gamma'_s + M)$ associated to any copy of t;

- if π and π' are isomorphic processes of $(\gamma_s + M)$ and $(\gamma'_s + M)$, respectively, and the process π is extended by an elementary occurrence net associated to a transition t, then π' can be extended by an elementary occurrence net associated to a copy of t and, moreover, the processes obtained in this way are isomorphic. Vice versa, if π' is extended by an elementary occurrence net associated to a copy of a transition t, then π can be extended by an elementary occurrence net associated to t and, moreover, the processes obtained in this way are isomorphic.

All these facts follow directly from the definition of γ'_s. □

Transformation-B: Let $\gamma = (\Sigma, M_0, l)$ be a net such that $W(f) = 1$ for all $f \in F$. Let $m_s = M_0(s)$, for all $s \in S$. Replace recursively the subnets γ_s generated by $T_s = {}^\bullet s^\bullet$, where s is a place with $m_s > 1$, by γ'_s defined as follows:

- $\gamma'_s = (\Sigma'_s, M'_s, l'_s)$, $\Sigma'_s = (S'_s, T'_s, F'_s, W'_s)$;

- $S'_s = (S_s - \{s\}) \cup C(s)$, where $C(s) = \{s^1, \ldots, s^{m_s}\}$ is a set of new places (copies of the place s);

- $T'_s := \bigcup_{t \in {}^\bullet s^\bullet} C(t)$, where $C(t) = \{t^1, \ldots, t^{m_s}\}$ is a set of new transitions (copies of the transition t), for each $t \in {}^\bullet s^\bullet$;

- $F'_s := F_1 \cup F_2$, where:

$$\begin{array}{rcl} F_1 & = & \{(s', t^i) | t^i \in T'_s \ \wedge \ s' \in S^c_s \ \wedge \ (s', t) \in F\} \cup \\ & & \{(t^i, s') | t^i \in T'_s \ \wedge \ s' \in S^c_s \ \wedge \ (t, s') \in F\}, \\ F_2 & = & \{(s^i, t^i) | t^i \in T'_s \ \wedge \ s^i \in C(s) \ \wedge \ (s, t) \in F\} \cup \\ & & \{(t^i, s^i) | t^i \in T'_s \ \wedge \ s^i \in C(s) \ \wedge \ (t, s) \in F\}; \end{array}$$

- $W'_s(f) = 1$ for all $f \in F'_s$;

- $M'_s(s^i) = 1$ for all $1 \leq i \leq m_s$, and $M'_s(s') := M_0(s')$ for all $s' \in S_s - \{s\}$;

- $l'_s(t^i) = l(t)$ for all $t^i \in T'_s$.

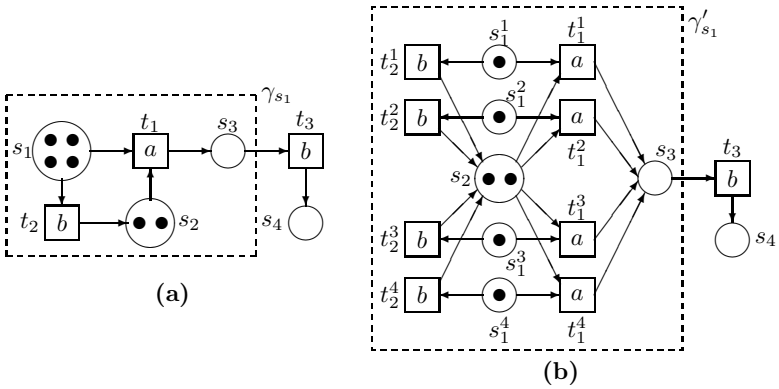


Figure 4.2: Example of Transformation-B applied to s_1

This transformation is exemplified in Figure 4.2 for the case of the place s_1. Clearly, the net yielded by Transformation-B is normalized.

Theorem 4.1.2 ([56, 62]) The net γ' yielded by Transformation-B on the input γ satisfies $\gamma \approx_P \gamma'$.

Proof Similar arguments as those in the proof of the theorem above work in this case too. □

The above two theorems assure the correctness of the normalization algorithm. One may compare the elegance of this solution with the proof in [39].

4.2 The Super Normal Form of Petri Nets

In [43], a systematic investigation of graph theoretic properties of Petri nets within the framework of language theory was initiated. In other words, various subclasses of Petri nets were introduced by imposing various restrictions on the in- and out- degree of nodes in the graph of the underlying net structure. Further these restrictions were refined in [50] by considering (n, m)-*transition restricted Petri nets* as being Petri nets for which the weight function takes values in $\{0, 1\}$ and $1 \leq |{}^\bullet t| \leq n$ and $1 \leq |t^\bullet| \leq m$ for all transitions t. Thus, interesting hierarchies of Petri net languages were obtained, and in the case of λ-labeled Petri nets, the normal form was improved with respect to the finite transition sequence behavior. More precisely, it was

shown that every λ-labeled Petri net is equivalent to a $(2,2)$-transition restricted net (with respect to the finite transition sequence behavior). This result was extended in [52] by showing that this new normal form, called the *super-normal form*, preserves the partial words but not the processes. We will give here short proofs of these results. Let us recall first the basic transformations.

Let γ be a labeled net. In the view of the results above we may assume that γ is normalized. Now we have to do two basic transformations on γ.

Transformation-C: Let γ be a normalized net. Replace recursively the subnets γ_t generated by t, where t is a transition such that $|{}^\bullet t| = 0$ or $|t^\bullet| = 0$, by $\gamma'_t = (\Sigma'_t, M'_t, l'_t)$ defined as follows:

- if Σ_t is the net in Figure 4.3(a) then Σ'_t is the net in Figure 4.3(b);
- if Σ_t is the net in Figure 4.3(c) then Σ'_t is the net in Figure 4.3(d);
- the initial marking of γ'_t on $s_1, \ldots, s_k$ is the same as the initial marking of γ on these places;
- the labeling is that specified in diagrams.

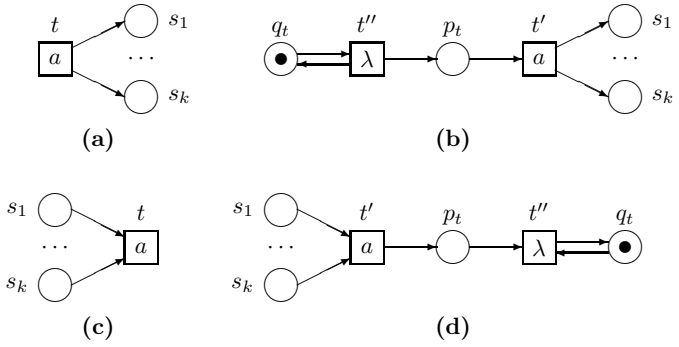


Figure 4.3: Transformation-C

It is clear that the net γ' yielded by Transformation-C is normalized and satisfies $|{}^\bullet t| \geq 1$ and $|t^\bullet| \geq 1$ for all transitions t.

Theorem 4.2.1 ([52, 56, 62]) The net γ' yielded by Transformation-C on the input γ satisfies $\gamma \approx_{PW} \gamma'$.

Proof In the view of Corollary 3.2.2 we have to prove that $\gamma_t \approx_{mPW} \gamma'_t$ for all t with the property $|{}^\bullet t| = 0$ or $|t^\bullet| = 0$, which is straightforward (for the nets in Figure 4.3(a)(b) one may use Theorem 3.3.1). □

Transformation-D: Let γ be a normalized net satisfying $|{}^\bullet t| \geq 1$ and $|t^\bullet| \geq 1$ for all $t \in T$. Replace recursively the subnets γ_t generated by t, where t is a transition such that $|{}^\bullet t| \geq 3$ or $|t^\bullet| \geq 3$, by γ'_t as given in Figure 4.4, but with the next remarks:

- for $n = 1$ the place s_1 is directly connected to t^n, and for $n = 2$ the places s_1 and s_2 are directly connected to t^n;
- for $m = 1$ the only successor of t^n is s_{n+1}, and for $m = 2$ the only successors of t^n are s_{n+1} and s_{n+2};
- the initial marking of γ'_t on the places in S is the same as the initial marking of γ on these places, and it is 0 for the other places;
- the labeling is that specified in diagram

(we explicitly mention that exactly one transition in the net in Figure 4.4 is labeled by a. Moreover, it is assumed that ${}^\bullet t = \{s_1, \ldots, s_n\}$, $t^\bullet = \{s_{n+1}, \ldots, s_{n+m}\}$, $s'_1, \ldots, s'_{n+m-3}$ are new places, and $t^1, \ldots, t^{n+m-2}$ are new transitions).

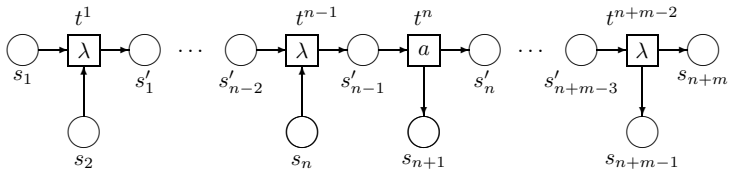


Figure 4.4: Transformation-D

It is clear that the net γ' yielded by Transformation-D is normalized and $(2, 2)$-transition restricted, and the proof of the next theorem can be easily completed.

Theorem 4.2.2 ([52, 56, 62]) The net γ' yielded by Transformation-D on the input γ satisfies $\gamma \approx_{PW} \gamma'$.

We want to place stress again on the simplicity and elegance of the correctness proofs of these transformations in comparison with the original ones. Their efficiency depends directly on the easiness of deciding the equivalences $\approx_{mP}$ and $\approx_{mPW}$. Intuitively, the simpler are γ_1 and γ_2 the easier we can check $\gamma_1 \approx_{mP} \gamma_2$ and $\gamma_1 \approx_{mPW} \gamma_2$ and, therefore, the more efficient we can apply Corollary 3.2.2 and Theorem 3.3.1.

4.3 Limitations of our Proof Technique

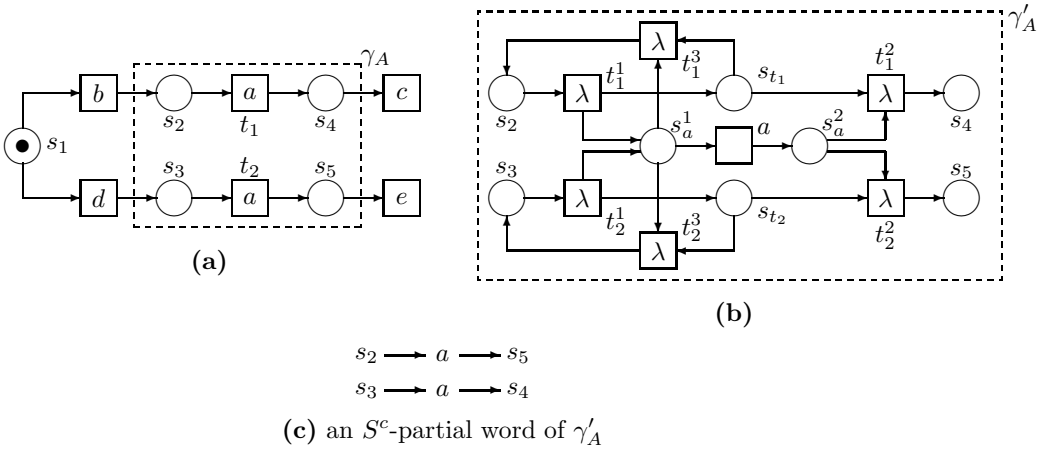
As one could expect, our method cannot be used to prove correctness of any Petri net structural transformation. We will show this by considering a transformation proposed in [8] (**Transformation-E** below). This transformation is in connection with *Petri nets with λ-transitions*, abbreviated λ-PTN, which are labeled Petri nets whose labeling function l has the property: for each transition t, $l(t)$ equals t or λ (in [8] such nets were called *strictly labeled nets*).

Transformation-E: Let $\gamma = (\Sigma, M_0, l)$ be a net and $A = \{t \in T | l(t) = a\}$, for all $a \in l(T) - \{\lambda\}$ such that at least two distinct transitions are labeled by a. Replace recursively the subnets γ_A generated by A, where A is as above, by γ'_A defined as follows:

- $\gamma'_A = (\Sigma', M'_0, l')$, $\Sigma' = (S', T', F', W')$;
- $S' = S_A \cup \{s^1_a, s^2_a\} \cup \{s_t | t \in A\}$, where S_A is the set of places of γ_A;
- $T' = \{a\} \cup \{t^1, t^2, t^3 | t \in A\}$;
- $W'(s, t^1) = W(s, t)$, for all $s \in S_A$ and $t \in A$,
  $W'(t^2, s) = W(t, s)$, for all $s \in S_A$ and $t \in A$,
  $W'(t^3, s) = W(s, t)$, for all $s \in S_A$ and $t \in A$,
  $W'(t^1, s') = 1$, if $s' = s_t$ or $s' = s^1_a$, for all $t \in A$,
  $W'(s', t^2) = 1$, if $s' = s_t$ or $s' = s^2_a$, for all $t \in A$,
  $W'(s^1_a, a) = W'(a, s^2_a) = 1$,
  $W'(s^1_a, t^3) = W'(s_t, t^3) = 1$, for all $t \in A$,
  $W'(x, y) = 0$, otherwise
  (this defines both F' and W');
- $M'_0(s') = M_0(s')$ for all $S' \in S_A$, and $M'_0(s') = 0$ for all $s' \in S' - S_A$;

- $l'(a) = a$ and $l'(t') = \lambda$ for all $t' \neq a$.

This transformation is exemplified in Figure 4.5(a)(b).



$s_2 \longrightarrow a \longrightarrow s_5$

$s_3 \longrightarrow a \longrightarrow s_4$

(c) an S^c-partial word of γ'_A

Figure 4.5: Example of Transformation-E

In [8] it was shown that if γ is without auto-concurrency (no two transitions, not necessarily distinct, of the same set A – as in Transformation-E – may be concurrently enabled) and multiple need (each transition in each set A – as in Transformation-E – only needs one token) then γ and γ' are fully concurrent bisimilar (γ' being the net yielded by Transformation-E); therefore, $\gamma \approx_{PW} \gamma'$ (see [8] for more details).

The correctness of Transformation-E cannot be proved by Corollary 3.2.2. Indeed, the nets γ_A and γ'_A in Figure 4.5(a)(b) (γ_A is without auto-concurrency and multiple need) are not $\approx_{mPW}$-related because the labeled partially ordered set in Figure 4.5(c) is an S^c-partial word of $(\gamma'_A + (1,1,0,0))$ but not of $(\gamma_A + (1,1,0,0))$ ($(1,1,0,0)$ is a marking on s_2, s_3, s_4, s_5).

As a conclusion, a more deeper insight on the nature of these transformations should be achieved. Concerning Petri nets with λ-transitions we can prove that, in general, there is no transformation of labeled nets into partial word equivalent λ-PTN's.

Proposition 4.3.1 ([51, 56, 62]) There is a labeled net which is not partial word equivalent to any net with λ-transitions.

Proof Consider the labeled net γ in Figure 4.5(a) with the difference that its initial marking contains 2 tokens in s_1 and no one in the other places.

Suppose by contradiction that there is a λ-PTN γ' such that $PW(\gamma) = PW(\gamma')$. Consider a process π of γ obtained by applying all six transitions of γ in an arbitrary but fixed way. $PW(\pi)$ will have two distinct paths b, a, c and d, a, e. $PW(\pi)$ is a partial word of γ' as well, and hence there is a process $\pi' = (N', p')$ of γ' such that $PW(\pi') = PW(\pi)$. There are the events $e_1, \ldots, e_6$ of π' labeled respectively by b, d, a, a, c, e, and such that there are paths from e_1 to e_3, from e_3 to e_5, from e_2 to e_4, and from e_4 to e_6. There is no path between the events e_3 and e_4 and, they being labeled by the same transition a, there are label-preserving bijections between their sets of postconditions; let h be such a bijection ($h : e_3^\bullet \to e_4^\bullet$). Now define a new process of γ' by interchanging the arcs starting from b' and $h(b')$, for all postconditions b' of e_3 (that is, the arc starting from b' will be transformed into an arc starting from $h(b')$ but with the same end as b', and vice versa).

It is easy to see that the procedure above defines a process π'' of γ'. Moreover, the partial word associated to π'' contains a path d, a, c but no partial word of γ contains such a path. Therefore, $PW(\pi'') \notin PW(\gamma)$; a contradiction. □

Chapter 5

Environmental Petri Net Reactive Modules

Environmental Petri Net Reactive Modules (*environmental modules* or *e-modules*, for short) [56, 62] are couples between a module and an environment specification that acts on the interface places of that module. E-modules are very important in that that they can be used as abstractions of Petri net models (chapters 6 and 7 focus on verification techniques based on e-modules).

5.1 Definitions and Examples

A module has a distinguished subset of places used to interact with an environment. But, what is an environment? In order to answer this question let us look again to the nets in Figure 2.7. From π_0's point of view (in the context of γ), the conditions b_{10} and b_{11} have been "pumped" by the environment (by π_1). However, for these two conditions, γ_0 has to pay by the condition b_7 (labeled by s_1). More precisely, γ_0 gives to γ_1 a condition labeled by s_1 and receives two conditions labeled by s_1 and s_3. This exchange of conditions can be formally described by the ordered pair $r_0 = ((1,0,1),(1,0,1))$. It says that whenever the configuration on the interface places $\{s_1, s_2, s_3\}$ is $(1,0,1)$, the net γ_1 may work (using the tokens in these places) and can produce the configuration $(1,0,1)$ (in this case, the yielded configuration is the same with the initial one, but this is not the case in general as we will see in the next example).

Considering the set R_0 of all such pairs, the couple (γ_0, R_0) captures

the "interactive" behavior of γ_0 and γ_1, making abstraction of the internal behavior of γ_1.

To make things clear enough, let us consider the case of the other process, namely π_1. From π_1's point of view, the condition b_7 has been "pumped" by the environment (by π_0). Of course, in order the environment gives b_7 to π_1, it needs b_1. That is, whenever the place s_2 holds one token, the net γ_0 may use it and generate one token in s_1. Making abstraction of the internal structure of γ_0, this exchange of tokens can be specified by the pairs:

- $((0,1,0),(1,0,0))$, corresponding to the case that t_5 has been applied before t_1, and
- $((0,1,1),(1,0,1))$, otherwise.

Similarly, considering the set R_1 of all such pairs, the couple (γ_1, R_1) captures the "interactive" behavior of γ_0 and γ_1, making abstraction of the internal behavior of γ_0.

Therefore, we may say that an environment is an abstract mechanism capable to interact with a module $\mathcal{M}$ by updating, from time to time, the content of the interface places. Such an interaction can be mathematically modeled by a binary relation $R \subseteq \mathbf{N}^{S^c} \times \mathbf{N}^{S^c}$. A pair $(M^c, \overline{M}^c)$ means that the environment reads the content M^c of the interface places and then update it to $\overline{M}^c$. From $\mathcal{M}$'s point of view this updating is done in exactly one step.

Definition 5.1.1 An *environment* for a module $\mathcal{M} = (\gamma, S^c)$ is any binary relation R on the set of markings $\mathbf{N}^{S^c}$.

A couple $\mathcal{J} = (\mathcal{M}, R)$, where $\mathcal{M}$ is a module and R is an environment for $\mathcal{M}$, is called an *environmental module* (*e-module*, for short); $\mathcal{M}$ is called the *underlying module*, and R the *environment*, of $\mathcal{J}$. E-modules will be used mainly to describe in a compact way module behaviors; they will abstract from some parts of the module behaviors by collapsing many consecutive steps into a single one.

Definition 5.1.2 Let $\mathcal{J} = (\mathcal{M}, R)$ be an e-module. The *transition relation* of the e-module $\mathcal{J}$ is the binary relation $[\cdot\rangle_{\mathcal{J}}$ on $\mathbf{N}^S$ given by

$$\begin{array}{rcl} M[x\rangle_{\mathcal{J}} M' & \Leftrightarrow & x \text{ is a transition and } M[x\rangle_{\gamma} M', \text{ or} \\ & & x = (M^c, \overline{M}^c) \in R \text{ and } M|_{S^c} = M^c \text{ and} \\ & & M' = M - M^c + \overline{M}^c, \end{array}$$

for all $M, M' \in \mathbf{N}^S$.

