Petri Net Reactive Modules

Ferucio Laurentiu Ţiplea **Aurora Ţiplea**

Report 1999-7 Dezember 1999

Petri Net Reactive Modules

Ferucio Laurenţiu ŢIPLEA [1] **and Aurora ŢIPLEA**

Faculty of Computer Science
"Al. I. Cuza" University
6600 Iaşi, Romania
e-mail: `fltiplea@infoiasi.ro`

Abstract

In this paper we model (discrete) reactive systems that may interact with each other by *Petri net modules* which are classical Petri nets together with a distinguished subset of *interface places*. We consider then an *asynchronous composition operation* of modules and, closely related to it, a *decomposition operation*. We show that any process (concurrent execution) of a composition of two modules can be decomposed into processes of "shifted" components for which a p-composition function exists, and vice versa. Based on this result, a *compositional semantics* of modules is then defined. The concurrent execution of a module inside a system is called a *process sample/fragment* of the system w.r.t. that module. We show that, in some circumstances, all the process samples of a system can be generated by *e-modules* which abstract from some parts of the behaviour by collapsing many consecutive steps into a single one. Some applications of process decomposition to *replacement techniques* of Petri nets, in proving *correctness of Petri net structural transformations*, and in *validation of Petri net models*, are further discussed. The last section takes into consideration the *model checking problem for Petri net modules*. A *simulation preoder* on Kripke structures with Büchi fairness constraints is considered, which is shown to preserve the *delayed version* of $\forall CTL^*$ formulas. Then, the results are transferred to Petri net modules, and discussions on *step fairness constraints* are provided.

1 Introduction and Preliminaries

In spite of the impressive progress in the development of methods for system design and verification, many realistic systems are still too large to be handled. Thus, it is important to find techniques that can be used in conjunction with the symbolic 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. In such systems it is essential to study and analyse each process as a *reactive system* (which is a collection of variables that, over time, change their values in a sequence of rounds). That is because, from the point of view of each process, the rest of the system can be viewed as an environment that continuously interacts with the process. Then, an obvious strategy is to derive properties

[1]This work was carried out while the first author was visiting Katholische Universität Eichstätt by a grant from *DAAD* (Germany), and Universität Augsburg by a grant from *Deutschen Akademien der Wissenschaften* (Germany).

(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.

For details and significant work in this direction the reader is referred, for example, to [38], [9], [15], [22], [41], [14], [39], [1], [2], [17], [20].

In this paper we model (discrete) reactive systems that may interact with each other by *Petri nets modules* which are classical Petri nets together with a distinguished subset of places (modelling the set of *interface/shared variables*). We consider then an *asynchronous composition operation* of modules that allows us to build systems from components and, closely related to it, a *decomposition operation* of systems in smaller parts (components). We show that every process (concurrent execution) of a system which is a composition of two modules can be decomposed into processes of its "shifted" components and, moreover, these processes can be related each other by means of a *p-composition function.* Conversely, composition of processes of shifted components, which are related by some p-composition function, are processes of the system. These two results lead naturally to a compositional semantics of modules, which is then defined. The concurrent execution of a module inside a system is called a *process sample/fragment* of the system w.r.t. that module. We show that, in some circumstancies, all the process samples of a system can be generated by e-modules which abstracts from some parts of the behaviour by collapsing many steps into a single one. It is shown further that, in some cases, e-modules are process equivalent to modules (the equivalence notion is suitable chosen but less restrictive). Then we point out some applications of process decomposition. First we take into consideration the replacement operation. One of the main problem is to find two equivalence relations $\approx_1$ and $\approx_2$ such that from $\gamma_1 \approx_1 \gamma_2$ one can infer $\gamma \approx_2 \gamma[\gamma_1 \leftarrow \gamma_2]$, where γ_1 is a subnet of γ and $\gamma[\gamma_1 \leftarrow \gamma_2]$ denotes the result of the replacement of γ_1 by γ_2 (inside γ). We give two pairs of such equivalences, $(\approx_{mP}, \approx_P)$ and $(\approx_{mPW}, \approx_{PW})$, and some properties of them are proved. The replacement operation we consider suggests a proof technology to be used reasoning about Petri net transformations. We exemplify it by giving shorter and elegant proofs of correctness to some transformations known from literature (compare, for instance, the proof of Theorem 4.2.1 with the proof of a similar theorem in [24]). The last application proposes a methodology for validation of Petri net models. Finally, we take into consideration the model checking problem for Petri net modules. For generality, the results are developed first on fair Kripke structures. Thus, we introduce a simulation preorder which preserves some versions of $\forall CTL^*$ formulas and offers abstraction facilities. Then we consider the asynchronous parallel composition of such structures and we show that every composed system is smaller in the simulation preorder than its "augmented" individual components. The connection to Petri net modules is then made by associating to each module a (fair) Kripke structure. The paper ends with a detailed discussion on related work, and references.

In the remainder of this section we recall the basic definitions and notations in Petri net theory (for further details the reader is referred to [5], [10], [25], [26]).