Instead of $M[(M^c, \overline{M}^c)\rangle_{\mathcal{J}} M'$ we will simply write $M\, R\, M'$, whenever $(M^c, \overline{M}^c)$ is not important to be specified.

Composition of e-modules is defined whenever their underlying modules are compatible. If $\mathcal{J}_0 = (\mathcal{M}_0, R_0)$ and $\mathcal{J}_1 = (\mathcal{M}_1, R_1)$ are two e-modules such that $\mathcal{M}_0$ and $\mathcal{M}_1$ are compatible, then

$$\mathcal{J}_0 \circ \mathcal{J}_1 = (\mathcal{M}_0 \circ \mathcal{M}_1, R_0 \cup R_1).$$

We have to note that the environment of an e-module may update the content of the interface places whenever it is possible. That is, whenever a marking M is reachable in the e-module, $M|_{S^c} = M^c$, and $(M^c, \overline{M}^c) \in R$, then the environment may change the marking on S^c to $\overline{M}^c$. Then, the module can execute further.

The approach we have considered for an environment, and for the corresponding transition rule, does not take into account the internal structure neither of the module nor of the environment. This one could appear unrealistic. But, we want to use e-modules for abstraction purposes, and if we should take into consideration the entire internal structure of the module and of the environment then such a purpose can be never reached. However, an intermediate variant of taking into account partial information about their internal structure (or to use semaphore variable like mechanisms) could be an worthy idea.

5.2 (j, λ)-isomorphisms

Could an environment for a module be simulated by the module itself? If the answer to this question is positive, then e-modules would not make much sense. But, one can easily argue in many ways (including a philosophical one too) that such a question cannot have a positive answer in general. From a theoretical point of view, the answer to this question is given as follows: e-modules are *jumping Petri nets* which are strictly more powerful than Petri nets [48, 49, 53].

Let us go a little bit into details. First of all we recall the concept of a *jumping Petri net* in a slightly different way than in [48, 49, 53].

Definition 5.2.1 A *jumping net* (*marked jumping net, labeled marked jumping net*) is a couple $\mathcal{J} = (\gamma, \mathcal{R})$, where γ is a net (marked net, labeled marked

net) and $\mathcal{R}$ is a finite set of binary relations on markings on subsets of S (that is, $\mathcal{R}$ is a set of couples (S', R), where $S' \subseteq S$ and R is a binary relation on $\mathbf{N}^{S'}$).

The pairs $(M, M') \in R$, where $(S', R) \in \mathcal{R}$, will be called (*local*) *jump on* S'.

When $\mathcal{R}$ is singleton, $\mathcal{R} = \{(S', R)\}$, we will simply write $(\gamma, (S', R))$ instead $(\gamma, \{(S', R)\})$, or even (γ, R) provided that S' is clear from context.

From technical reasons we extend jumps to the whole set S of places of γ, as follows:

$$M\, R\, M' \quad \Leftrightarrow \quad M|_{S'}\, R\, M'|_{S'} \text{ and } M|_{S-S'} = M'|_{S-S'},$$

for all $M, M' \in \mathbf{N}^S$ and $(S', R) \in \mathcal{R}$. Moreover, we write $(M, M') \in \mathcal{R}$ or $M\, \mathcal{R}\, M'$ whenever $M\, R\, M'$ for some $(S', R) \in \mathcal{R}$.

A *computation step* in a jumping net $\mathcal{J} = (\gamma, \mathcal{R})$ is performed either by a transition in the usual way, $M[t\rangle_\gamma M'$, or by a jump, $M\, \mathcal{R}\, M'$. A (*finite*) *computation* is a finite sequence of computation steps. For example,

$$M_1[t_1\rangle_\gamma M_2\, \mathcal{R}\, M_3\, \mathcal{R}\, M_4[t_2\rangle_\gamma M_5$$

is a computation in a suitable chosen jumping net.

As we can see, e-modules can be thought as special cases of jumping nets, namely the case of singleton sets $\mathcal{R}$. Indeed, if $\mathcal{R} = \{(S^c, R)\}$, then the difference between $(\gamma, \{(S^c, R)\})$, which denotes a jumping net, and $((\gamma, S^c), R)$, which denotes an e-module, is just a notational one. Therefore, we may identify e-modules by jumping nets whose set of jumps is singleton.

Now, the answer to the question at the beginning of this section is based on the fact that jumping Petri nets are strictly more powerful than Petri nets even if singleton sets $\mathcal{R}$ are considered [48].

In some cases, jumping nets can be simulated by classical Petri nets. We will describe in the sequel such a case. The simulation relation is based on isomorphism of processes and, therefore, we begin by introducing the concept of a jumping net process.

First of all we make the following important assumption:

- whenever we deal with processes of a jumping net $\mathcal{J}$ we implicitly assume that $\mathcal{J}$ does not contain any jump $(\underline{0}, \underline{0})$. This is because such jumps may induce isolated transitions in processes which is contrary to our assumption in Section 1.1.

Now, let $\mathcal{J} = (\gamma, \mathcal{R})$ be a marked jumping net, t a transition and $r = (M, M')$ a jump on a subset $S' \subseteq S$. Then:

- an *elementary occurrence net* associated to t is any labeled occurrence net $\pi = (N, p)$ with the properties: π contains only one event e which is labeled by t, $W(s,t)$ preconditions and $W(t,s)$ postconditions of e labeled by s, for all $s \in S$ (and no other element);

- an *elementary occurrence net* associated to r is any labeled occurrence net $\pi = (N, p)$ with the properties: π contains only one event e which is labeled by r, $M(s)$ preconditions and $M'(s)$ postconditions of e labeled by s, for all $s \in S'$ (and no other element). Pictorially, the event e will be drawn by a double box;

- an *initial occurrence net* of γ is any occurrence net (N, p) which does not contain any event and, for each $s \in S$, it contains exactly $M_0(s)$ conditions labeled by s.

Definition 5.2.2 Let $\mathcal{J} = (\gamma, \mathcal{R})$ be a marked jumping net. The *set of processes* of $\mathcal{J}$, denoted by $\Pi(\mathcal{J})$, is the smallest set with the properties:

1. $\Pi(\mathcal{J})$ contains all the initial occurrence nets associated to $\mathcal{J}$;

2. if $\pi_1 \in \Pi(\mathcal{J})$ and π_2 is an elementary occurrence net associated to a transition t such that ${}^\circ\pi_2 \subseteq \pi_1^\circ$, then the composition of π_1 and π_2 along ${}^\circ\pi_2$, whenever it is defined, is in $\Pi(\mathcal{J})$;

3. if $\pi_1 \in \Pi(\mathcal{J})$ and π_2 is an elementary occurrence net associated to a local jump $r = (M, M')$ on S' such that $|\pi_1^\circ \cap p_1^{-1}(s)| = M(s)$ for all $s \in S'$, then the composition of π_1 and π_2 along ${}^\circ\pi_2$, whenever it is defined, is in $\Pi(\mathcal{J})$.

In cases (2) and (3) we say that π_1 is *extended* (to the right) by π_2.

We want to place stress on the fact that a necessary requirement for the composition of π_1 and π_2 along ${}^\circ\pi_2$, in the definition above, is that π_1 and π_2 have in common only the conditions in ${}^\circ\pi_2$. This requirement is supplied by the words "whenever it is defined".

It is clear that for every $\pi \in \Pi(\mathcal{J})$ there is a sequence

$$\pi_0, \pi_1, \ldots, \pi_m = \pi,$$

where π_0 is an initial occurrence net and π_{i+1} can be constructed from π_i as described in Definition 5.2.2, for all $0 \leq i \leq m-1$. Processes of labeled jumping nets are obtained as for labeled nets, by relabeling the events associated to transitions.

Every net can be viewed as a jumping net by taking the empty set as the set of jumps. Consequently, processes of nets are particular cases of processes of jumping nets. Moreover, $\Pi(\gamma) \subseteq \Pi(\mathcal{J})$, for any jumping net $\mathcal{J} = (\gamma, \mathcal{R})$.

Example 5.2.1 Let $\mathcal{J}_0 = (\gamma_0, R_0)$ and $\mathcal{J}_1 = (\gamma_1, R_1)$, where γ_0 and γ_1 are the nets in Figure 2.3, and R_0 and R_1 are relations on $\mathbf{N}^{S^c}$ containing $r_0 = ((1,0,1),(1,0,1)$ and $r_1 = ((0,1,0),(1,0,0))$, respectively ($S^c = \{s_1, s_2, s_3\}$ and r_0 is the pair given at the beginning of Section 5.1). Then, π_0 (π_1) in Figure 5.1 is a process of $\mathcal{J}_0$ ($\mathcal{J}_1$).

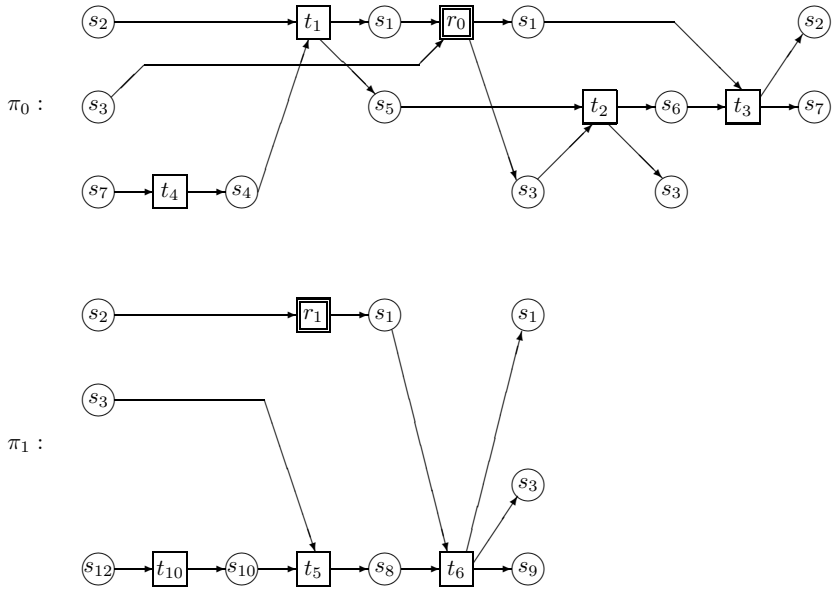


Figure 5.1: Processes of Jumping Petri Nets

Definition 5.2.3 Let $\mathcal{J} = (\gamma, \mathcal{R})$ be a jumping net. Its set of jumps is called Δ-*finite* if there is a finite set V of vectors with integer components such that $M' - M \in V$ for every $(M, M') \in \mathcal{R}$.

Definition 5.2.4 Let $\mathcal{J} = (\gamma, \mathcal{R})$ be a jumping net. Its set of jumps is called *complete* if for every reachable marking M in γ and for every jump

$(M_1, M_2) \in \mathcal{R}$, if (M_1, M_2) is a jump on $S' \subseteq S$ and $M|_{S'} \geq M_1$, then there is a marking $M' \in \mathbf{N}^{S'}$ such that $(M|_{S'}, M') \in \mathcal{R}$ and $M' - M|_{S'} = M_2 - M_1$.

In order to compare processes of jumping nets with processes of nets we need the concept of (j,λ)-*isomorphism*.

Definition 5.2.5 A process $\pi_1 = (N_1, p_1)$ of a labeled jumping net is (j,λ)-*isomorphic* to a process $\pi_2 = (N_2, p_2)$ of a labeled net if there is a bijection $\varphi : B_1 \cup E_1 \to B_2 \cup E_2$ such that:

1. $p_1(x) = p_2(\varphi(x))$ for all $x \in B_1$ and for all $x \in E_1$ with the property that $p_1(x)$ is not a jump; if $p_1(x)$ is a jump then $p_2(\varphi(x))$ is λ;
2. $x \prec_{\pi_1} y$ iff $\varphi(x) \prec_{\pi_2} \varphi(y)$ for all $x, y \in B_1 \cup E_1$.

This notion of (j,λ)-isomorphism of processes is a slight generalization of the classical one: it is an isomorphism of occurrence nets preserving all the condition-labels and all the non-jump event-labels; the events labeled by jumps are mapped into events labeled by λ.

Now we are in a position to prove the main result of this section.

Theorem 5.2.1 ([56]) Let $\mathcal{J} = (\gamma, \mathcal{R})$ be a labeled marked jumping net. If $\mathcal{R}$ is Δ-finite and complete then there is a labeled marked net γ' such that each process of $\mathcal{J}$ is (j,λ)-isomorphic to a process of γ', and vice-versa.

Proof Since $\mathcal{R}$ is Δ-finite, there are finitely many vectors $V_1, \ldots, V_k$ with integer components as in Definition 5.2.3, where $k \geq 1$. Consider

$$U_i = \{M | \exists M' : \ (M, M') \in \mathcal{R} \ \wedge \ M' - M = V_i\},$$

and let $min(U_i)$ the set of minimal elements of U_i, for all i.

It is clear that $min(U_i)$ is finite, and for each $M \in min(U_i)$ there is an unique M' such that $(M, M') \in \mathcal{R}$ and $M' - M = V_i$.

To each i and $M \in min(U_i)$ we associate a transition $t_{i,M}$ such that

$$W'(s, t_{i,M}) = M(s) \ \text{ and } \ W'(t_{i,M}, s) = M'(s),$$

for all places $s \in S'$ (M' is as above and S' is the subset of places which the jump (M, M') is considered for). Moreover, $t_{i,M}$ will be labeled by λ. We also remark that $t_{i,M}$ is not an isolated transition due to our assumption at the beginning of this section.

Let $\gamma' = (\Sigma', M_0, l')$ be the net defined as follows:

- $\Sigma' = (S, T', F', W')$;
- $T' = T \cup \{t_{i,M} | 1 \leq i \leq k,\ M \in min(U_i)\}$;
- $l'(t)$ is $l(t)$ for all $t \in T$, and λ otherwise;
- F' and W' are the extensions of F and W as we sketched above.

$\mathcal{J}$ and γ' have the same initial marking. Assume now that M is a reachable marking in both $\mathcal{J}$ and γ'. Consider the following two cases.

Case 1: $M[t\rangle_\gamma M'$, for some transition $t \in T$. From the definition of γ' we have $M[t\rangle_{\gamma'} M'$ too, and conversely.

Case 2: $M \mathcal{R} M'$ by a local local jump $(M|_{S'}, M'|_{S'})$ on $S' \subseteq S$. Then, there is a jump (M_1, M_2) on S' such that $M_1 \in min(U_i)$ and $M'|_{S'} - M|_{S'} = M_2 - M_1$, for some i. Consequently, the transition t_{i,M_1} may occur at M and will yield M'; that is, $M[t_{i,M_1}\rangle_{\gamma'} M'$.

Conversely, if $M[t_{i,M_1}\rangle_{\gamma'} M'$ for some i and M_1, then there are M_2 and $S' \subseteq S$ such that (M_1, M_2) is a local jump on S'. Moreover, $M'|_{S'} - M|_{S'} = M_2 - M_1$. As the set of jumps is complete, it follows that $(M|_{S'}, M'|_{S'})$ is a local jump on S' and $M \mathcal{R} M'$ by this jump.

Using the facts above the proof can be easily completed. □

5.3 Petri Net Model Validation

One aim of simulation is to *validate* the model w.r.t. some desired behavioral properties (that is, to check whether the desired properties are reflected or not in the simulated runs of the model). As we could expect, many problems are encountered when dealing with simulation of a Petri net model: fairness, alternatives (solving conflicts), termination conditions, visualization and property checking for large processes, etc. All these problems could be grouped into two main classes: *generation* and *analysis* of processes (for a detailed discussion the reader is referred to [15]). Therefore, it turns out to be an important task to look for an adequate tool to represent and generate processes as well as for an efficient strategy for analyzing them. In this section we will show how the mechanism developed in Section 5.2 can be used for Petri net model validation. This approach has first been proposed in [55].

Suppose that a net γ can be decomposed as follows:

$$\begin{array}{lcll} \gamma & = & \gamma_0 \circ \gamma'_0 & (S^{c,0}) \\ \gamma'_0 & = & \gamma_1 \circ \gamma'_1 & (S^{c,1}) \\ \cdots & & & \\ \gamma'_{n-2} & = & \gamma_{n-1} \circ \gamma_n & (S^{c,n-1}) \end{array}$$

where the sets of interface places are $S^{c,0}, S^{c,1}, \ldots, S^{c,n-1}$, respectively.

Formally, we may write

$$\gamma = \gamma_0 \circ (\gamma_1 \circ \cdots \circ (\gamma_{n-1} \circ \gamma_n) \cdots).$$

Suppose that the net γ_n is of reasonable small size such that it supports an ad hoc validation. Then, we can define the jumping net $\mathcal{J}_{n-1}$ in order to generate process samples and validate γ_{n-1} in the context of γ_n. If this step is successfully performed then we may consider valid the net $(\gamma_{n-1} \circ \gamma_n)$ and continue the validation. The main problem we encounter when dealing with the construction of jumping Petri nets (as above) is to find a convenient way to describe the set of jumps. In fact, this problem has two main aspects:

1. decide whether it is possible to define a jumping Petri net $\mathcal{J}_i$ (as above);
2. if the answer to the question above is positive, then find a convenient way to describe its set of jumps.

In practice it could be not necessary to construct *a priori* the jumping Petri nets $\mathcal{J}_i$. We may take γ_i and generate processes of it until a jump is necessary. Then, we take the current marking on the interface places and, together with the current marking on the internal places of γ'_i we generate a jump for γ_i (and save the current marking on the internal places of γ'_i).

Not much is known about Petri net model validation based on simulated runs. The technique sketched above is the only known formal attempt to attack this problem.

Chapter 6

Modular Model Checking of Petri Nets

Model Checking is an automatic technique for verifying finite-state reactive systems, such as sequential circuit designs and communication protocols. Specifications are expressed in a temporal logic, and the reactive system is modeled as a state-transition graph. An efficient search procedure is used to determine automatically if the specifications are satisfied by the state-transition graph. The technique was developed by E.M. Clarke and A. Emerson [11, 12], and an alternative approach based on showing inclusion between ω-automata was later devised by R. Kurshan [32]. Unfortunately, temporal logic model checking procedures suffers from the *state space explosion problem*. This problem arises in systems which are composed of many parallel processes; in general, the size of the state space grows exponentially with the number of processes. An obvious method for trying to avoid the state space explosion problem is to use the natural decomposition of the system into simpler components. Properties of the individual components are verified first, and then properties of the global system are deduced from these. The state space explosion problem is only one motivation for pursuing modular verification. Modular verification is advocated also for other methodological reasons; a *robust verification methodology* should provide rules for deducing properties of systems from properties of their constituent modules.

In this chapter we show that, by means of e-modules, we can transfer properties from the constituent modules to the entire system. The main advantage lies in a possible significant reduction of the state space, in case that e-modules are suitable chosen (see the example at the end of Section

6.4). Since for bounded Petri net modules we can associate in a very natural way a finite state-transition graph, and also for generality, we will present our results first in terms of *fair Kripke structures*, and then we will relate Petri net modules to these structures.

The transition relation of a fair Kripke structure will be divided into two relations, *internal* and *external*, the first one modeling the internal behaviour of the system, while the second one is used to model the interaction with the environment. Consider then a *preorder of simulation* intended to capture two basic aspects:

- a system K_1 may be embedded into a system K_2 having "more behavior";
- the system K_2 may abstract from some parts of the behavior of K_1 by collapsing several consecutive steps into a single one (this is what makes our preorder different from those known from literature – see, for example, [21, 12, 31]).

Then, we show that the *delayed version* of a formula holds in K_1, whenever the original formula holds in K_2 and there is a simulation from K_1 to K_2. Finally, we relate Petri net modules to fair Kripke structures, and discuss briefly *step fairness constraints*.

The results in this chapter are drawn from [57].

6.1 Temporal Logic

We will use the *universal branching-time temporal logic* $\forall CTL^*$ to specify properties of reactive systems. This logic is obtained from CTL^* by eliminating the existential path quantifier. This restriction is necessary in order to be able to transfer properties from systems with "more behavior" to systems smaller in the simulation preorder, which have "less behavior" (see Section 6.2). However, to ensure that existential path quantifiers do not arise via negation, we will assume that formulas are expressed in *negation normal form* (that is, negations are applied only to atomic propositions). As a result, the logics contain both $\vee$ and $\wedge$ as boolean operators. The temporal operators are X ("nexttime"), U ("until"), and V ("releases").

There are two types of formulas in $\forall CTL^*$:

- *state formulas*, whose satisfaction is related to a specific state, and

- *path formulas*, whose satisfaction is related to a specific path.

Let $\mathcal{A}$ be a set of atomic propositions. The syntax of state formulas is given by the following rules:

(i) **true**, **false**, p and $\neg p$ are state formulas, for all $p \in \mathcal{A}$;

(ii) if φ and ψ are state formulas, then $\varphi \vee \psi$ and $\varphi \wedge \psi$ are state formulas;

(iii) if φ is a path formula, then $\forall(\varphi)$ is a state formula.

Two additional rules are needed to specify the syntax of path formulas:

(iv) if φ is a state formula, then φ is a path formula;

(v) if φ and ψ are path formulas, then $\varphi \vee \psi$, $\varphi \wedge \psi$, $X\varphi$, $\varphi U \psi$ and $\varphi V \psi$ are path formulas.

$\forall CTL^*$ (over the set $\mathcal{A}$ of atomic propositions) is the set of state formulas generated by the above rules. Note that since negation in $\forall CTL^*$ can be applied only to atomic propositions, assertions of the form $\neg\forall(\varphi)$, which are equivalent to $\exists(\varphi)$, are not possible. Thus, the logic $\forall CTL^*$ is not closed under negation.

We give the semantics of the logic $\forall CTL^*$ using *fair Kripke structures* (or *structures*, for short) as defined in [12]. Such a structure is a 6-tuple $K = (Q, Q_0, \mathcal{A}, \mathcal{L}, \rho, \mathcal{F})$, where:

- Q is a finite set of *states*;
- $Q_0 \subseteq Q$ is a set of *initial states*;
- $\mathcal{A}$ is a finite set of *atomic propositions*;
- $\mathcal{L} : Q \to \mathcal{P}(\mathcal{A})$ is a function that labels each state with the set of atomic propositions true in that state;
- $\rho \subseteq Q \times Q$ is a *transition relation*;
- $\mathcal{F} \subseteq \mathcal{P}(Q)$ is a set of *fairness constraints* given as Büchi acceptance conditions.

The fairness requirements intend to guarantee that every path (infinite computation in K) contains infinitely many states from each $A \in \mathcal{F}$. Formally, we define the concepts of *path* and *fair path* as follows. First, for an infinite sequence of arbitrary elements $\sigma = x_0 x_1 \cdots$ we define

$$inf(\sigma) = \{x | x = x_i \text{ for infinitely many } i\}.$$

A *path* (*starting* or *beginning at* q_0) in a structure K is an infinite sequence of states

$$\sigma = q_0 q_1 q_2 \cdots$$

satisfying $q_i \, \rho \, q_{i+1}$, for all $i \geq 0$. The notation σ^i will be used to denote the suffix of σ which begins at q_i. The path σ is called *fair* if $inf(\sigma) \cap A \neq \emptyset$, for all $A \in \mathcal{F}$.

We extend the labeling function $\mathcal{L}$ to paths and denote by $\mathcal{L}(\sigma)$ the infinite word

$$\mathcal{L}(q_0)\mathcal{L}(q_1)\mathcal{L}(q_2)\cdots$$

If φ is a state formula, the notation

$$K, q \models \varphi$$

means that φ *holds at state* q in the structure K. Similarly,

$$K, \sigma \models \varphi$$

means that φ *holds along path* σ in the structure K. When K is clear from context, we will usually omit it. The relation $\models$ is defined inductively as follows (q is a state, σ is a path, $p \in \mathcal{A}$, φ_1 and φ_2 are state formulas, and φ and ψ are path formulas):

- $q \models \mathbf{true}$, and $q \not\models \mathbf{false}$.
  $q \models p$ iff $p \in \mathcal{L}(q)$, and $q \models \neg p$ iff $p \notin \mathcal{L}(q)$;

- $q \models \varphi_1 \vee \varphi_2$ iff $q \models \varphi_1$ or $q \models \varphi_2$.
  $q \models \varphi_1 \wedge \varphi_2$ iff $q \models \varphi_1$ and $q \models \varphi_2$;

- $q \models \forall(\varphi)$ iff for all fair paths σ starting at q, $\sigma \models \varphi$;

- $\sigma \models \varphi$ iff $q_0 \models \varphi$, where q_0 is the first state of σ;

– $\sigma \models \varphi \vee \psi$ iff $\sigma \models \varphi$ or $\sigma \models \psi$.
  $\sigma \models \varphi \wedge \psi$ iff $\sigma \models \varphi$ and $\sigma \models \psi$.
  $\sigma \models X\varphi$ iff $\sigma^1 \models \varphi$.
  $\sigma \models \varphi U \psi$ iff $(\exists j \geq 0)(\sigma^j \models \psi \ \wedge \ (\forall 0 \leq i < j)(\sigma^i \models \varphi))$.
  $\sigma \models \varphi V \psi$ iff $(\forall j \geq 0)((\forall 0 \leq i < j)(\sigma^i \not\models \varphi) \ \Rightarrow \ \sigma^j \models \psi)$.

When a state formula φ is true in all initial states of K, we will write $K \models \varphi$.

Consider the operators $\Diamond$ and $\overline{U}$ given by:

- $\Diamond\varphi$ iff $\mathbf{true}\, U \varphi$;
- $\varphi \,\overline{U}\, \psi$ iff $\varphi\, U (\varphi \wedge \psi)$,

and call them *eventually* and *until with equality*. For a formula φ, by $\overline{\varphi}$ we denote the formula obtained from φ by replacing all the occurrences of U by $\overline{U}$.

Let φ a formula. Define recursively the formula $\hat{\varphi}$ as follows:

– if $\varphi = \mathbf{true}$, $\mathbf{false}$, p or $\neg p$, then $\hat{\varphi} = \varphi$;

– if $\varphi = \varphi_1 \vee \varphi_2$, then $\hat{\varphi} = \hat{\varphi}_1 \vee \hat{\varphi}_2$;

– if $\varphi = \varphi_1 \wedge \varphi_2$, then $\hat{\varphi} = \hat{\varphi}_1 \wedge \hat{\varphi}_2$;

– if $\varphi = \forall(\varphi_1)$, then $\hat{\varphi} = \forall(\hat{\varphi}_1)$;

– if $\varphi = X\varphi_1$, then $\hat{\varphi} = \Diamond\hat{\varphi}_1$;

– if $\varphi = \varphi_1 U \varphi_2$, then $\hat{\varphi} = (\Diamond\hat{\varphi}_1)\, U \hat{\varphi}_2$;

– if $\varphi = \varphi_1 V \varphi_2$, then $\hat{\varphi} = \hat{\varphi}_1 V (\Diamond\hat{\varphi}_2)$.

The formula $\hat{\varphi}$ is called the *delayed version* of the formula φ. We can also apply this construction to formulas $\overline{\varphi}$ by replacing the operator U by $\overline{U}$.

6.2 A Simulation Preorder

In the context of modular verification it is helpful to define a preorder relation capturing the idea of "more behaviors" and to use a logic whose semantics relate to the preorder. The preorder should preserve, in some sense, the satisfaction of formulas of the logic, i.e., if a formula is true for a model, a

clear specified variant of it should also be true for every model which is smaller in the preorder. Additionally, composition should preserve the preorder, and a system should be smaller in the preorder than its individual components.

We will make now a basic assumption valid for the rest of the paper (another two will be made in the next section):

- the transition relation ρ of each structure K is the union of two given binary relations on states, $\rho = \rho^i \cup \rho^e$, not necessarily disjoint. The relation ρ^i models the *internal state-changes* in K (that is, proper atomic steps performed by K), and ρ^e models *external state-changes* in K (that is, state-changes caused by the environment). Usually, ρ^e is not completely known, but we can approximate it starting from the remark that in many real cases we know the response of an environment to an output of the module.

Each structure K_j, $j = 0, 1, 2, \ldots$, we will consider is assumed to have the components $K_j = (Q_j, Q_0^j, \mathcal{A}_j, \mathcal{L}_j, \rho_j, \mathcal{F}_j)$, where $\rho_j = \rho_j^i \cup \rho_j^e$.

Definition 6.2.1 Let K_1 and K_2 be two structures, and $\mathcal{A} \subseteq \mathcal{A}_1 \cap \mathcal{A}_2$. Let q and q' be states in K_1 and K_2, respectively. A *simulation from* (K_1, q) *to* (K_2, q') *w.r.t.* $\mathcal{A}$ is a binary relation $H \subseteq Q_1 \times Q_2$ such that:

(1) $(q, q') \in H$;

(2) for all q_0 and q_0', if $(q_0, q_0') \in H$, then:

(2.1) $\mathcal{L}_1(q_0) \cap \mathcal{A} = \mathcal{L}_2(q_0') \cap \mathcal{A}$;

(2.2) for every fair path $\sigma = q_0 q_1 \cdots$ in K_1 there is a fair path

$$\sigma' = q_0' q_1' \cdots$$

in K_2 and a decomposition of σ,

$$\sigma = q_{i_0} \cdots q_{i_1} \cdots q_{i_2} \cdots$$

where $i_0 = 0$, such that for all $j \geq 0$ the following hold:

(a) if $i_{j+1} = i_j + 1$ and $(q_{i_j}, q_{i_{j+1}}) \in \rho_1^e$, then $(q_j', q_{j+1}') \in \rho_2^e$ and $(q_{i_{j+1}}, q_{i_{j+1}}') \in H$;

(b) if $i_{j+1} = i_j + 1$ and $(q_{i_j}, q_{i_{j+1}}) \in \rho_1^i$, then $(q_j', q_{j+1}') \in \rho_2$ and $(q_{i_{j+1}}, q_{i_{j+1}}') \in H$;

(c) if $i_{j+1} > i_j + 1$, then $(q'_j, q'_{j+1}) \in \rho_2^e$ and $(q_{i_{j+1}}, q'_{i_{j+1}}) \in H$.

To indicate that two fair paths σ and σ' correspond as in Definition 6.2.1(2) we will write $H(\sigma, \sigma')$. Clearly, if $H(\sigma, \sigma')$ holds, then $H(\sigma^{i_j}, (\sigma')^j)$ holds for all $j \geq 0$, where i_j are as in Definition 6.2.1. When there is a simulation relation from (K_1, q) to (K_2, q') w.r.t. $\mathcal{A}$, we will write $(K_1, q) \prec_{\mathcal{A}} (K_2, q')$.

Definition 6.2.2 A binary relation H is a *simulation from K_1 to K_2 w.r.t. $\mathcal{A}$* if for each initial state q of K_1 there is an initial state q' of K_2 such that H is a simulation from (K_1, q) to (K_2, q') w.r.t. $\mathcal{A}$.

We will use the notation $K_1 \prec_{\mathcal{A}} K_2$ whenever there is a simulation from K_1 to K_2 w.r.t. $\mathcal{A}$. In the case $\rho_1^e = \rho_2^e = \emptyset$ and $\mathcal{A} = \mathcal{A}_2 \subseteq \mathcal{A}_1$ our definition of simulation is that from [21] (except for the fact that we use fairness constraints given as Büchi but not as Streett acceptance conditions).

Proposition 6.2.1 ([57]) The simulation relation $\prec_{\mathcal{A}}$ is a preorder (i.e., a reflexive and transitive order) on structures whose set of atomic propositions include $\mathcal{A}$.

Proof The relation $H = \{(q, q) | q \in Q\}$ is a simulation from K to K w.r.t. $\mathcal{A}$. Thus, $\prec_{\mathcal{A}}$ is reflexive.

Assume that H_1 is a simulation from K_1 to K_2 w.r.t. $\mathcal{A}$, and H_2 is a simulation from K_2 to K_3 w.r.t. $\mathcal{A}$. Let H_3 be the usual product of the binary relations H_1 and H_2. We show that H_3 is a simulation from K_1 to K_3 w.r.t. $\mathcal{A}$.

First of all we note that $\mathcal{L}_1(q) \cap \mathcal{A} = \mathcal{L}_3(q'') \cap \mathcal{A}$, for all $(q, q'') \in H_3$. Indeed, for each $(q, q'') \in H_3$ there is a state q' in H_2 such that $(q, q') \in H_1$ and $(q', q'') \in H_2$. Since H_1 and H_2 are simulations, it follows that

$$\mathcal{L}_1(q) \cap \mathcal{A} = \mathcal{L}_2(q') \cap \mathcal{A} = \mathcal{L}_3(q'') \cap \mathcal{A},$$

which proves our statement above.

For each initial state q_0 in K_1 there is an initial state q'_0 in K_2 such that H_1 is a simulation from (K_1, q_0) to (K_2, q'_0) w.r.t. $\mathcal{A}$. Similarly, there is an initial state q''_0 in K_3 such that H_2 is a simulation from (K_2, q'_0) to (K_3, q''_0) w.r.t. $\mathcal{A}$. Let

$$\sigma = q_0 q_1 \cdots = q_{i_0} \cdots q_{i_1} \cdots q_{i_2} \cdots$$

and

$$\sigma' = q'_0 q'_1 \cdots$$

be fair paths in K_1 and K_2, respectively, as in Definition 6.2.1 ($i_0, i_1, \ldots$ specify the decomposition of σ). For the fair path σ' there is a fair path

$$\sigma'' = q''_0 q''_1 \cdots$$

in K_3 and a decomposition of σ',

$$\sigma' = q'_0 q'_1 \cdots = q'_{j_0} \cdots q'_{j_1} \cdots q'_{j_2} \cdots$$

as in Definition 6.2.1. We will define recursively a partition of σ

$$\sigma = q_0 q_1 \cdots = q_{k_0} \cdots q_{k_1} \cdots q_{k_2} \cdots$$

such that $H_3(\sigma, \sigma'')$ holds. There are several cases to be considered.

Case 1: $i_1 = 1 = j_1$. Clearly, if $(q_0, q_1) \in \rho_1^e$ then $(q'_0, q'_1) \in \rho_2^e$ and, consequently, $(q''_0, q''_1) \in \rho_3^e$. Moreover, $(q_1, q'_1) \in H_1$ and $(q'_1, q''_1) \in H_2$, which shows that $(q_1, q''_1) \in H_3$. We consider in this case $k_1 = 1$, and the decomposition of σ continues with σ^1, $(\sigma')^1$ and $(\sigma'')^1$ ($H_1(\sigma^1, (\sigma')^1)$ and $H_2((\sigma')^1, (\sigma'')^1)$ hold).

Case 2: $i_1 = 1$ and $j_1 > 1$. Consider $k_1 = i_{j_1}$. It is easy to verify that $(q_{k_1}, q'_{j_1}) \in H_1$ and $(q'_{j_1}, q''_1) \in H_2$; therefore, $(q_{k_1}, q''_1) \in H_3$. The decomposition of σ continues with σ^{k_1}, $(\sigma')^{j_1}$ and $(\sigma'')^1$.

The other two cases, $i_1 > 1$ and $j_1 = 1$, and $i_1 > 1$ and $j_1 > 1$, can be discussed in a similar way. We conclude that $\prec_{\mathcal{A}}$ is transitive and, therefore, $\prec_{\mathcal{A}}$ is a preorder. □

Theorem 6.2.1 ([57]) Let K_1 and K_2 be two structures. Then, for every two states q and q' of K_1 and K_2, respectively, and every two fair paths σ and σ' in K_1 and K_2, respectively, if H is a simulation from (K_1, q) to (K_2, q') w.r.t. a set $\mathcal{A} \subseteq \mathcal{A}_1 \cap \mathcal{A}_2$ and $H(\sigma, \sigma')$ holds true, then for every $\forall CTL^*$ formula φ over $\mathcal{A}$ we have:

(1) if φ is a state formula and $q' \models \overline{\varphi}$ then $q \models \hat{\overline{\varphi}}$;

(2) if φ is a path formula and $\sigma' \models \overline{\varphi}$ then $\sigma \models \hat{\overline{\varphi}}$.

Proof We prove the theorem by induction on the structure of $\overline{\varphi}$.

Base: If $\overline{\varphi}$ is **true** or **false**, the result is trivial. If $\overline{\varphi} = p$ for $p \in \mathcal{A}$, then $q' \models p$ iff $p \in \mathcal{L}_2(q')$. By the definition of simulation, $\mathcal{L}_1(q) \cap \mathcal{A} = \mathcal{L}_2(q') \cap \mathcal{A}$, and so $p \in \mathcal{L}_1(q)$ iff $p \in \mathcal{L}_2(q')$. Thus, $q \models p$. The case $\varphi = \neg p$ for $p \in \mathcal{A}$ is similar to the previous one.

Induction: There are several cases.

1. $\overline{\varphi} = \overline{\varphi}_1 \wedge \overline{\varphi}_2$, a state formula. Then,

$$\begin{array}{rcll} q' \models \overline{\varphi} & \Rightarrow & q' \models \overline{\varphi}_1 \text{ and } q' \models \overline{\varphi}_2 & \\ & \Rightarrow & q \models \hat{\overline{\varphi}}_1 \text{ and } q \models \hat{\overline{\varphi}}_2 & \text{(induction hypothesis)} \\ & \Rightarrow & q \models \hat{\overline{\varphi}} & \end{array}$$

   The same reasoning holds if $\overline{\varphi}$ is a path formula (replacing q' by σ' and q by σ).

2. $\overline{\varphi} = \overline{\varphi}_1 \vee \overline{\varphi}_2$, a state or path formula. This case is similar to the previous case.

3. $\overline{\varphi} = \forall(\overline{\varphi}_1)$, a state formula ($\varphi_1$ is a path formula). Suppose $q' \models \overline{\varphi}$. Let σ_1 be a fair path in K_1 starting at q. By the definition of simulation relation, there is a fair path σ_2 in K_2 starting at q' and such that $H(\sigma_1, \sigma_2)$ holds. Then,

$$\begin{array}{rcll} q' \models \overline{\varphi} & \Rightarrow & \sigma_2 \models \overline{\varphi}_1 & \text{(definition of } \models\text{)} \\ & \Rightarrow & \sigma_1 \models \hat{\overline{\varphi}}_1 & \text{(induction hypothesis)} \end{array}$$

   As σ_1 has been arbitrarily chosen, we obtain $q \models \hat{\overline{\varphi}}$.

4. If φ is a path formula consisting of only a state formula and $\sigma' \models \overline{\varphi}$, then the initial state of σ' satisfies $\overline{\varphi}$. By the induction hypothesis, the initial state of σ will satisfy $\hat{\overline{\varphi}}$. Thus, $\sigma \models \hat{\overline{\varphi}}$.

5. $\overline{\varphi} = X\overline{\varphi}_1$, a path formula. Suppose $\sigma' \models \overline{\varphi}$. Then, $(\sigma')^1 \models \overline{\varphi}_1$. Since $H(\sigma, \sigma')$ holds, there is $i_1 \geq 1$ such that $H(\sigma^{i_1}, (\sigma')^1)$ holds. Therefore, by the induction hypothesis, $\sigma^{i_1} \models \hat{\overline{\varphi}}_1$, and so $\sigma \models \Diamond\hat{\overline{\varphi}}_1 = \hat{\overline{\varphi}}$.

6. $\overline{\varphi} = \overline{\varphi}_1 \overline{U} \overline{\varphi}_2$, a path formula. Suppose $\sigma' \models \overline{\varphi}$. Then, there is $j \geq 0$ such that $(\sigma')^j \models \overline{\varphi}_1 \wedge \overline{\varphi}_2$ and, for all $0 \leq i < j$, $(\sigma')^i \models \overline{\varphi}_1$.

   The definition of simulation leads to the existence of an $i_j \geq j$ such that $H(\sigma^{i_j}, (\sigma')^j)$ holds, and from the induction hypothesis we obtain $\sigma^{i_j} \models \hat{\overline{\varphi}}_1 \wedge \hat{\overline{\varphi}}_2$. Clearly, $\sigma^i \models \Diamond\hat{\overline{\varphi}}_1$, for all $0 \leq i \leq i_j$, and so $\sigma \models \hat{\overline{\varphi}}$.

7. $\overline{\varphi} = \overline{\varphi}_1 V \overline{\varphi}_2$, a path formula. The argument in this case is similar to that for the previous case.

The theorem is proved. □

An immediate consequence of the Theorem 6.2.1 is the following result.

Corollary 6.2.1 ([57]) Let K_1 and K_2 be two structures and $\mathcal{A} \subseteq \mathcal{A}_1 \cap \mathcal{A}_2$. If $K_1 \prec_{\mathcal{A}} K_2$ then, for every $\forall CTL^*$ formula φ over $\mathcal{A}$, $K_2 \models \overline{\varphi}$ implies $K_1 \models \hat{\overline{\varphi}}$.