The empty set is denoted by $\emptyset$, and $|A|$ denotes the cardinality of the finite set A. $A \subseteq B$ denotes the inclusion of the set A into the set B, and $\mathcal{P}(A)$ is the set of all subsets (the powerset) of A. The set of integers is $\mathbf{Z}$, and the set of nonnegative integers is $\mathbf{N}$. For a binary relation R, $Dom(R)$ and $Cod(R)$ denote the domain and the codomain, respectively, of R, $R(x)$ is the image of x under R, R^+ (R^*) is the transitive (reflexive and transitive) closure of R.

If $f_i : A_i \to \mathbf{Z}$ are functions, $i = 1, 2$, $f_1 + f_2$ is the function from $A_1 \cup A_2$ into $\mathbf{Z}$ given by $(f_1 + f_2)(a) = f_1(a)$ for all $a \in A_1 - A_2$, $(f_1 + f_2)(a) = f_2(a)$ for all $a \in A_2 - A_1$, and $(f_1 + f_2)(a) = f_1(a) + f_2(a)$ for all $a \in A_1 \cap A_2$. In a similar way is defined $f_1 - f_2$. The restriction of a function $f : A \to B$ to the set $C \subseteq A$ is denoted by $f|_C$; f^{-1} is the function from B into the powerset of A given by $f^{-1}(b) = \{a \in A | f(a) = b\}$ for all $b \in B$.

A (finite) *Petri net* (or *net*, for short) is a 4-tuple $\Sigma = (S, T, F, W)$, where S and T are two finite sets (of *places* and *transitions*, respectively), $S \cap T = \emptyset$, $F \subseteq (S \times T) \cup (T \times S)$ is the *flow relation*, and $W : (S \times T) \cup (T \times S) \to \mathbf{N}$ is the *weight function* of Σ verifying $W(x, y) = 0$ iff $(x, y) \notin F$. In our paper we shall suppose that all the nets we consider do not have isolated transitions (but they may have isolated places). When $W(x, y) \leq 1$ for all $(x, y) \in F$, we may (and will) simplify the 4-tuples (S, T, F, W) to the 3-tuple (S, T, F).

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 operations and relations on vectors are componentwise defined. For $x \in S \cup T$ we set

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

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

A *marked net* is a pair $\gamma = (\Sigma, M_0)$, where Σ is a net and M_0, the *initial marking* of γ, is a marking of Σ. A *labelled marked net* is a 3-tuple $\gamma = (\Sigma, M_0, l)$, where the first two components form a marked net and l, the *labelling function* of γ, assigns to each transition either a letter or the empty word λ. 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 definitions above Σ is called the *underlying net* of γ. A marking (place, transition, arc, weight) of a net γ is any marking (place, transition, arc, weight) of the underlying net of γ.

Pictorially, a net γ is represented by a graph. Then the places are denoted by circles and transitions by boxes; the flow relation is represented by arcs. The arc $f \in F$ is labelled 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 labelling function is denoted by placing letters into the boxes representing transitions (when some boxes are empty we will understand that the corresponding transitions are labelled by themselves).

Let γ be a net and M a marking of it. The *transition 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 transition $w \in T^*$ in the usual way. If $M_0[w\rangle_\gamma M$ then M is called *reachable*; $[M_0\rangle_\gamma$, called the *reachability set* of γ, denotes the set of all reachable markings of γ. When the components of all reachable markings do not exceed some natural number $n \geq 1$, the net is called *n-safe* (or *safe*, in general). For safe nets, the reachability set is finite. The notation "$[\cdot\rangle_\gamma$" will be simplified to "$[\cdot\rangle$" whenever γ is clear from context.

The concurrent behaviour of Petri nets is well-expressed by the notion of a *process*. Generally speaking, processes of Petri nets are obtained by running the 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 *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 to the relation $\prec_N$. As we will only use B-cuts we call them shortly *cuts* (see [5] for more details). The *initial* (*final*) *cut* of N is ${}^\circ N = \{b \in B | |^\bullet b| = 0\}$ ($N^\circ = \{b \in B | |b^\bullet| = 0\}$). A *path in N from x to y* is any finite sequence of elements $x = x_1, \ldots, x_n = y$ such that for all $1 \leq i < n$, $(x_i, x_{i+1}) \in R$. In defining processes we need *V-labelled occurrence nets* which are couples $\pi = (N, p)$, where N is a 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 labelled 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 $(S \cup T)$-labelled 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 ([5]). We adopt here the axiomatic definition (the inductive one will be given in Section 3.2 as a particular case of the inductive definition of processes of jumping nets). A *process* of $\gamma = (\Sigma, M_0)$ is any $(S \cup T)$-labelled occurrence net $\pi = (N, p)$ satisfying:

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

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

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

Processes of labelled nets $\gamma = (\Sigma, M_0, l)$ are obtained from processes $\pi = (N, p')$ of (Σ, M_0) by replacing the function p' by p, where $p(x) = p'(x)$ for all condition x, and $p(x) = l(p'(x))$ for all event x. That is, the events are labelled by $l \circ p'$. From this reason we will use sometimes $l \circ p'$ instead p (with the meaning above). The set of all processes of a net γ is denoted by $\Pi(\gamma)$. A *path* in a process π is any path in its underlying occurrence net.