6.3 Asynchronous Composition of Structures

We consider in this section an asynchronous composition of structures, strongly motivated by its correspondence with composition of e-modules. This operation implies that two structures execute concurrently by performing steps in an interleaved way. That is, at every step, the composed structure may choose to perform a step from one or the other of its components. In order to simplify our discussion and close more structures to e-modules, we will make two basic assumptions:

- the states of each structure K will be considered as *interpretations* over a finite set V of typed variables. That is, each state q is a function assigning to each variable $v \in V$ a value $q(v)$ in its domain. For the case of finite-state systems we have to assume that all variables range over finite domains. We also assume that with each set V, a subset $V^e \subseteq V$ is specified. V^e defines the set of *external* or *interface variables* that are used by the system to communicate with an environment. The set $V^i = V - V^e$ is the set of *internal variables* of K; it is related to the relation ρ^e by:

  $$(\forall q, q')((q, q') \in \rho^e \;\Rightarrow\; q|_{V^i} = q'|_{V^i}).$$

  That is, the environment may update only the external variables, whereas the system may update all the variables.

  From now on we will assume that for a system K_j, $j = 0, 1, 2, \ldots$, its sets of variables are denoted by V_j, V_j^e and V_j^i, without adding them to the tuple defining K.

- the fairness constraints we consider are of the form:

$$\mathcal{F} = \mathcal{F}^i \cup \mathcal{F}^e, \quad \text{where } \mathcal{F}^i \subseteq \mathcal{P}(Dom(\rho^i)) \text{ and } \mathcal{F}^e \subseteq \mathcal{P}(Dom(\rho^e)).$$

  The sets in $\mathcal{F}^i$ are called *internal fairness constraints*, whereas those in $\mathcal{F}^e$ are called *external fairness constraints*. These fairness requirements intend to capture the idea that the environment is given the chance to interfere with the system (by entering infinitely many times in states where the communication with the environment is possible), but also the system may have a proper behavior (by entering infinitely many times in states where internal steps may be done).

Two structures K_1 and K_2 are called *compatible* if $V_1^i \cap V_2^i = \emptyset$ and $V_1^e = V_2^e$. The first condition requires that a variable can only be owned by one of the systems, whereas the second condition requires that the external variables are common for both systems.

Definition 6.3.1 Let K_1 and K_2 be two compatible structures. The *asynchronous composition of* K_1 *and* K_2, denoted by $K_1 \circ K_2$, is the structure $K = (Q, Q_0, \mathcal{A}, \mathcal{L}, \rho, \mathcal{F})$, where:

(1) the set Q of states consists of all the interpretations q of

$$V = V_1^i \cup V^e \cup V_2^i,$$

where $V^e = V_1^e = V_2^e$, such that $q|_{V_1}$ and $q|_{V_2}$ are states in K_1 and K_2, respectively, and $\mathcal{L}_1(q|_{V_1}) \cap \mathcal{A}_2 = \mathcal{L}_2(q|_{V_2}) \cap \mathcal{A}_1$;

(2) $Q_0 = \{q \in Q | q|_{V_1} \in Q_0^1 \ \wedge \ q|_{V_2} \in Q_0^2\}$;

(3) $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$;

(4) $\mathcal{L}(q) = \mathcal{L}_1(q|_{V_1}) \cup \mathcal{L}_2(q|_{V_2})$ for all $q \in Q$ (the definition of Q avoids the existence of atomic propositions p both true and false at q);

(5) $(q, q') \in \rho$ iff

- $(q|_{V_1}, q'|_{V_1}) \in \rho_1$ and $q'|_{V_2^i} = q|_{V_2^i}$, or
- $(q|_{V_2}, q'|_{V_2}) \in \rho_2$ and $q'|_{V_1^i} = q|_{V_1^i}$.

If a step performed in one of the systems is external (internal), then the corresponding step in K is external (internal). A step may be both external and internal;

(6) $\mathcal{F} = \{\{q \in Q | q|_{V_1} \in A_1\} | A_1 \in \mathcal{F}_1\} \cup \{\{q \in Q | q|_{V_2} \in A_2\} | A_2 \in \mathcal{F}_2\}$.

Our definition of asynchronous composition is mainly the same in [27] and [28], excepting that we do not consider strong fairness constraints (our fairness constraints are like weak fairness constraints in the papers cited above). States of the composition are "pairs" of component states that agree on the common variables and on the common atomic propositions. Each transition of the composition involves a transition of one of the two components.

It is straightforward but tedious to prove that asynchronous parallel composition is commutative and associative (up to isomorphism).

For a structure K we denote by $Reach(K)$ the set of all *reachable states* in K; that is,

$$Reach(K) = \{q \in Q | \exists q_0 \in Q_0 : \ (q_0, q) \in \rho^*\}.$$

Given two compatible structures K_1 and K_2, consider

$$K_{1,2} = (Q_1, Q_0^1, \mathcal{A}_1, \mathcal{L}_1, \rho_{1,2}, \mathcal{F}_{1,2})$$

defined as follows:

- $\rho_{1,2}^i = \rho_1^i$, $\rho_{1,2}^e = \rho_1^e \cup \bar{\rho}_2^e \cup \bar{\rho}_2^i$;
- $\bar{\rho}_2^e$ is the set of all pairs $(q_1|_{V_1}, q_2|_{V_1})$, where q_1 and q_2 are states in $K_1 \circ K_2$, $q_1 \in Reach(K_1 \circ K_2)$, $(q_1|_{V_2}, q_2|_{V_2}) \in \rho_2^e$ and $q_1|_{V_1^i} = q_2|_{V_1^i}$;
- $\bar{\rho}_2^i$ is the set of all pairs $(q_1|_{V_1}, q_2|_{V_1})$ such that there is a sequence of states in $K_1 \circ K_2$,
$$q_1 = q_1^1, \ldots, q_1^n = q_2,$$
with the properties: $q_1 \in Reach(K_1 \circ K_2)$, $(q_1^j|_{V_2}, q_1^{j+1}|_{V_2}) \in \rho_2^i$ and $q_1^j|_{V_1^i} = q_1^{j+1}|_{V_1^i}$ for all $1 \leq j < n$;
- $\mathcal{F}_{1,2} = \mathcal{F}_1 \cup \overline{\mathcal{F}}_2$, where

$$\overline{\mathcal{F}}_2 = \{\{q|_{V_1} | q \in Reach(K_1 \circ K_2) \ \wedge \ q|_{V_2} \in A_2\} | A_2 \in \mathcal{F}_2\}.$$

The fairness constraints in $\overline{\mathcal{F}}_2$ are obtained from the (internal and external) fairness constraints in $\mathcal{F}_2$; all of them are external fairness constraints in $K_{1,2}$. Indeed:

- if A_2 is an external fairness constraint in $\mathcal{F}_2$, then every state $q \in Reach(K_1 \circ K_2)$ with the property $q|_{V_2} \in A_2$ verifies also $q|_{V_2} \in Dom(\rho_2^e)$, and so $q|_{V_1} \in Dom(\bar{\rho}_2^e)$;
- if A_2 is an internal fairness constraint in $\mathcal{F}_2$, then every state $q \in Reach(K_1 \circ K_2)$ with the property $q|_{V_2} \in A_2$ verifies also $q|_{V_2} \in Dom(\rho_2^i)$, and so $q|_{V_1} \in Dom(\bar{\rho}_2^i)$.

Therefore, for every fairness constraint $A_2 \in \mathcal{F}_2$,

$$\{q|_{V_1} | q \in Reach(K_1 \circ K_2) \ \wedge \ q|_{V_2} \in A_2\} \subseteq Dom(\bar{\rho}_2^i \cup \bar{\rho}_2^e),$$

which shows that $\overline{\mathcal{F}}_2$ is a set of external fairness constraints in $K_{1,2}$.

Theorem 6.3.1 ([57]) Let K_1 and K_2 be two compatible structures. Then, $K_1 \circ K_2 \prec_{\mathcal{A}_1} K_{1,2}$.

Proof Let $K = K_1 \circ K_2$. Consider $H = \{(q, q|_{V_1}) | q \in Q\}$ and show that H is a simulation from $K = K_1 \circ K_2$ to $K_{1,2}$ w.r.t. $\mathcal{A}_1$.

For every state $q \in Q$ we have:

$$\begin{array}{rcll} \mathcal{L}(q) \cap \mathcal{A}_1 & = & (\mathcal{L}_1(q|_{V_1}) \cup \mathcal{L}_2(q|_{V_2})) \cap \mathcal{A}_1 & \\ & = & (\mathcal{L}_1(q|_{V_1}) \cap \mathcal{A}_1) \cup (\mathcal{L}_2(q|_{V_2}) \cap \mathcal{A}_1) & \\ & = & \mathcal{L}_1(q|_{V_1}) \cup (\mathcal{L}_1(q|_{V_1}) \cap \mathcal{A}_2) & \text{(definition of } Q) \\ & = & \mathcal{L}_1(q|_{V_1}). & \end{array}$$

Then, we note that for every initial state q_0 in K, $q_0|_{V_1}$ is an initial state in $K_{1,2}$.

Let $\sigma = q_0 q_1 q_2 \cdots$ be a fair path in K. Decompose the path σ,

$$\sigma = q_{i_0} \cdots q_{i_1} \cdots q_{i_2} \cdots$$

such that, for all $j \geq 0$, the following requirements are satisfied:

(a) if $i_{j+1} = i_j + 1$, then $(q_{i_j}|_{V_1}, q_{i_{j+1}}|_{V_1}) \in \rho_1$ or $(q_{i_j}|_{V_2}, q_{i_{j+1}}|_{V_2}) \in \rho_2^e$;

(b) if $i_{j+1} \geq i_j + 1$, then $(q_{i_j}|_{V_2}, q_{i_{j+1}}|_{V_2}) \in (\rho_2^i)^+$, $q_{i_j}|_{V_1} = \cdots = q_{i_{j+1}-1}|_{V_1}$, $q_{i_j}|_{V_1^i} = q_{i_{j+1}}|_{V_1^i}$.

Define now a path σ' of $K_{1,2}$ by modifying the path σ as follows:

- keep all (a)-type steps (but restrict all states to V_1);
- replace each (b)-type sequence by $(q_{i_j}|_{V_1}, q_{i_{j+1}}|_{V_1})$.

Clearly, this is an infinite path of $K_{1,2}$. We will prove that this path is fair. Let $A \in \mathcal{F}_{1,2}$.

Case 1: $A \in \mathcal{F}_1$. Then, $A' = \{q \in Q | q|_{V_1} \in A\}$ is a fairness constraint in K. Since σ is a fair path it follows that $inf(\sigma) \cap A' \neq \emptyset$, and so there is $q \in Q \cap inf(\sigma)$. It is enough to show that $q|_{V_1}$ occurs infinitely many times in σ'. In fact, the only problem we encounter is the following one: "condensing" a (b)-type sequence by its left and right most states we may loose some occurrences of q and, therefore, of $q|_{V_1}$. However, at least one occurrence of $q|_{V_1}$ is kept in the left or right most state, and this is enough to ensure that $q|_{V_1}$ occurs infinitely many times in σ'.

Case 2: $A \in \overline{\mathcal{F}}_2$. Then, there is a fairness constraint $A_2 \in \mathcal{F}_2$ such that

$$A = \{q|_{V_1} | q \in Reach(K_1 \circ K_2) \ \wedge \ q|_{V_2} \in A_2\}.$$

But, the set $A' = \{q \in Q | q|_{V_2} \in A_2\}$ is a fairness constraint in K, and so there is $q \in A'$ occurring infinitely many times in σ. Moreover, q is reachable in K and $q|_{V_1} \in Dom(\bar{\rho}_2^i \cup \bar{\rho}_2^e)$ (as we have shown above the theorem). By a similar argument as in Case 1 we can prove that $q|_{V_1} \in inf(\sigma')$. Hence, $inf(\sigma') \cap A \neq \emptyset$.

Therefore, the path σ' is fair and it is straightforward to prove that $H(\sigma, \sigma')$ holds. Thus, H is a simulation from $K_1 \circ K_2$ to $K_{1,2}$ w.r.t. $\mathcal{A}_1$. □

Corollary 6.3.1 ([57]) Let K_1 and K_2 be two compatible structures. Then, for every $\forall CTL^*$ formula φ over $\mathcal{A}_1$, $K_{1,2} \models \overline{\varphi}$ implies $K_1 \circ K_2 \models \hat{\overline{\varphi}}$, where $K_{1,2}$ is the structure defined as above.

Proof From Theorem 6.3.1 and Corollary 6.2.1. □

What we have already done in this section acts as an abstraction methodology. Given a system $K_1 \circ K_2$, we abstract from the internal variables of K_2, obtaining $K_{1,2}$. The structure $K_{1,2}$ collapses consecutive steps in $K_1 \circ K_2$ by a single one, ensuring a simulation from $K_1 \circ K_2$ to $K_{1,2}$. The number of

states in $K_{1,2}$ could be reduced in comparison with $K_1 \circ K_2$ (the number of arcs could be increased but this is not as important as the reduction in the number of states is). If the main meaning of a formula is not diminished by delaying it, then we may try to check its satisfaction in $K_{1,2}$.

It is generally recognized that abstractions are not efficient if all the variables in a system are *visible* (if we cannot abstract from the internal variables of K_2, in our case – see [12] and [28] for more comments). On the other side, to have a good abstraction it is important to produce exactly $\rho^e_{1,2}$, or to produce approximations sufficiently closed to $\rho^e_{1,2}$ so that we can still verify interesting properties of the system.

6.4 Modular Model Checking of Petri Nets

To each bounded net γ we can associate, in a natural way, a *Kripke structure without fairness constraints* $K(\gamma) = (Q, Q_0, \mathcal{A}, \mathcal{L}, \rho)$, as follows:

- regard places as variables which range over finite sets of positive integers. Then, the set of states is the set of all interpretations of variables (markings of γ component wise bounded by some integer n). The only initial state is the initial marking;

- we may define a set $\mathcal{A}$ of atomic propositions using the variables in S and the constants, functions and predicates over the corresponding domains (as in [34], p. 182). These propositions should be either true or false at a marking (state) M, and they will be used to define state and path formulas. Let $\mathcal{L}$ be the function which associate to each marking M the set of all atomic propositions in $\mathcal{A}$ satisfied at M;

- the transition relation is specified by the set of transitions of γ in an obvious way; that is, $(M, M') \in \rho$ iff there is a transition t such that $M[t\rangle_\gamma M'$. The relation ρ is considered internal ($\rho = \rho^i$).

We may also add to $K(\gamma)$ a set $\mathcal{F}$ of fairness constraints getting in such a way a fair Kripke structure $K(\gamma, \mathcal{F})$ associated to γ.

We suppose from now on that for every net (module, e-module) there is given a set of atomic propositions (referring to its set of markings). Moreover, we will assume that whenever we merge (combine) two markings M_1 and M_2 which agree on some places (in order to obtain a marking of the composed

net, module or e-module), the propositions that are satisfied at the new marking are exactly those that are satisfied at M_1 and M_2 [1].

We extend the notations above to modules and e-modules by:

- for a bounded module $\mathcal{M} = (\gamma, S^c)$, $K(\mathcal{M})$ is obtained from $\mathcal{K}(\gamma)$ by considering S^c as the set of external (interface) variables;
- for a bounded e-module $\mathcal{J} = (\mathcal{M}, R)$, $K(\mathcal{J})$ is obtained from $K(\mathcal{M})$ by adding the external transition relation

  $$\rho^e = \{(M, M') \in \mathbf{N}^S \times \mathbf{N}^S | M'|_{S^i} = M|_{S^i} \ \wedge \ (M|_{S^c}, M'|_{S^c}) \in R\}$$

  to the transition relation of $\mathcal{M}$;
- for a bounded module $\mathcal{M}$ (e-module $\mathcal{J}$) and a set $\mathcal{F}$ of fairness constraints, $K(\mathcal{M}, \mathcal{F})$ ($K(\mathcal{J}, \mathcal{F})$) is the structure obtained by adding $\mathcal{F}$ to the 5-tuple $K(\mathcal{M})$ ($K(\mathcal{J})$). For e-modules, the fairness constraints we use are like in Section 6.3.

The pairs $(\gamma, \mathcal{F})$ ($(\mathcal{M}, \mathcal{F})$, $(\mathcal{J}, \mathcal{F})$) as above are called *fair nets* (*modules*, *e-modules*). The simulation and satisfaction relations are defined for them by means of the structures they induce. For example, if $(\mathcal{J}_1, \mathcal{F}_1)$ and $(\mathcal{J}_2, \mathcal{F}_2)$ are two fair bounded e-modules whose underlying nets are elements of $PN(S^c, M_0^c)$, $\mathcal{A}$ is a set of common atomic propositions, and φ is a $\forall CTL^*$ formula over the set of atomic proposition of $\mathcal{J}_1$, then we write

- $(\mathcal{J}_1, \mathcal{F}_1) \prec_{\mathcal{A}} (\mathcal{J}_2, \mathcal{F}_2)$ for $K(\mathcal{J}_1, \mathcal{F}_1) \prec_{\mathcal{A}} K(\mathcal{J}_2, \mathcal{F}_2)$, and
- $(\mathcal{J}_1, \mathcal{F}_1) \models \varphi$ for $K(\mathcal{J}_1, \mathcal{F}_1) \models \varphi$.

Let $(\mathcal{J}_1, \mathcal{F}_1)$ and $(\mathcal{J}_2, \mathcal{F}_2)$ be two compatible fair e-modules whose underlying nets are elements of $PN(S^c, M_0^c)$. Define their composition, denoted $(\mathcal{J}_1, \mathcal{F}_1) \circ (\mathcal{J}_2, \mathcal{F}_2)$, by $(\mathcal{J}_1 \circ \mathcal{J}_2, \mathcal{F})$, where $\mathcal{F}$ is defined as in Definition 6.3.1. Further, consider the fair e-module $(\mathcal{J}_{1,2}, \mathcal{F}_{1,2})$, where:

- $\mathcal{J}_{1,2} = (\gamma_1, R_{1,2})$;
- $R_{1,2} = R_1 \cup R_2' \cup R_2''$;

[1] It was pointed out in [18] that in the case of 1-bounded nets we may restrict the set of atomic propositions to propositions p_s, where s is a place, with the following meaning: a marking M satisfies p_s iff it marks the place s. Clearly, for such nets, our supposition trivially holds. Anyway, it is not a severe restriction for the case of bounded nets.

- R_2' is the set of all pairs $(M|_{S^c}, M'|_{S^c}) \in R_2$, where M is reachable in $\mathcal{J}_1 \circ \mathcal{J}_2$;
- R_2'' is the set of all pairs $(M|_{S^c}, M'|_{S^c})$, where M is reachable in $\mathcal{J}_1 \circ \mathcal{J}_2$ and $M'|_{S_2}$ is reachable from $M|_{S_2}$ (in γ_2) by at least one transition occurrence;
- $\mathcal{F}_{1,2} = \mathcal{F}_1 \cup \overline{\mathcal{F}}_2$, where $\overline{\mathcal{F}}_2 = \{\{M|_{S_1} | M \text{ is reachable in } \mathcal{J}_1 \circ \mathcal{J}_2 \wedge M|_{S_2} \in A_2\} | A_2 \in \mathcal{F}_2\}$.

The following theorem makes the connection between modules and structures.

Theorem 6.4.1 ([57]) Let $(\mathcal{J}_1, \mathcal{F}_1)$ and $(\mathcal{J}_2, \mathcal{F}_2)$ be two compatible fair e-modules whose underlying nets are elements of $PN(S^c, M_0^c)$. If $\mathcal{J}_1 \circ \mathcal{J}_2$ is bounded, then:

(1) $K((\mathcal{J}_1, \mathcal{F}_1) \circ (\mathcal{J}_2, \mathcal{F}_2)) = K(\mathcal{J}_1, \mathcal{F}_1) \circ K(\mathcal{J}_2, \mathcal{F}_2)$;

(2) $(\mathcal{J}_1, \mathcal{F}_1) \circ (\mathcal{J}_2, \mathcal{F}_2) \prec_{\mathcal{A}_1} (\mathcal{J}_{1,2}, \mathcal{F}_{1,2})$;

(3) for every $\forall CTL^*$ formula φ over the set of atomic proposition of $\mathcal{J}_1$, $(\mathcal{J}_{1,2}, \mathcal{F}_{1,2}) \models \overline{\varphi}$ implies $(\mathcal{J}_1, \mathcal{F}_1) \circ (\mathcal{J}_2, \mathcal{F}_2) \models \hat{\overline{\varphi}}$.

Proof If $\mathcal{J}_1 \circ \mathcal{J}_2$ is bounded, then $\mathcal{J}_1$ and $\mathcal{J}_2$ are bounded. Then, (1) follows immediately from definitions (see also the assumption on composing markings at the beginning of the section), and (3) from (2) and Corollary 6.3.1.

(2) Let $K_1 = K(\mathcal{J}_1, \mathcal{F}_1)$ and $K_2 = K(\mathcal{J}_2, \mathcal{F}_2)$. We have:

$$K((\mathcal{J}_1, \mathcal{F}_1) \circ (\mathcal{J}_2, \mathcal{F}_2)) = K(\mathcal{J}_1, \mathcal{F}_1) \circ K(\mathcal{J}_2, \mathcal{F}_2) = K_1 \circ K_2 \prec_{\mathcal{A}_1} K_{1,2}.$$

By the remark that $K_{1,2} = K(\mathcal{J}_{1,2}, \mathcal{F}_{1,2})$ we get (2). □

When the set $\mathcal{F}$ of fairness constraints of a fair e-module $(\mathcal{J}, \mathcal{F})$ contains only the set of all reachable markings, then all paths of the e-module are fair. In such a case we may simplify the pair $(\mathcal{J}, \mathcal{F})$ to $\mathcal{J}$ (but understanding that all the paths of $\mathcal{J}$ are fair). Composition of such e-modules leads to such an e-module (all paths are fair). Then, directly from the theorem above we obtain:

Corollary 6.4.1 ([57]) Let $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$ be two compatible nets. If $\gamma_1 \circ \gamma_2$ is bounded, then for every $\forall CTL^*$ formula φ over the set of atomic proposition of γ_1, $\mathcal{J} \models \overline{\varphi}$ implies $\gamma_1 \circ \gamma_2 \models \hat{\overline{\varphi}}$, where $\mathcal{J} = (\gamma_1, R)$ and R is the set of jumps induced by γ_2 in $\gamma_1 \circ \gamma_2$.

Proof Considering $\mathcal{J}_1 = (\gamma_1, \emptyset)$ and $\mathcal{J}_2 = (\gamma_2, \emptyset)$, the e-module $\mathcal{J}_{1,2}$ is just the e-module $\mathcal{J}$ in theorem. Moreover, $\mathcal{F}_{1,2}$ contains the set of all reachable marking in γ_1 and also a subset, possible strict, of this one. However,

$$\gamma_1 \circ \gamma_2 \prec_{\mathcal{A}_1} (\mathcal{J}_{1,2}, \mathcal{F}_{1,2}) = (\mathcal{J}, \mathcal{F}_{1,2}) \prec_{\mathcal{A}_1} \mathcal{J}.$$

Then, $\mathcal{J} \models \overline{\varphi}$ implies $\gamma_1 \circ \gamma_2 \models \hat{\overline{\varphi}}$. □

This corollary tells us how properties of components are transferred to the entire system. As we have mentioned in the previous section, the main goal is to find an approximation of the set of jumps, sufficiently closed to the real set of jumps on the interface places. A very convenient case is when a net $\gamma = \gamma_0 \circ \gamma_1$ is context-free w.r.t. γ_0 or γ_1. This is the case of the net in Figure 2.2 which is context-free w.r.t. both γ_0 and γ_1. Then, $\mathcal{J}_0 = (\gamma_0, R_0)$, where

$$\begin{aligned} R_0 \;=\; \{&((0,1,1),(0,1,1)),((0,1,1),(0,1,0)),\\ &((0,1,0),(0,1,0)),((0,1,0),(0,1,1)),\\ &((1,0,1),(1,0,1)),((1,0,1),(1,0,0))\}, \end{aligned}$$

is an e-module which assures a simulation from γ to it. The state space of γ is reduced to the state space of γ_0 (we have to add some more arcs, corresponding to R_0, but this is not as important as the reduction of the state space is). Therefore, properties of $\mathcal{J}_0$ can be transferred then to γ.

6.5 Step Fairness Constraints

The fairness constraints we considered in the last sections assure that the environment is given the chance to interfere with the system. However, this does not assure that the environment will do it; an environment step $(q, q') \in \rho^e$ may be "simulated" by the system (that is, $(q, q') \in (\rho^i)^+$). If the system enters infinitely many times in q then the system itself may simulate each time the environment step (q, q'). In such a case, there is no "proper" cooperation with the environment w.r.t. q. From this point of view we may ask for

strengthening the fairness constraints. A way to do that is to consider sets of steps as constraints. That is,

$$\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2, \quad \text{where } \mathcal{F}_1 \subseteq \mathcal{P}(\rho^i) \text{ and } \mathcal{F}_2 \subseteq \mathcal{P}(\rho^e).$$

The fairness requirements are defined now by

$$(\forall A \in \mathcal{F})(inf s(\sigma) \cap A \neq \emptyset),$$

where, for a path σ,

$$inf s(\sigma) = \{(q, q') | q = q_i \wedge \ q' = q_{i+1} \ \text{ for infinitely many } i\}.$$

Excepting for the case of e-modules where each step is both internal and external, the step fairness constraints are expressive enough: they can select only the paths containing both infinitely many internal steps and infinitely many external steps.

All the results developed in Section 6.2 hold for such fairness requirements as well.

The composition of two structures (under these fairness constraints) may be defined as in Section 6.3 but with the difference:

$$\begin{aligned} \mathcal{F} \ = \ & \{\{(q, q') \in Q \times Q | q|_{V_2^i} = q'|_{V_2^i} \ \wedge \ (q|_{V_1}, q'|_{V_1}) \in A_1\} | A_1 \in \mathcal{F}_1\} \cup \\ & \{\{(q, q') \in Q \times Q | q|_{V_1^i} = q'|_{V_1^i} \ \wedge \ (q|_{V_2}, q'|_{V_2}) \in A_2\} | A_2 \in \mathcal{F}_2\}. \end{aligned}$$

The construction $K_{1,2}$ in the same section may be transferred to this case but with the following definition for $\overline{\mathcal{F}_2}$:

- for each external fairness constraint $A_2 \in \mathcal{F}_2$, consider the set A_2' of all pairs $(q_1|_{V_1}, q_2|_{V_1})$ such that q_1 is reachable in $K_1 \circ K_2$, $(q_1|_{V_2}, q_2|_{V_2}) \in A_2$ and $q_1|_{V_1^i} = q_2|_{V_1^i}$.

  Let $\overline{\mathcal{F}}_2^e = \{A_2' | A_2 \in \overline{\mathcal{F}}_2 \text{ is an external fairness constraint}\}$;

- for each internal fairness constraint$A_2 \in \mathcal{F}_2$ consider the set A_2' of all pairs $(q_1|_{V_1}, q_2|_{V_1})$ such that there is a sequence of states as in the definition of $\bar{\rho}_2^i$ (see Section 6.3) and $(q_1^j|_{V_2}, q_1^{j+1}|_{V_2}) \in A_2$ and $q_1^j|_{V_1^i} = q_1^{j+1}|_{V_1^i}$, for some $1 \leq j < n$.

  Let $\overline{\mathcal{F}}_2^i = \{A_2' | A_2 \in \overline{\mathcal{F}}_2 \text{ is an internal fairness constraint}\}$;

- $\overline{\mathcal{F}}_2 = \overline{\mathcal{F}}_2^i \cup \overline{\mathcal{F}}_2^e$.

With this definition of $K_{1,2}$, Theorem 6.3.1 holds in this case as well. Indeed, the proof keeps the same line up to the fairness requirements of σ'. Then, let $A'_2 \in \mathcal{F}_{1,2}$ be a fairness constraint in $K_{1,2}$. There are three cases to be considered: $A'_2 \in \mathcal{F}_1$, $A'_2 \in \overline{\mathcal{F}}^e_2$, and $A_2 \in \overline{\mathcal{F}}^i_2$. The first two can be treated in the same way as the case 1 in the proof of Theorem 6.3.1. For the last case, let $A_2 \in \mathcal{F}_2$ be the set which A'_2 is obtained from. Since σ is fair, there is a pair (q, q') of states in $K_1 \circ K_2$ such that $q|_{V_1^i} = q'|_{V_1^i}$, $(q|_{V_2}, q'|_{V_2}) \in A_2$, and the pair (q, q') occurs infinitely many times in σ. However, each occurrence of (q, q') is inside of a (b)-type sequence, and each such (b)-type sequence is collapsed in σ' by the pair of its left and right most states; moreover, these pairs are in A'_2. As the set of states is finite, at least a pair as that described above occurs infinitely many times in σ'. That is, $inf s(\sigma') \cap A'_2 \neq \emptyset$, and so the path σ' is fair.

The results in Section 6.4 may now be easily reformulated for the case of step fairness constraints.

Chapter 7

Modular Analysis of Workflow Nets

In this chapter, a model checking technique for verifying bounded workflow nets against $\forall CTL^*$ formulas is discussed [59]. The technique is based on abstraction, and it works as follows. The original bounded workflow net is split into two sub-workflow nets. One of them is augmented by the relation induced by the other one. The result is a *workflow module*, which is a structure similar to e-modules. When this workflow module is bounded, a simulation from the original workflow net to it can be established. This simulation assures that a substantial fragment of $\forall CTL^*$ formulas is preserved by delaying from the workflow module to the original workflow net. In this way, the state space of the original workflow net can be reduced drastically. Then, several splitting criteria which lead to bounded workflow modules (the boundedness requirement is necessary in order to apply model checking procedures), are presented.

7.1 Introduction to Workflow Modeling and Verification

A *workflow* is a collection of cooperating, coordinated activities designed to carry out a well-defined complex process, such as trip planning, graduate student registration procedure, or a business process in a large enterprise. An activity in a workflow might be performed by a human, a device, or a program.

Workflow management systems provide a framework for capturing the interaction among the activities in a workflow and are recognized as a new paradigm for integrating disparate systems, including legacy systems. Automated workflow management is becoming increasingly important, as it is one of the enabling technologies for business-to-business e-commerce. Ideally, a workflow management system should be able to help the user in analyzing and reasoning about complex business processes. Unfortunately, present-day systems do not provide sufficient support for this activity and the virtual lack of analysis and reasoning facilities in current workflow management systems is considered a serious problem. To tackle this problem, a formal specification model with a well defined semantics is needed.

Two approaches for workflow modeling, one based on Petri nets (*workflow nets*), and one based on directed acyclic graphs (*workflow graphs*) seem most promising. The interest in Petri nets for workflow modeling, [2, 16, 17, 37, 69], is amply justified by their properties. Petri nets have a well-defined semantics, provide a graphical language, and are expressive. Moreover, many properties and analysis techniques for Petri nets are now available. *Workflow graphs* introduced in [44, 45] provide a more direct way of modeling workflows. They are based on directed acyclic graphs, with two types of nodes, *tasks* and *conditions*. The graph has one node with no incoming flows, called the *initial node*, and one node with no outgoing flows, called the *final node*. Several control structures for workflow graphs are defined: *sequence*, *and-split*, *and-join*, *or-split*, and *or-join*. Workflow graphs can be used to identify structural conflicts in process models, such as *deadlock* and *lack of synchronization* [1].

One of the most important correctness properties of a workflow model is *proper termination*. To satisfy the proper termination property a workflow net must be *sound*. The soundness property of a workflow net requires that (i) the marking with a single token on the final place should be reachable from any marking, which is in turn reachable from the marking with a single token on the initial place, (ii) the marking containing a token on the final place should be unique, and (iii) the net with a single token on the initial place should have no dead transitions. The soundness of a workflow net is equivalent to liveness and boundedness (which are decidable properties) and can be verified using standard Petri net algorithms and techniques [1].

[1]The algorithm given by Sadiq and Orlowska in [44, 45] to verify structural conflicts proved to be incomplete. See [33] and [3] for a complete solution.

The soundness of workflow nets has been extended to support modeling various situations encountered in practice. For example, *structural soundness* allows modeling of systems with shared resources [5]. Structural soundness permits $k > 1$ tokens on the initial and final place. *Generalized soundness*, introduced in [22] requires that the marking with k tokens on the final place should be reachable from any marking, which is in turn reachable from the marking with k tokens on the initial place. The generalized soundness is decidable [23].

Soundness is a very important correctness criterion for a workflow net, but it is not the only one. In [22], two new properties have been considered, *separability* and *serialisability*. The former is a behavioral property stating that the behavior of a workflow net with k tokens in the initial node can be seen as a combination of the behavior of k copies of the net, each of them with one token in the initial node. Serialisability requires that the set of traces of a workflow net with id-markings (in this case each token has an identifier) is equal to the set of traces of an abstraction of the workflow net. In general, we wish to have a model that meets some specifications given for example by a temporal logic formula. This becomes even more important when workflow models are data dependent. Recently, data has been recognized as a fundamental aspect of workflow specification [46], but there are few results in the validation of process data.

There have been a few proposals for verifying workflow specification by model checking or by using compositional reasoning. For example, [26] reports practical experience in using the TRACTA approach for verifying workflow schemes. Originally, the TRACTA approach has been developed for modeling and analysis of concurrent and distributed systems. It is based on the use of labeled transition systems for modeling the behavior of system components and for expressing system properties. The properties are verified by reachability analysis combined with compositionality in order to avoid the state space explosion. The approach in [68] introduces certain operators for composing simple workflow nets in order to get complex workflow nets and to verify their properties. In fact, the paper focuses mainly on the termination property (the guaranteed option to terminate successfully).

7.2 Workflow Nets

According to the Workflow Management Coalition [70], a *workflow management system* is a system that completely defines, manages, and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic.

Workflows are *cased-based*, and the goal of workflow management is to handle cases as efficiently as possible. Cases are handled by executing *tasks* in a specific order. In order to do that, and to model states between tasks as well, *conditions* are added. A condition may hold or not. Each task has *pre-conditions* and *post-conditions*. The pre-conditions should hold before the task is executed, and the post-conditions should hold after the task is executed.

Workflow systems are modeled by Petri nets in a rather straightforward way: tasks are modeled by transitions, conditions are modeled by places, and cases are modeled by tokens. A Petri net which models a workflow system should satisfy two basic requirements. First, it should have two distinguished places, i and o. A token in i corresponds to a case which needs to be handled, and a token in o corresponds to a case already handled. Therefore, i should be an *input place*, and o an *output place*. Secondly, every task and condition should contribute to the case processing. As a conclusion, any element of the Petri net modeling the workflow system should be on a path from the place i to the place o.

Definition 7.2.1 ([1]) A *worflow net*, abbreviated WF net, is a Petri net Σ with the following two properties:

1. Σ has two special places, i and o. The place i is called the *input place* of Σ and it satisfies ${}^\bullet i = \emptyset$, and the place o is called the *output place* of Σ and it satisfies $o^\bullet = \emptyset$;

2. Any node $x \in S \cup T$ in the graph of Σ is on a path from i to o.

As we can easily see, in any WF net the following properties hold true:

- $\{i\}$ and $S - \{o\}$ are siphons, and $S - \{o\}$ is the maximal siphon;
- $\{o\}$ and $S - \{i\}$ are traps, and $S - \{i\}$ is the maximal trap.

A few notational conventions regarding markings of WF nets are in order. Let Σ be a WF net.

1. For any place s of Σ and any natural number $k \geq 1$, we denote by M_{ks} the marking given by $M_{ks}(s) = k$ and $M_{ks}(s') = 0$ for all $s' \neq s$. When $k = 1$ the notation is simplified to M_s.

2. Markings of Σ will be sometimes written in the form $M = (p, M', q)$, where $M(i) = p$, $M(o) = q$, and $M(s) = M'(s)$, for all $s \in S - \{i, o\}$.

A workflow net should satisfy two natural correctness requirements:

- for any case, the procedure will eventually terminate and, when it terminates the place o is marked by exactly one token and the other places are unmarked;

- all tasks are quasi-live.

These two requirements lead to the following definition.

Definition 7.2.2 ([1]) A WF net Σ is called *sound* if it satisfies the following two properties:

1. $(\forall M \in [M_i\rangle)(M_o \in [M\rangle)$ (*proper termination*);

2. $(\forall t \in T)(\exists M \in [M_i\rangle)(M[t\rangle)$ (*quasi-liveness*).

This is the soundness concept originally introduced by van der Aalst in [1], except for the fact that his definition requires one more property to be satisfied:

$$(\forall M \in [M_i\rangle)(M(o) \geq 1 \quad \Rightarrow \quad M = M_o).$$

However, as it has been remarked in [23], this property can easily be obtained from Definition 7.2.2(1). We will prove this result below in a more general framework.

There are situations where an extension of the above definition to allow simultaneous cases, is necessary [5, 22].

Definition 7.2.3 ([5]) A WF net Σ is called *k-sound*, where $k \geq 1$, if it satisfies the following two properties:

1. $(\forall M \in [M_{ki}\rangle)(M_{ko} \in [M\rangle)$ (*proper termination*);
2. $(\forall t \in T)(\exists M \in [M_{ki}\rangle)(M[t\rangle)$ (*quasi-liveness*).

Clearly, 1-sound WF nets are exactly the sound WF nets. The original definition of k-soundness in [5] requires one more property to be satisfied, namely

$$(\forall M \in [M_{ki}\rangle)(M(o) \geq k \quad \Rightarrow \quad M = M_{ko}).$$

However, one can easily show that this property follows from Definition 7.2.3(1), as the next proposition proves it.

Proposition 7.2.1 ([23]) If Σ is a k-sound WF net then the following property holds true:

$$(\forall M \in [M_{ki}\rangle)(M(o) \geq k \quad \Rightarrow \quad M = M_{ko})$$

Proof If there were $s \in S - \{o\}$ such that $M(o) \geq k$ and $M(s) \neq 0$, then the marking M_{ko} would never be reached from M because each transition in Σ has at least one outgoing arc and $o^\bullet = \emptyset$. □

In some sense it is surprising that this simple property has not been remarked by anyone for so many years. Nevertheless, it is a very useful property.

A simple but interesting relationship between the first property in Definition 7.2.2 and the home marking property [19, 36] can be drawn.

Definition 7.2.4 ([19, 36]) A marking M is called a *home marking of a Petri net* Σ *with respect to a marking* M_0 if $M \in [M'\rangle$ for all $M' \in [M_0\rangle$.

It is clear that the proper termination property in Definition 7.2.3 can be replaced by:

"M_{ko} is a home marking of Σ with respect to M_{ki}."

But, we can prove a little bit more.

Proposition 7.2.2 ([61]) Let Σ be a k-sound WF net. Then, M_{ko} is the unique home marking of Σ with respect to M_{ki}.

Proof If we assume that M is a home marking of Σ with respect to M_{ki}, then $M(o) \geq k$ because M should be reachable from M_{ko} and $o^\bullet = \emptyset$. Therefore, $M \geq M_{ko}$, which leads to $M = M_{ko}$ by Proposition 7.2.1. □

Reachable markings of WF nets enjoy special properties. Some of them are given in the following proposition.

Proposition 7.2.3 ([60]) Let Σ be a WF net.

1. If Σ is k-sound, where $k \geq 1$, then any two distinct reachable markings of Σ are incomparable and, consequently, Σ is bounded with respect to M_{ki}.
2. Let $k \geq 1$. Then, for all $p \leq k$ and q, $(k+1-p, M, q) \in [M_{(k+1)i}\rangle_\Sigma$ iff $(k-p, M, q) \in [M_{ki}\rangle_\Sigma$.
3. If Σ is k'-sound for all $1 \leq k' \leq k$, then any reachable marking from M_{ki} is of the form $(k-p, M, q)$, where $0 \leq q \leq p \leq k$. Moreover, $q = p$ iff $M = \underline{0}$.

Proof 1. If Σ had two distinct markings M_1 and M_2 such that $M_1 < M_2$, then any transition sequence leading M_1 to M_{ko} would lead M_2 to a marking strictly greater than M_{ko}, contradicting Proposition 7.2.1. As any set of pairwise incomparable markings is finite, Σ is bounded with respect to M_{ki}.

2. Directly from the firing rule applied to WF nets.

3. From (2) we have that $(k-p, M, q)$ is reachable from M_{ki} iff $(0, M, q)$ is reachable from M_{pi}, for all $1 \leq p \leq k$. As Σ is p-sound, $q \leq p$. Moreover, $q = p$ iff $M = \underline{0}$. □

Proposition 7.2.3(3) does not generally hold if we require only the k-soundness of Σ. For example, the Petri net in Figure 7.1 is a 5-sound WF net but the marking $(2, 2, 0, 4)$ is reachable in Σ from M_{5i} and $4 > 5 - 3 = 2$ (Σ is not k'-sound, for any $k' = 1, 2, 3, 4$).

If Σ is k'-sound for all $1 \leq k' \leq k+1$, then by Proposition 7.2.3 we obtain that the set $[M_{(k+1)i}\rangle_\Sigma$ can be written as $[M_{(k+1)i}\rangle_\Sigma = \mathcal{M}^1_{k+1} \cup \mathcal{M}^2_{k+1}$, where:

- $\mathcal{M}^1_{k+1} = \{(k+1-p, M, q) | (k-p, M, q) \in [M_{ki}\rangle_\Sigma\}$, and
- $\mathcal{M}^2_{k+1} = \bigcup_{t_i \in i^\bullet,\ (0,M,q) \in [M_{ki}\rangle_\Sigma} [(0, M + W(t_i, -), q)\rangle_\Sigma$, where $W(t_i, -)$ is the vector $W(t_i, -)(s) = W(t_i, s)$ for all s.

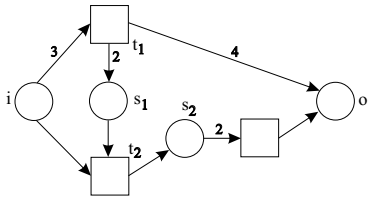


Figure 7.1: A 5-sound WF net which violates Proposition 7.2.3(3)

In [1], Wil van der Aalst has proposed a characterization result for soundness. His result is based on liveness and boundedness which are intensively studied properties in Petri net theory. In 1998, this characterization has been extended to k-soundness by Barkaoui and Petrucci [5] . We present below this more general result.

Let Σ be a WF net and $k \geq 1$. Define the *k-closer* of Σ, denoted $\Sigma(k)$, as being the Petri net obtained from Σ by adding a new transition t^* connecting only the places o and i and such that $W(o, t^*) = W(t^*, i) = k$. When $k = 1$ we shall refer to $\Sigma(1)$ as being the *closer* of Σ.

Proposition 7.2.4 ([61]) Let Σ be a WF net and $k \geq 1$ be a natural number.

1. If Σ is k-sound then $[M_{ki}\rangle_{\Sigma} = [M_{ki}\rangle_{\Sigma(k)}$.
2. If Σ is k-sound then $\Sigma(k)$ is cyclic with respect to M_{ki} (i.e., M_{ki} is a home marking).

Proof Both properties in the proposition follow from the fact that the only reachable marking enabling t^* is M_{ko} and, by applying t^* the marking M_{ki} is produced. □

Now, we can prove the following characterization result.

Theorem 7.2.1 ([5]) A WF net Σ is k-sound iff its k-closer $\Sigma(k)$ is live and bounded with respect to M_{ki}.

Proof Assume first that Σ is k-sound. Then, Σ is bounded with respect to M_{ki} (by Proposition 7.2.3(1)) and $[M_{ki}\rangle_{\Sigma} = [M_{ki}\rangle_{\Sigma(k)}$ (by Proposition 7.2.4(1)). Therefore, $\Sigma(k)$ is bounded with respect to M_{ki}.

Moreover, $\Sigma(k)$ is quasi-live (because Σ is quasi-live with respect M_{ki} and t^* is quasi-live in $\Sigma(t)$ with respect to M_{ki}) and cyclic (by Proposition 7.2.4(2)) with respect to M_{ki}. Therefore, $\Sigma(k)$ is live with respect to M_{ki}.

Conversely, assume that $\Sigma(k)$ satisfies the properties in theorem. Let M be a marking reachable in Σ from M_{ki}. M is also reachable in $\Sigma(k)$ from M_{ki} and, due to the fact that $\Sigma(k)$ is live with respect to M_{ki}, there exists M' reachable in $\Sigma(t)$ from M such that $M'[t^*\rangle_{\Sigma(t)}$. Moreover, M' is reachable in Σ too, and $M' \geq M_{ko}$. If $M' > M_{ko}$ then one can easily see that $\Sigma(k)$ would be unbounded. Therefore, $M' = M_{ko}$ which shows that M_{ko} is reachable in Σ from M. This proves the first requirement in Definition 7.2.3.

The second requirement in Definition 7.2.3 follows easily from the liveness property of $\Sigma(k)$. Therefore, Σ is a k-sound WF net. □

7.3 Asynchronous Composition of Workflows

For the design and analysis of complex workflow nets, place/transition refinement seems to be a very good idea. In this context, several refinement operations have been introduced [1, 22]. All these refinements are based on a suitable chosen *composition operation*. For example, the transition refinement in [1] is based on the following composition operation on Petri nets. Given two Petri nets Σ_1 and Σ_2 such that they have in common only a subset S^c of places, their composition, denoted $\Sigma_1 \circ \Sigma_2$, is just the union of the Petri nets. Example in Figure 7.2 is from [1]. Both Petri nets Σ_1' and Σ_2 in the figure are generalized sound WF nets. The refinement of t in Σ_1' by Σ_2 means to remove t from Σ_1', let Σ_1 be the Petri nets obtained in this way, and then to compose Σ_1 and Σ_2 along $S^c = \{s_4, s_6\}$. The composition is given in Figure 7.3.

Concerning verification of WF nets we are interested in the "inverse" of the composition operation (and not in the inverse of the transition refinement operation). For example, the Petri net Σ in Figure 7.3 is a WF net which can be decomposed into Σ_1 and Σ_2 (but not into Σ_1' and Σ_2). Both of them are WF nets, but it might happen that one of them is not a WF net, as the example in Figure 7.4 shows us. Decomposing the Petri net Σ_1 in Figure 7.2 along $\{s_1, s_5\}$ we get the Petri nets in Figure 7.4. Σ_1^2 is a WF net, but Σ_1^1 is not. In general, this fact is not very important for our purposes and it will be addressed later.

By decomposing Σ as above, we want to verify properties of it by analyz-

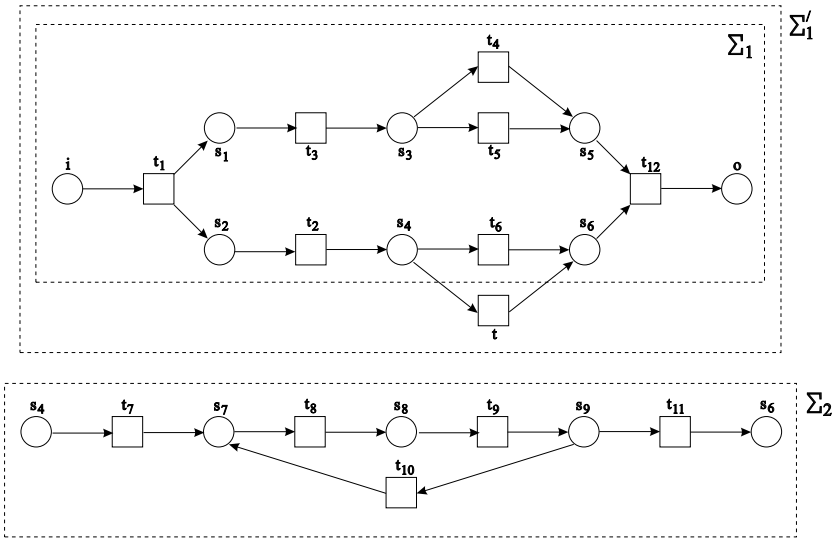


Figure 7.2: Two WF nets, Σ_1' and Σ_2

ing only Σ_1. The behavior of Σ_1 in Σ is highly influenced by Σ_2 by means of places in S^c. The good thing is that Σ_2 is a bounded WF net (being a generalized WF net) and, consequently, the only transformation it can generate is to move one token from s_4 to s_6 (assuming that M_i is the initial marking of Σ). This transformation can be described by using a binary relation R on S^c as given in Figure 7.5. The couple $(\Sigma_1, (\{s_4, s_6\}, R))$ acts as an *abstraction* of $\Sigma = \Sigma_1 \circ \Sigma_2$. It is in fact a jumping net (Section 5.2) which is neither labeled nor marked. They do not need to be labeled or marked because workflow nets are neither labeled nor marked.

Definition 7.3.1 A *WF module* is any jumping net $\mathcal{J} = (\Sigma, \mathcal{R})$ satisfying the following two properties:

(i) There are two special places i and o with the same properties as for WF nets;

(ii) Any node $x \in S \cup T$ in the graph associated to Σ is on a path from i to o, or there is $(S^c, R) \in \mathcal{R}$ and $y \in S^c$ such that $x \in S^c$ and there is a path from i to x and from y to o, or vice versa (i.e., a path from i to y and from x to o).

The concept of soundness for WF modules is introduced as for WF nets by taking into consideration the transition relation of WF modules.

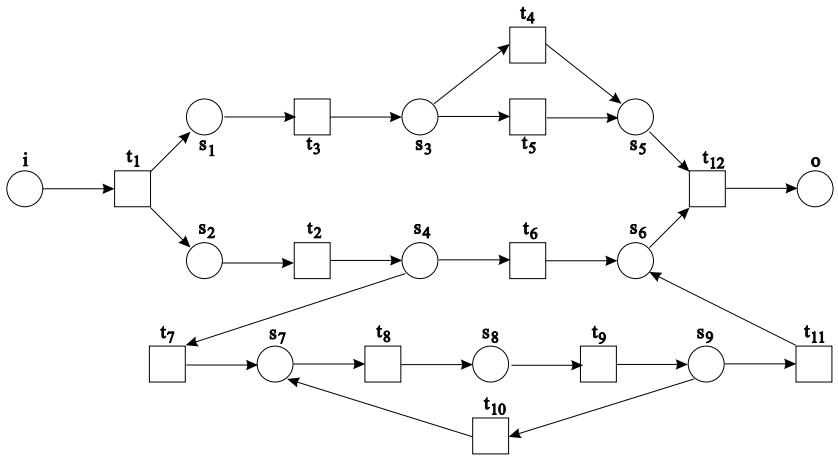


Figure 7.3: $\Sigma = \Sigma_1 \circ \Sigma_2$ obtained by refining t in Σ_1' by Σ_2

Example 7.3.1

(1) The couple $\mathcal{J}_1 = (\Sigma_1, (\{s_4, s_6\}, R))$ in Figure 7.5 is a WF module. It acts as an abstraction of $\Sigma_1 \circ \Sigma_2$.

(2) If we compose the Petri nets Σ_1^1 and Σ_1^2 in Figure 7.4 along $\{s_1, s_5\}$ we get the Petri net Σ_1 in Figure 7.2. Let R' be the binary relation on $\mathbf{N}^{\{s_1,s_5\}}$ given by

$$R' = \{((1,0),(0,0)),((0,0),(0,1)),((1,0),(0,1))\}.$$

It can be easily verified that $\mathcal{J}_1^1 = (\Sigma_1^1, (\{s_1, s_5\}, R'))$ is a WF module. This WF module acts as an abstraction of $\Sigma_1 = \Sigma_1^1 \circ \Sigma_1^2$.

7.4 From Workflow Nets to Kripke Structures

Our main goal is to set up a model checking technique for (structurally, generalized, n-) sound WF nets based on abstraction. As such WF nets are bounded, most of the constructions in this paragraph hold for bounded Petri nets as well. The main difference between bounded Petri nets and n-sound WF nets, for instance, consists in the fact that the relation R obtained by decomposing n-sound WF nets is very small. This is because n-sound WF nets start from a marking M_{ni} and end up with a marking M_{no}.

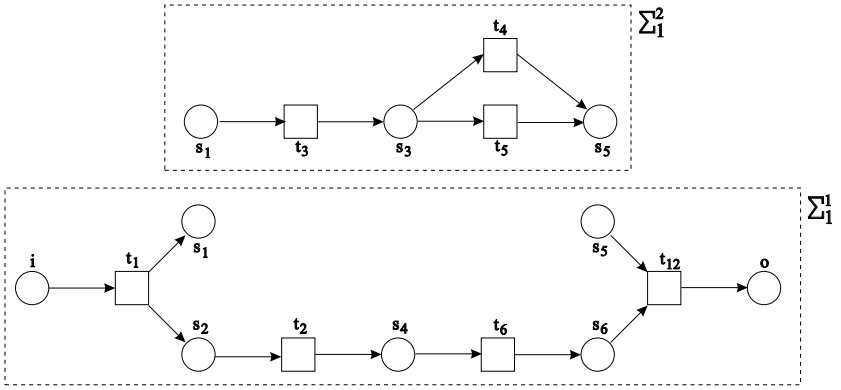


Figure 7.4: $\Sigma_1 = \Sigma_1^1 \circ \Sigma_1^2$, Σ_1^2 is a WF net, but Σ_1^1 is not a WF net

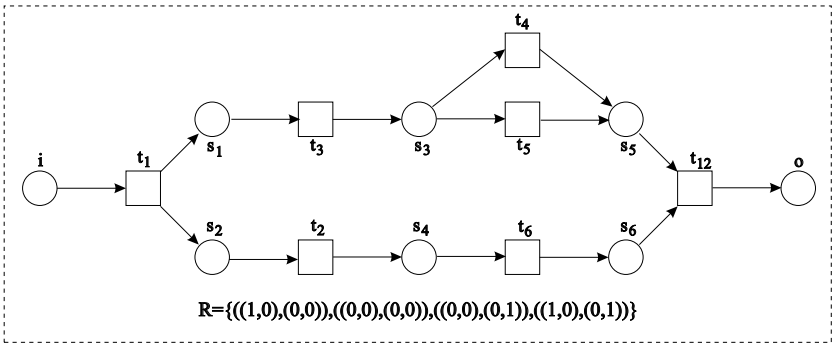


Figure 7.5: $(\Sigma_1, (\{s_4, s_6\}, R))$ acts as an abstraction of $\Sigma = \Sigma_1 \circ \Sigma_2$

The results we are going to present in this section extend properly the results in Chapter 6. The extension consists in the fact that arbitrary sets $\mathcal{R}$ of jumps are considered and not only singleton sets. In this way, the results developed in this sections are as general as possible (the fact that jumping nets are neither labeled nor marked is not a restriction at all).

We use the same construction as in Section 6.4 which associates a Kripke structure without fairness constraints to any bounded marked net (Σ, M_0). Then, extend this construction to *bounded jumping nets* $\mathcal{I} = (\Sigma, \mathcal{R})$ with respect to M_0 (i.e., Σ is bounded with respect to M_0) by simply adding the external transition relation

$$\rho^e = \{(M, M') \in \mathbf{N}^S \times \mathbf{N}^S | (\exists (S^c, R) \in \mathcal{R})$$
$$(M'|_{S^i} = M|_{S^i} \ \wedge \ (M|_{S^c}, M'|_{S^c}) \in R)\}$$

to the Kripke structure $K(\Sigma, M_0)$. Moreover, the set S^c is regarded as a set of external (interface) variables, for any $(S^c, R) \in \mathcal{R}$. We denote this structure by $K(\mathcal{J}, M_0)$.

We suppose from now on that for every net or jumping net a set of atomic propositions is given (referring to its set of markings). Moreover, we assume that whenever we merge (combine) two markings M_1 and M_2 which agree on some places (in order to obtain a marking of the composed net or jumping net), the propositions that are satisfied at the new marking are exactly those that are satisfied at M_1 and M_2.

Example 7.4.1

(1) The Kripke structure associated to the net $\Sigma = \Sigma_1 \circ \Sigma_2$ in Figure 7.3, together with the initial marking M_i, is pictorially represented in Figure 7.6 (each state/marking is represented only by the places marked by it). It has 20 states.

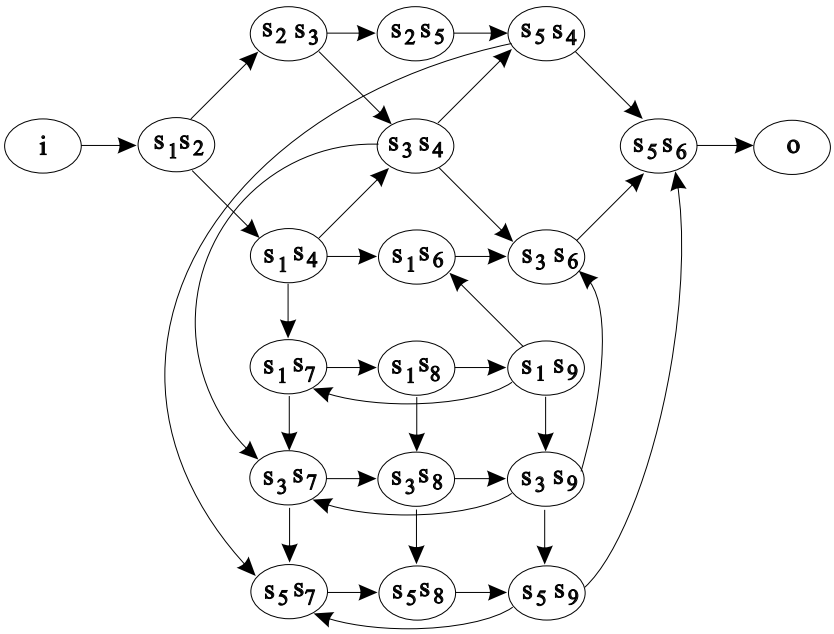


Figure 7.6: The Kripke structure associated to (Σ, M_i) in Figure 7.3

(2) The Kripke structure $K(\mathcal{J}_1, M_i)$ associated to the WF module $\mathcal{J}_1$ in Example 7.3.1(1) is given in Figure 7.7. It has only 14 states. We have

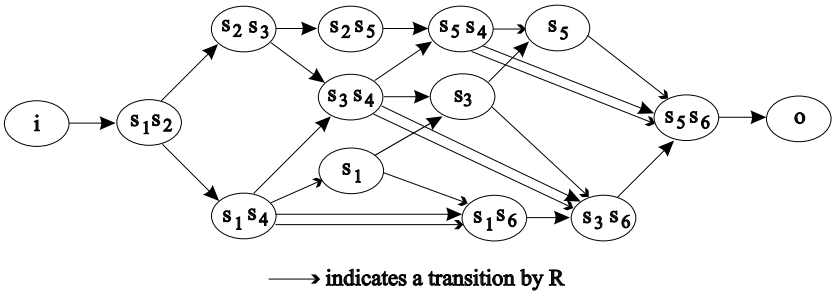


Figure 7.7: The Kripke structure $K(\mathcal{J}_1, M_i)$

mentioned that $\mathcal{J}_1$ is an abstraction of the Petri net Σ in (1), and this fact is reflected at the Kripke structure level as well.

(3) If we consider the initial state M_{2i} for the Petri net in (1), then the associated Kripke structure has about 200 states. The Kripke structure associated to $\mathcal{J}_1$, with the same initial marking M_{2i}, has less than 60 states. We will see later that we can transfer properties from $\mathcal{J}_1$ to Σ. This shows us clearly how important is such an abstraction when we wish to verify temporal properties of WF nets by model checking.

We may also add a set $\mathcal{F}$ of fairness constraints to all Kripke structures obtained as above. We will denote the newly obtained structures by $K(\Sigma, M_0, \mathcal{F})$ and $K(\mathcal{J}, M_0, \mathcal{F})$. The tuples $(\Sigma, M_0, \mathcal{F})$ and $(\mathcal{J}, M_0, \mathcal{F})$ will be called *fair nets* and *fair jumping nets*, respectively. The simulation and satisfaction relations are defined for them by means of the structures they induce. For example, for two fair bounded jumping nets $(\mathcal{J}_1, M_1, \mathcal{F}_1)$ and $(\mathcal{J}_2, M_2, \mathcal{F}_2)$ we write:

- $(\mathcal{J}_1, M_1, \mathcal{F}_1) \prec_{\mathcal{A}} (\mathcal{J}_2, M_2, \mathcal{F}_2)$ for $K(\mathcal{J}_1, M_1, \mathcal{F}_1) \prec_{\mathcal{A}} K(\mathcal{J}_2, M_2, \mathcal{F}_2)$, and
- $(\mathcal{J}_1, M_1, \mathcal{F}_1) \models \varphi$ for $K(\mathcal{J}_1, M_1, \mathcal{F}_1) \models \varphi$.

Now we have to define what an abstraction of a jumping net is. We need first the concept of a *narrowed relation*. If R is a binary relation on a set $\mathbf{N}^A$ and $B \subset A$, then the *narrowed relation associated to R and B*, denoted $R/_B$, is the relation obtained from R by removing all the components that are not in B. For example, if $A = \{a_1, a_2, a_3\}$, $R = \{((1,2,0),(2,3,1))\}$ and $B = \{a_1, a_3\}$, then $R/_B = \{((1,0),(2,1))\}$.

Now, let $(\mathcal{J}, M_0, \mathcal{F})$ be a fair jumping net which is a composition of two nets along a subset S^c of places,

$$(\mathcal{J}, M_0, \mathcal{F}) = ((\Sigma_1 \circ \Sigma_2, \mathcal{R}), M_0, \mathcal{F}).$$

Define the fair jumping net $(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2}) = ((\Sigma_1, \mathcal{R}_{1,2}), M_{1,2}, \mathcal{F}_{1,2})$ as follows:

- $\mathcal{R}_{1,2} = \mathcal{R}_2 \cup \mathcal{R}'$, where:
  1. $\mathcal{R}_2$ is the set of all pairs $(M|_{S^c}, M'|_{S^c})$, where M is reachable from M_0 in $(\Sigma_1 \circ \Sigma_2, \mathcal{R})$ and M' is reachable from M by transitions in Σ_2, but at least one transition occurrence. This relation will be called the *relation induced by* Σ_2 *in* Σ *with respect to* M_0;
  2. $\mathcal{R}' = \{(S - S_2^i, R/_{S-S_2^i})|(S, R) \in \mathcal{R}\}$;
- $M_{1,2} = M_0|_{S_1}$;
- $\mathcal{F}_{1,2} = \{\{M|_{S_1}|M \in A\}|A \in \mathcal{F}\}$.

In general, $\mathcal{J}_{1,2}$ has more behavior than $\mathcal{J}$. The next example shows this.

Example 7.4.2 Let $(\mathcal{J}, M_0) = ((\Sigma, \mathcal{R}), M_0)$, where Σ and M_0 are given in Figure 7.8(a) and $\mathcal{R}$ contains only one binary relation

$$\mathcal{R} = \{(\{s_1, s_4\}, \{((0,1),(1,0))\})\}.$$

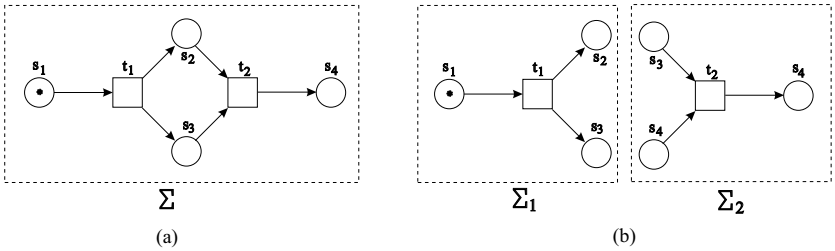


Figure 7.8: Constructing $\mathcal{J}_{1,2}$

Consider now $S^c = \{s_2, s_3\}$ and decompose the net Σ along it (Figure 7.8(b)). The jumping net $\mathcal{J}_{1,2} = (\Sigma_1, \mathcal{R}_{1,2})$ is given by

$$\mathcal{R}_{1,2} = \{(\{s_1\}, \{((0),(1))\}), (\{s_2, s_3\}, \{((1,1),(0,0))\})\}$$

Any reachable marking in $\mathcal{J}$ has the first three components of the form $(1,0,0)$, $(0,1,1)$ and $(0,0,0)$. In $\mathcal{J}_{1,2}$, besides these markings, $(0,k,k)$ and $(1,k,k)$ are reachable markings, for all $k \geq 1$.

Example 7.4.3 We will give an example of a two step abstraction applied to the WF net Σ in Figure 7.3 (for simplicity, fairness constraints are omitted).

In the first step, we view Σ as a composition of two Petri nets

$$(\mathcal{J}, M_i) = ((\Sigma_1 \circ \Sigma_2), \mathcal{R}), M_i),$$

where the composition is along $\{s_4, s_6\}$ and $\mathcal{R} = \emptyset$. We abstract from Σ_2 and we get the fair jumping net

$$(\mathcal{J}', M_i') = ((\Sigma_1, \mathcal{R}'), M_i'),$$

where $\mathcal{R}' = \{(\{s_4, s_6\}, R)\}$, R is the relation in Figure 7.5, and $M_i' = M_i|_{S_1}$.

In the second step, we write Σ_1 as a composition of two Petri nets as in Figure 7.4,

$$((\Sigma_1, \mathcal{R}'), M_i') = ((\Sigma_1^1 \circ \Sigma_1^2, \mathcal{R}'), M_i').$$

Now, we abstract from Σ_1^2 and we get

$$(\mathcal{J}'', M_i'') = ((\Sigma_1^1, \mathcal{R}''), M_i''),$$

where $\mathcal{R}'' = \{(\{s_4, s_6\}, R), (\{s_1, s_5\}, R')\}$, R' is the binary relation in Example 7.3.1(2), and $M_i'' = M_i'|_{S_1^1}$.

The Kripke structure associated to $(\mathcal{J}'', M_i'')$ is given in Figure 7.9. It has 13 states.

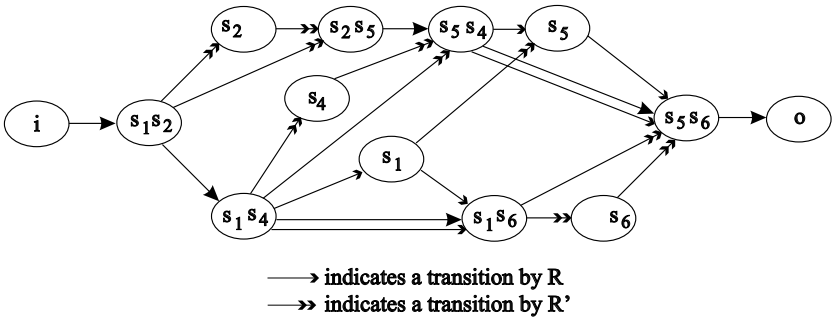


Figure 7.9: The Kripke structure $K(\mathcal{J}'', M_i'')$

The following theorem makes the connection between jumping nets and structures.

Theorem 7.4.1 ([59]) If $(\mathcal{J}, M_0, \mathcal{F})$ and $(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2})$ given as above are bounded jumping nets, then the following holds true

$$(\mathcal{J}, M_0, \mathcal{F}) \prec_{\mathcal{A}_1} (\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2})$$

($\mathcal{A}_1$ denotes the set of atomic propositions associated to Σ_1).

Proof Due to the fact that states of Kripke structures associated to Petri nets or jumping nets are markings, we will prove the theorem is working directly with Petri nets and jumping nets.

Let $(\mathcal{J}, M_0, \mathcal{F}) = ((\Sigma_1 \circ \Sigma_2, \mathcal{R}), M_0, \mathcal{F})$ and

$$H = \{(M, M|_{S_1}) | M \in [M_0\rangle_{\mathcal{J}}\}.$$

We show that H is a simulation from $(\mathcal{J}, M_0, \mathcal{F})$ to $(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2})$ with respect to $\mathcal{A}_1$.

By restricting the markings to S_1, we may assume without any loss of generality that all the atomic properties in $\mathcal{A}_1$ (specific to Σ_1) are preserved. That is, we may assume that $\mathcal{L}(M) \cap \mathcal{A}_1 = \mathcal{L}(M|_{S_1}) \cap \mathcal{A}_1$ holds true for all $M \in [M_0\rangle_{\mathcal{J}}$.

Let $\sigma = M_0 M_1 M_2 \cdots$ be a fair path in $(\mathcal{J}, M_0, \mathcal{F})$. There are $x_0, x_1, \ldots$ such that $M_i[x_i\rangle_{\mathcal{J}} M_{i+1}$ for all $i \geq 0$, where x_i is either a tansition or an element (a pair of markings) of some binary relation in $\mathcal{R}$ (according to the definition of the transition relation $[\cdot\rangle_{\mathcal{J}}$).

Decompose the path σ,

$$\sigma = M_{i_0} \cdots M_{i_1} \cdots M_{i_2} \cdots$$

such that $i_0 = 0$ and, for all $j \geq 0$, the following requirements are satisfied:

(a) if $i_{j+1} = i_j + 1$, then x_{i_j} is either a transition in Σ_1 or an element of some binary relation in $\mathcal{R}$;

(b) if $i_{j+1} \geq i_j + 1$, then $x_{i_j}, \ldots, x_{i_{j+1}-1}$ are transitions in Σ_2 and $M_{i_j}|_{S_1} = \cdots = M_{i_{j+1}-1}|_{S_1}$ (this property is necessary in order to preserve the fairness constraints). Remark that $M_{i_{j+1}-1}|_{S_1^i} = M_{i_{j+1}}|_{S_1^i}$ but it could be the case that $M_{i_{j+1}-1}|_{S^c} \neq M_{i_{j+1}}|_{S^c}$.

Consider now the sequence

$$\sigma' = M_{i_0}|_{S_1} M_{i_1}|_{S_1} M_{i_2}|_{S_1} \cdots$$

It is straightforward to see that σ' is a valid path in $(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2})$ and, moreover, it is fair (for all $A \in \mathcal{F}$ and $M \in A$ such that σ passes infinitely many times through M, σ' will pass infinitely many times through $M|_{S_1}$).

As $H(\sigma, \sigma')$ trivially holds true, we conclude that H is a simulation from $(\mathcal{J}, M_0, \mathcal{F})$ to $(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2})$ with respect to $\mathcal{A}_1$. □

Corollary 7.4.1 ([59]) If $(\mathcal{J}, M_0, \mathcal{F})$ and $(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2})$ given as above are bounded jumping nets, then

$$(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2}) \models \overline{\varphi} \quad \Rightarrow \quad (\mathcal{J}, M_0, \mathcal{F}) \models \hat{\overline{\varphi}},$$

for any $\forall CTL^*$ formula φ over the set of atomic proposition of Σ_1.

Proof Directly from Theorem 7.4.1 and Corollary 6.2.1. □

7.5 Separating Bounded Modules from Bounded Modules

The main problem in using Theorem 7.4.1 and its corollary is to find a method for decomposing bounded jumping nets $(\mathcal{J}, M_0, \mathcal{F})$ so that $(\mathcal{J}_{1,2}, M_{1,2}, \mathcal{F}_{1,2})$ is a bounded jumping net too.

We begin by the following remark. If Σ is a bounded Petri net with respect to M_0, then any decomposition $\Sigma = \Sigma_1 \circ \Sigma_2$ leads to bounded Petri nets Σ_1 and Σ_2. But, $(\Sigma_1, \mathcal{R}_{1,2})$ might be unbounded with respect to $M_0|_{S_1}$, as Example 7.4.2 or the decomposition in Figure 7.10 shows us. The situation in Figure 7.10 can be explained as follows. Σ is a safe net (with respect to the marking in the figure). Σ_2 may insert a token to s_3 only when s_4 is marked (it contains a token). The relation $\mathcal{R}_{1,2}$ contains only one pair $(\{s_2, s_3\}, R)$, where

$$R = \{((1,0),(0,0)), ((0,0),(0,1)), ((1,0),(0,1))\}.$$

In $(\Sigma_1, \mathcal{R}_{1,2})$, a token to s_3 may be inserted whenever s_2 is unmarked or marked by exactly one token (without taking into consideration s_4 which does not exist anymore in this jumping net). Therefore,

$$(0,1,0)R(0,0,1)[t_1\rangle(1,0,0)R(1,0,1)[t_1\rangle(2,0,0)\cdots$$

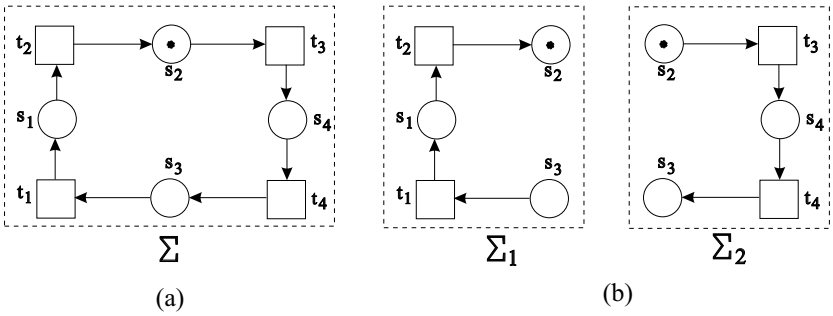


Figure 7.10: $(\Sigma, (0,1,0,0))$ is safe, but $((\Sigma_{1,2}, \mathcal{R}_{1,2}), (0,1,0))$ is unbounded

is a valid computation in $(\Sigma_1, \mathcal{R}_{1,2})$, and it shows that $(\Sigma_1, \mathcal{R}_{1,2})$ is not bounded with respect to $M_0|_{S_1}$.

We say that a jumping net $(\Sigma, \mathcal{R}) = (\Sigma_1 \circ \Sigma_2, \mathcal{R})$ is *context-free with respect to* Σ_2 *and* M_0 if for every pair $(M^c, \overline{M}^c)$ induced by Σ_2 and for every reachable marking M from M_0 in $(\Sigma, \mathcal{R})$, if $M|_{S^c} = M^c$ then Σ_2 can induce $(M^c, \overline{M}^c)$ at M (that is, a marking M' is reachable from M only by transitions in Σ_2 and $M'|_{S^c} = \overline{M}^c$).

The jumping net $(\Sigma, \emptyset) = (\Sigma_1 \circ \Sigma_2, \emptyset)$, where Σ, Σ_1 and Σ_2 are as in Figure 7.10, is not context-free with respect to Σ_2 and $(0,1,0,0)$ because $((0,0),(0,1))$ cannot be induced by Σ_2 (only by transitions in Σ_2) at $(1,0,0,0)$.

Theorem 7.5.1 ([59]) If $\mathcal{J} = (\Sigma_1 \circ_{S^c} \Sigma_2, \mathcal{R})$ is a jumping net satisfying the properties:

(1) it is bounded with respect to M_0;

(2) it is context-free with respect to Σ_2 and M_0;

(3) $S' \cap S_2^i = \emptyset$, for all $(S', R) \in \mathcal{R}$,

then $\mathcal{J}_{1,2} = (\Sigma_1, \mathcal{R}_{1,2})$ is a bounded jumping net with respect to $M_0|_{S_1}$.

Proof Let $\mathcal{R}_{1,2} = \mathcal{R}_2 \cup \mathcal{R}'$ be the set of binary relations obtained according to the definition of $\mathcal{J}_{1,2}$ (i.e., $\mathcal{R}_2$ is induced by Σ_2, and $\mathcal{R}'$ is obtained by narrowing $\mathcal{R}$). Due to the property (3) we have $\mathcal{R}' = \mathcal{R}$.

It is clear that for any marking M' reachable from $M_0|_{S_1}$ in $\mathcal{J}_{1,2}$ only by transitions or relations in $\mathcal{R}'$, there exists a reachable marking M from M_0

in $\mathcal{J}$ such that $M|_{S_1} = M'$. Therefore, $\mathcal{J}_{1,2}$ could be unbounded only if there were a marking M' reachable from $M_0|_{S_1}$ by using relations in $\mathcal{R}_2$ as well and $M' \neq M|_{S_1}$ for all $M \in [M_0\rangle_{\mathcal{J}}$. However, this is not possible because $\mathcal{J}$ is context-free with respect to Σ_2 and M_0 (any relation in $\mathcal{R}_2$ can be simulated be some sequences of transitions in $\mathcal{J}$). As a conclusion, $\mathcal{J}_{1,2}$ is bounded with respect to $M_0|_{S_1}$. □

If the second condition in the above theorem fails, then $\mathcal{J}_{1,2} = (\Sigma_1, \mathcal{R}_{1,2})$ might not be bounded with respect to $M_0|_{S_1}$, as the jumping net in Figure 7.10 shows us. Similarly, if the third condition in the above theorem fails, then $\mathcal{J}_{1,2} = (\Sigma_1, \mathcal{R}_{1,2})$ might not be bounded with respect to $M_0|_{S_1}$, as the jumping net in Figure 7.11 shows us. The jumping net $\mathcal{J} =$

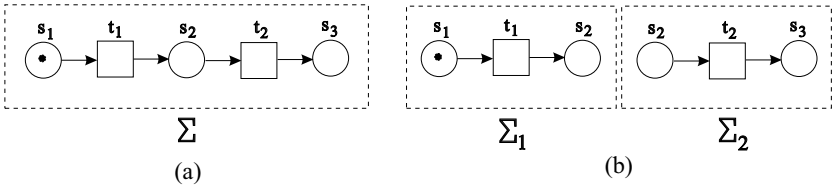


Figure 7.11: Requirement (3) in Theorem 7.5.1 is not fulfilled

$(\Sigma, \{(\{s_1, s_3\}, R)\})$, where Σ is the Petri net in Figure 7.11(a) and $R = \{((0,1),(1,0))\}$, is safe with respect to $M_0 = (1,0,0)$. If we decompose it along $S^c = \{s_2\}$ we get the Petri nets in Figure 7.11(b). The jumping net $\mathcal{J}_{1,2}$ is given by

$$\mathcal{R}_{1,2} = \{(\{s_1\}, \{((0),(1))\}), (\{s_2\}, \{((1),(0))\})\}.$$

The first pair in this set is obtained by narrowing R to $\{s_1\} = \{s_1, s_3\} - \{s_3\}$, while the second pair is the relation induced by Σ_2 into $\mathcal{J}$.

This jumping net is not bounded with respect to $M_0|_{\{s_1,s_2\}}$, as the following computation shows:

$$(1,0)[t_1\rangle(0,1)R/_{\{s_1\}}(1,1)[t_1\rangle(0,2)R/_{\{s_1\}}(1,2)\cdots$$

Theorem 7.5.2 ([59]) If $\mathcal{J} = (\Sigma_1 \circ_{\{i,o\}} \Sigma_2, \mathcal{R})$ is a jumping net satisfying the properties:

(1) it is n-bounded with respect to M_0, for some $n \geq 1$;

(2) Σ_2 is a k-sound WF net, for all $k \leq n_0$, where $n_0 = max\{M(i) | M \in [M_0\rangle_{\mathcal{J}}\}$ (i is the input place and o is the output place of Σ_2);

(3) $S' \cap S_2^i = \emptyset$, for all $(S', R) \in \mathcal{R}$,

then $\mathcal{J}_{1,2} = (\Sigma_1, \mathcal{R}_{1,2})$ is a bounded jumping net with respect to $M_0|_{S_1}$.

Proof Let $\mathcal{R}_{1,2} = \mathcal{R}_2 \cup \mathcal{R}'$ be the set of binary relations obtained according to the definition of $\mathcal{J}_{1,2}$ (i.e., $\mathcal{R}_2$ is induced by Σ_2, and $\mathcal{R}'$ is obtained by narrowing $\mathcal{R}$ by S_2^i).

Due to the fact that $\mathcal{J}$ satisfies property (3) in the theorem, it is clear that for any marking M' reachable from $M_0|_{S_1}$ in $\mathcal{J}_{1,2}$ only by transitions or relations in $\mathcal{R}'$, there exists a reachable marking M from M_0 in $\mathcal{J}$ such that $M|_{S_1} = M'$ (this is similar to the same case in the proof of Theorem 7.5.1).

Let us consider now the relation in $\mathcal{R}_2$. We will write the markings of $\mathcal{J}$ in the form (M_1, x, M_2, y), where M_1 is a marking on S_1^i, x is on i, M_2 is a marking on S_2^i, and y is on o. Let $\underline{0}$ be the $|S_2^i|$-dimensional zero vector.

Because Σ_2 satisfies the property 2), its markings reachable from $(k, \underline{0}, 0)$ are of the form $(k-p, M_2, q)$ for some M_2 and for all $0 \leq q \leq p \leq k \leq n_0$. Therefore, Σ_2 can induce the pair $((x, y), (x-p, y+q))$ at any marking of the form $(M_1, x, \underline{0}, y)$, where $0 \leq q \leq p \leq x \leq n_0$. At markings (M_1, x, M_2, y), Σ_2 may induce pairs $((x, y), (x-p, y+q+r))$, where $0 \leq q \leq p \leq x \leq n_0$ and $y + q + r \leq n$ (r depends on M_2).

Now, it is easy to see that for any M' reachable from $M_0|_{S_1}$ by using the relation in $\mathcal{R}_2$ as well, there is $M \in [M_0\rangle_{\mathcal{J}}$ such that $M' \leq M|_{S_1}$.

As a conclusion, $\mathcal{J}_{1,2} = (\Sigma_1, \mathcal{R}_{1,2})$ is a bounded jumping net with respect to $M_0|_{S_1}$ because $\mathcal{J}$ is bounded with respect M_0. □

It is worth to mention that the property (2) in Theorem 7.5.2 does not imply, in general, that $\mathcal{J}$ is context-free with respect to Σ_2 and M_0.

Of course, Theorem 7.5.2 holds when Σ_2 is generalized sound as well.

As a conclusion, if we have an n-bounded WF jumping net and we are able to separate from it a WF net as in Theorem 7.5.2, then Corollary 7.4.1 can be applied to it in order to verify properties that can be expressed by our temporal logic. For example, the jumping nets in Example 7.4.3 satisfy the hypothesis of Theorem 7.5.2 and, therefore, Corollary 7.4.1 can be applied to them.

Conclusions

The aim of this thesis was to introduce Petri net reactive modules as models of reactive systems and to study basic properties of them in order to help modular analysis of Petri net models. We have focused on two classes of problems: semantics of Petri net modules and correctness proof techniques. A throughout discussion about related work is in order.

Nets equipped with subsets of interface places (modules, in our paper) appear naturally when a distributed systems is modeled as a set of actors communicating through buffers by message passing. In this context, composition of nets by merging places (asynchronous composition, in our paper) is an important operation. In literature, different variants of modules and asynchronous compositions have been considered. In [10], the modules (called there *open interface nets*) are non-labeled and endowed with a set of markings on the internal places (called stable states). Our modules are exactly those from [66] (called there *host nets*) or [64] (called there *net components*), with the difference that in [64] they are not labeled; the asynchronous composition for modules we considered is like in [66] (called there *place composition*). The set of interface places may be partitioned into subsets of input and output places as in [30]. The concept of an e-module is a new one; however, the idea of considering the interaction between a module and an environment has been touched on in [66] (by adding two transitions t_s^- and t_s^+, for each interface place s), in [64] (by means of *actions*, which are jumps in our paper), and in [30], but in a totally different way and with different purposes than ours. The terminology of *Petri net reactive module*, as we considered, seems to be the most adequate one in the context of modeling reactive systems which may interact with each other.

We have considered the (plain) process semantics together with a notion of process isomorphism (different than the classical one), suitable in proving correctness of Petri net transformations. It was shown that processes of

composed nets can be decomposed in processes of "shifted" components (that is, components whose initial markings are increased), and vice versa. Clearly, this semantics is not compositional with respect to the composition operation we considered, but with a little effort we were able to obtain a compositional one (Theorem 2.4.1); it is totally different than the semantics considered in the papers cited above. For example, the process semantics in [30] was suitable modified such that the compositional property was achieved, and the CFFD-semantics in [64] is a conjunction of stable failures, divergence traces, and infinite traces (which lead to compositionality).

The main line we have followed in Chapter 3 is a classical one: find two equivalence relations $\approx_1$ and $\approx_2$ such that from $\gamma_1 \approx_1 \gamma_2$ one can infer $\gamma \approx_2 \gamma[\gamma_1 \leftarrow \gamma_2]$. We have proposed two pairs of such equivalences; they are based on our semantics and, therefore, they are different than the others known from literature. We believe that they are very suitable in proving the correctness of structural transformations of Petri nets, as this chapter proves it by applying them to prove correctness of several Petri net transformations.

Research Subject 1 *We have showed that our correctness proof method cannot be used to prove the correctness of any Petri net structural transformation. Therefore, a more deeper insight on the nature of these transformations should be achieved. This one is an interesting further research subject.*

In order to develop a model checking technique for Petri net modules, we have partitioned the transition relation of Kripke structures into two parts, internal and external. This allows an adequate modeling of reactive systems. We have propose a simulation preorder suitable for use with such Kripke structures. The preorder captures the relation between a component and a system containing that component, treating the transition relation of that component as internal in the system. We have identified a substantial subset of $\forall CTL^*$, which is appropriate to be used with our preorder. We have also proposed an abstraction method, and also illustrated its applications to Petri net reactive modules. There are two parts of this method: first, we decompose the system into modules and compute the relation induced by some submodules, and second, we check the satisfaction of properties in the "augmented" modules.

Research Subject 2 *Finding efficient methods to describe or approximate the relation induced by submodules is of great importance for practical ap-*

plications. For Petri net reactive modules, we have discussed briefly how to approximate the relation induced by a component into a system. Possible future work is to identify classes of systems with good properties for automated approximation.

We have showed that our model checking technique can be applied to model check workflow nets. In order to do that we have decomposed the original workflow nets into two subnets, and then augmented one of them by the relation induced by the other one. We have proved that, when one of the subnets is a k-sound worflow net, for any $k \leq n$ where n is the upper bound for all reachable marking components of the original workflow net, then the workflow module is bounded. Moreover, the relation induced by a k-sound workflow net is very small, thus such a decomposition leads to a substantial reduction of the effort required for model checking. The set of properties of the workflow module we are able to prove by model checking, using this technique, can be expressed by a substantial fragment of $\forall CTL^*$. As a common practice when using abstraction, our technique is also based on a simulation relation from the original workflow net to the abstract one (the workflow module). This simulation preserves the delayed version of our formulas. Many interesting properties are invariant to delaying.

Research Subject 3 *One remaining problem in the context of model checking workflow nets is to find more decomposition criteria leading to bounded modules and induced relations which can be easily computed for the class of workflow nets mentioned above.*

Bibliography

[1] V.M.P. van der Aalst. *Structural Characterization of Sound Workflow Nets*, Computing Science Report 96/23, Eindhoven University of Technology, 1996.

[2] V.M.P. van der Aalst. *Three Good Reasons for Using a Petri-net-based Workflow Management System*, Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), Cambridge (Massachusetts), 1996, 179–201.

[3] V.M.P. van der Aalst, A. Hirnschall, H.M.W. Verbeek. *An Alternative Way to Analyze Workflow Graphs*, Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02), Lecture Notes in Computer Science 2348, 2002, 535-552.

[4] R. Alur, Th.A. Henziger: *Reactive Modules*, in: Proc. of the 11th IEEE Symposium on Logic in Computer Science LICS, 1996, 207–218.

[5] K. Barkoui, L. Petrucci. *Structural Analysis of Workflow Nets with Shared Resources*, Proceeding of the Workshop "Workflow Management: Net-based Concepts, Models, Techniques and Tools WFM'98", Lisbon (portugal), june 1998, 82–95.

[6] S. Berezin, S. Campos, E.M. Clarke: *Compositional Reasoning in Model Checking*, in: Proc. of the International Symposium "Compositionality: The Significant Difference" COMPOS'97, Bad Malente (Germany), Sept 8–12, 1997, Lecture Notes in Computer Science 1536, Springer-Verlag, 1998, 81–102.

[7] W. Brauer, R. Gold, W. Vogler: *A Survey of Behaviour and Equivalence Preserving Refinements of Petri Nets*, in: Advances of Petri Nets 1990, Lecture Notes in Computer Science 483, Springer-Verlag, 1990, 1–46.

[8] E. Best, R. Devillers, A. Kiehn, L. Pomelo: *Concurrent Bisimulations in Petri Nets*, Acta Informatica 28, 1991, 231–264.

[9] E. Best, C. Fernandez: *Nonsequential Processes. A Petri Net Point of View*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1988.

[10] G. Chehaibar: *Replacement of Open Interface Subnets and Stable State Transformation Equivalence*, in: Advances in Petri Nets 1993, Lecture Notes in Computer Science 674, Springer-Verlag, 1993, 1–25.

[11] E.M. Clarke, E.A. Emerson: *Synthesis of Synchronizations Skeletons for Branching Time Temporal Logic*, in: Workshop on Logic of Programs, Yorktown Heights, May 1981, Lecture Notes in Computer Science 131, Springer-Verlag, 1981.

[12] E.M. Clarke, O. Grumberg, D.E. Long: *Model Checking*, in: Model Checking, Abstraction and Composition, vol 152 of NATO ASI Series F, Springer-Verlag, 1996, 477-498.

[13] W. Damm, G. Döhmen, V. Gerstner: *Modular Verification of Petri Nets. The Temporal Logic Approach*, in: Proc. of the REX Workshop on Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness (J.W. Bakker, W.-P. de Roever, G. Rozenberg, eds.), Lecture Notes in Computer Science 430, Springer-Verlag, 1989, 180–207.

[14] J. Desel, W. Reisig: *Place/Transition Petri Nets*, in: Lectures on Petri Nets I: Basic Models (W. Reisig, G. Rozenberg, eds.), Lecture Notes in Computer Science 1491, Springer-Verlag, 1998, 122–173.

[15] J. Desel: *Validation of System Models Using Partially Ordered Runs*, in: Business Process Management-Models, Techniques and Empirical Studies (W.M.P. van der Aalst, J. Desel, A. Oberweis, eds.), Lecture Notes in Computer Science, Springer-Verlag, 1999.

[16] C.A. Ellis, K. Keddara, G. Rozenberg. *Dynamic Change Within Workflow Systems*, Proceedings of the International Conference on Organizational Computing Systems, ACM SIGOIS, 1995, 10–21.

[17] C.A. Ellis, G.J. Nutt. *Modelling and Enactment of Workflow Systems*, Proceedings of the 14th International Conference on Application and

Theory of Petri Nets, Lecture Notes in Computer Science 691, 1993, 1–16.

[18] J. Esparza, S. Melzer: *Model Checking LTL Using Constraint Programming*, Technical Report, Technische Universität München, 1997.

[19] D. Frutos-Escrig, C. Johnen: *Decidability of Home Space Property*, LRI, Technical Report LRI 503, 1989.

[20] R.J.v. Glabbeck, U. Golz: *Equivalence Notions for Concurrent Systems and Action Refinement*, in: Mathematical Foundations of Computer Science 1989, Lecture Notes in Computer Science 379, Springer-Verlag, 1989, 237–248.

[21] O. Grumberg, D.E. Long: *Model Checking and Modular Verification*, ACM Transactions on Programming Languages and Systems 16, 1994, 843–871.

[22] K. van Hee, N. Sidorova, M. Voorhoeve. *Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach*, Proceedings of the 24th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science 2679, 2003, 337–356.

[23] K. van Hee, N. Sidorova, M. Voorhoeve. *Generalised Soundness of Workflow Nets is Decidable*, Proceedings of the 25th International Conference on Application and Theory of Petri Nets, Bologna (Italy), 2004.

[24] B. Josko: *Verifying the Correctness of AADL-Modules Using Model Checking*, in: Proc. of the REX Workshop on Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness (J.W. Bakker, W.-P. de Roever, G. Rozenberg, eds.), Lecture Notes in Computer Science 430, Springer-Verlag, 1989, 386–400.

[25] T. Jucan, F.L. Ţiplea: *Petri Nets. Theorie and Practice*, Romanian Academy Press, Bucharest, 1999.

[26] C. Karamanolis, D. Giannakopoulou, J. Magee, S.M. Wheater. *Model Checking of Workflow Schemas*, Fourth International Enterprise Distributed Object Computing Conference, Mahukari (Japan), September 25–28, 2000.

[27] Y. Kersten, A. Pnueli, L. Raviv: *Algorithmic Verification of Linear Temporal Logic Specifications*, in: Proc. of the 25th International Colloquium on Automata, Languages, and Programming ICALP'98, Lecture Notes in Computer Science 1443, Springer-Verlag, 1998, 1–16.

[28] Y. Kersten, A. Pnueli: *Modularization and Abstraction: The Keys to Practical Formal Verification*, in: Proc. of the 23rd International Symposium on Mathematical Foundations of Computer Science MFCS'98, Lecture Notes in Computer Science 1450, Springer-Verlag, 1998, 54–71.

[29] A. Kiehn: *Petri Net Systems and their Closure Properties*, in: Advances in Petri Nets 1989, Lecture Notes in Computer Science 424, Springer-Verlag, 1990, 306–328.

[30] E. Kindler: *A Compositional Partial Order Semantics for Petri Net Components*, in Proc. of the 18th International Conference on Application and Theory of Petri Nets, Toulouse (France), Lecture Notes in Computer Science 1248, Springer-Verlag, 1998, 235–252.

[31] O. Kupferman, M.Y. Vardi: *Modular Model Checking*, in: Proc. of the International Symposium "Compositionality: The Significant Difference" COMPOS'97, Bad Malente (Germany), Sept 8–12, 1997, Lecture Notes in Computer Science 1536, Springer-Verlag, 1998, 381–401.

[32] B. Kurshan: *Analysis of Discrete Event Coordination*, in: Proc. of the REX Workshop on Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness (J.W. Bakker, W.-P. de Roever, G. Rozenberg, eds.), Lecture Notes in Computer Science 430, Springer-Verlag, 1989, 414-453.

[33] H. Lin, Z. Zhao, H. Li, Z. Chen. *A Novel Graph reduction Algorithm to Identify Structural Conflicts*, Proceedings of the 35th Hawaii International Conference on System Sciences, IEEE Computer Society Press, 2002.

[34] Z. Manna, A. Pnueli: *The Temporal Logic of Reactive and Concurrent Systems. Specification*, Springer-Verlag, 1992.

[35] D. C. Marinescu: *Internet-Based Workflow Management: Towards a Semantic Web*. Wiley, New York, NY, 2002, 627+xxiii pages.

[36] R. Melinte, O. Oanea, I. Olga, F. L. Ţiplea: *The Home Marking Problem and Some Related Concepts*, Acta Cybernetica 15 (3), 2002, 467-478.

[37] G. De Michelis, C. Ellis, G. Memmi (Eds.). *Proceedings of the 2nd Workshop on Computer-Supported Cooperative Work, Petri Nets and Related Formalisms*, Zaragoza (Spain), 1994.

[38] K. Müller: *Constructable Petri Nets*, Journal of Information Processing and Cybernetics EIK 21, 1985, 171–199.

[39] E. Pelz: *Normalization of Place/Transition-Systems Preserves Net Behaviour*, in: Reseaux et logique. L'etude des semantique de la concurrence dans le cadre des reseaux de Petri (These d'etat), Universite de Paris-Sud, 1990 (also in: RAIRO Informatique Theorique 26, no.1, 1992, 19–44).

[40] J.L. Peterson: *Petri Net Theory and the Modelling of Systems*, Prentice-Hall, 1981.

[41] W. Reisig: *Petri Nets. An Introduction*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1985.

[42] W. Reisig: *Elements of Distributed Algorithms. Modeling and Analysis with Petri Nets*, Springer-Verlag, 1998.

[43] G. Rozenberg, R. Verraedt: *Restricting the In-Out Structure of Graphs of Petri Nets. A Language Theoretic Point of View*, Fundamenta Informaticae VII.2, 1984, 151–189.

[44] W. Sadiq, M.E. Orlowska. *Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models*, Proceedings of the 11th International Conference on Advanced Information Systems Engineering CAiSE'99, Lecture Notes in Computer Science 1626, 1999, 195–209.

[45] W. Sadiq, M.E. Orlowska. *Analyzing Process Models using Graph Reduction Techniques*, Information Systems 25(2), 2000, 117–134.

[46] S. Sadiq, M.E. Orlowska, W. Sadiq, C. Foulger. *Data Flow and Validation in Workflow Modelling*, Proceedings of the International Conference in Research and Practice in Information Technology, Dunedin (New Zealand), 2003.

[47] I. Suzuki, T. Murata: *A Method for Stepwise Refinement and Abstraction of Petri Nets*, Journal of Computer and System Science 27, 1983, 51–76.

[48] F.L. Ţiplea: *Contributions to the Language Theory of Petri Nets*, Ph.D. Thesis, "Al.I.Cuza" University of Iasi (Romania), 1993.

[49] F.L. Ţiplea, T. Jucan: *Jumping Petri Nets*, Foundations of Computing and Decision Sciences 19, 1994, 319–332.

[50] F.L. Ţiplea, C. Ene: *Hierarchies of Petri Net Languages and a Super-Normal Form*, in: Proc. of the 2nd International Conference "Developments in Languages Theory", Magdeburg (Germany), 1995.

[51] F.L. Ţiplea, M. Katsura, M. Ito: *On Replacement of Petri Nets and Some Applications*, in: Proc. of the Workshop on Semigroups, Formal Languages and Computer Systems, RIMS Kokyuroku 960, Kyoto (Japan), 1996, 178–180.

[52] F.L. Ţiplea, M. Katsura, M. Ito: *On a Normal Form of Petri Nets*, Acta Cybernetica 12, 1996, 295–308.

[53] F.L. Ţiplea, E. Mäkinen: *Jumping Petri Nets. Specific Properties*, Fundamenta Informaticae 32, 1997, 373–392.

[54] F.L. Ţiplea, A. Ţiplea: *On Normalization of Petri Nets*, in: Proc. of the 11th Romanian Symposium on Computer Science ROSYCS'98, Iasi (Romania), 1998.

[55] F.L. Ţiplea, J. Desel: *Petri Net Process Decomposition with Application to Validation*, in: Proc of the 6th Workshop "Algorithmen und Werkzeuge für Petrinetze", Frankfurt am Main (Germany), Oct 11–12, 1999.

[56] F.L. Ţiplea, A. Ţiplea: *Petri Net Reactive Modules*, Technical Report 1999-7, Institut für Informatik, Universität Augsburg, Augsburg (Germany), Dec 1999, 50 pages.

[57] F.L. Ţiplea, A. Ţiplea: *A Simulation Preorder for Abstraction of Reactive Systems*, Third International Workshop VMCAI 2002, Venice (Italy), Lecture Notes in Computer Science 2294, 2002, 272-288.

[58] F.L. Ţiplea, A. Ţiplea: *A Compositional Semantics for Petri Nets Reactive Modules*, Proc. of NATO Advanced Research Workshop "Concurrent Information Processing and Computing", Sinaia (Romania), 2004, IOS Press (to appear).

[59] F.L. Ţiplea, D.C. Marinescu, C. Lin: *Model Checking and Abstraction for Workflow Net Verification*, Proc. of the first International Workshop on Petri Nets and Coordination PNC04, Bologna (Italy), June 21, 2004.

[60] F.L. Ţiplea, D.C. Marinescu: *Home Markings and Generalized Sound Workflow Nets*, Proc. of the 12th International Conference on Cooperative Information Systems CoopIS 2004, Larnaca (Cyprus), Oct 25-29, 2004.

[61] F.L. Ţiplea, D.C. Marinescu: *Reflections on Soundness Properties for Workflow Nets*, Festschrift volume in honor of Ken Secvcik's 60th Birthday, 2004 (to appear).

[62] F.L. Ţiplea, A. Ţiplea: *Petri Net Reactive Modules*, Theoretical Computer Science, 2004 (to appear).

[63] R. Valette: *Analysis of Petri Nets by Stepwise Refinement*, Journal of Computer and System Science 18, 1979, 35–46.

[64] A. Valmari: *Compositional Analysis with Place-Bordered Subnets*, in Proc. of the 15th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science 815, Springer-Verlag, 1994, 531–547.

[65] W. Vogler: *Behaviour Preserving Refinements of Petri Nets*, in: Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science 246, Springer-Verlag, 1987, 82–93.

[66] W. Vogler: *Modular Construction and Partial Order Semantics of Petri Nets*, Lecture Notes in Computer Science 625, Springer-Verlag, 1992.

[67] W. Vogler: *Efficiency of Asynchronous Systems, Read Arcs, and the MUTEX-Problem*, in: Proc. of ICALP'97 (P. Degano, R. Gorrieri, A. Marchetti-Spaccamela, eds.), Lecture Notes in Computer Science 1256, Springer-Verlag, 1997, 538–548 (full version as Technical Report 352, Institut für Informatik, Universität Augsburg, 1996).

[68] M. Voorhoeve. *Compositional Modeling and Verification of Workflow Processes*, In "Business Process Managements - Models, Techniques and Empirical Studies", W.M.P. van der Aalst, J. Desel, A. Oberweis (Eds.), Lecture Notes in Computer Science 1806, 2000, 184–200.

[69] M. Wolfand, U. Reimer (Eds.). *Proceedings of the International Conference on Practical Aspects of Knowledge Management*, Basel (Switzerland), 1996.

[70] Workflow Management Coalition: *Workflow Management Coalition Terminology and Glossary*, Brussels, WFMC-TC-1011, 1996.