2 Petri Net Modules and their Asynchronous Composition

As we have said in the first section, we model discrete reactive systems that may interact with each other by Petri net modules which are defined as follows.

Definition 2.1 *A* Petri net module (PN-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 ([1]), or to the distinction between unobservable owned variables and observable variables in the formalism of fair Kripke structures as given in ([17]) [2].

The environment interacts with a module $\mathcal{M}$ by updating, from time to time, the content of the interface places. Such an interaction can be mathematically modelled 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 the module $\mathcal{M}$ point of view this updating is done in exactly one step.

Definition 2.2 *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 mainly used to describe in a compact way the behaviour of modules; they will abstract from some parts of the behaviour of modules by collapsing many consecutive steps into a single one (see Section 3.2).

Definition 2.3 *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*

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

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

It is important 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 [3].

We define now the *asynchronous parallel composition* of modules. Generally, 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

[2]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.

[3]The approach we 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 something like semaphor variables) could be an worthy idea.

asynchronous, and for each of them different variants are known ([2], [17]). 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 [9], [41], [39], [33], [42], [19] (see also the discussion at the end of the paper).

In order to avoid some annoying and totally unessential things for our purposes we assume given two disjoint countable sets $\mathcal{S}$ and $\mathcal{T}$, 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 2.4 *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 $\Sigma = (S, T, F, W)$;*
- *S, T, F, W, M_0 and l are the union of the sets of places, transitions, flow relations, weight functions, markings and labelling functions of $\mathcal{M}_0$ and $\mathcal{M}_1$, respectively.*

We note that the unions of functions in Definition 2.4 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; correspondingly, an e-module $\mathcal{J} = (\mathcal{M}, R)$ will be written as $\mathcal{J} = (\gamma, R)$. 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*.

Let $ePN(S^c, M_0^c)$ be the set of all e-modules whose underlying modules are in $PN(S^c, M_0^c)$. Two e-modules in this set are called *compatible* when their underlying modules are compatible. The asynchronous parallel composition can be extended to compatible e-modules in $ePN(S^c, M_0^c)$ by

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

The asynchronous parallel composition of (e-)modules in $(e)PN(S^c, M_0^c)$ 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 (e-)modules x_0, x_1 and x_2. Moreover, the modul $\gamma_\emptyset = ((S^c, \emptyset, \emptyset, \emptyset), M_0^c, \emptyset)$, or $(\gamma_\emptyset, \emptyset)$ in case of e-modules, is the unit of this operation [4].

[4]One may consider the equivalence relation $\equiv$ on $PN(S^c, M_0^c)$ induced by isomorphisms of labelled 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 that classes. Then, this operation is totally defined on the quotient set $PN(S^c, M_0^c)/\equiv$ and structures it as a monoid.

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 some 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 unique and, moreover, if we want to have some properties of components, it is not all the time possible. 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 lead to the original net γ. However, the decomposition based on subnets generated by subsets of transitions is indeed

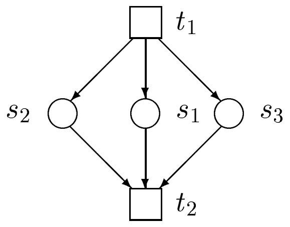
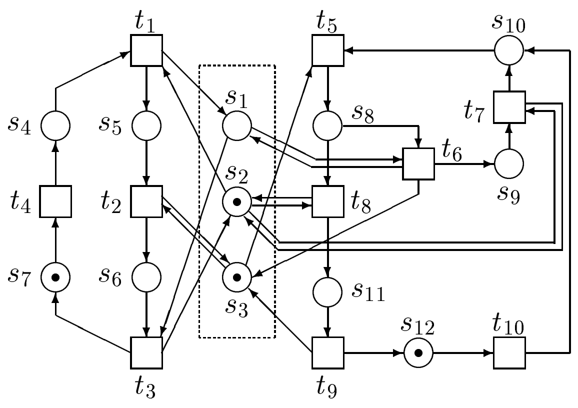


Figure 2.1

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 $S_1 = {}^\bullet T_1^\bullet$, 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 (labelled) 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 an example of decomposition which will be used intensively in the next section in order to exemplify process decomposition and process sample generation (another example discussed in detail may be found in [37]). The net 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 dashbox 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 [27]).

That two sites of the net in Figure 2.2 are connected each other by means of the places s_1, s_2 and s_3. We may separate them into two nets γ_0 and γ_1 by multiplying twice these places togheter 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.



s_1 = writer involved
s_2 = writer detached
s_3 = reader detached
s_4 = prep1
s_5 = prep2
s_6 = writing
s_7 = producing
s_8 = pend2
s_9 = failed
s_{10} = pend1
s_{11} = reading
s_{12} = using

Figure 2.2

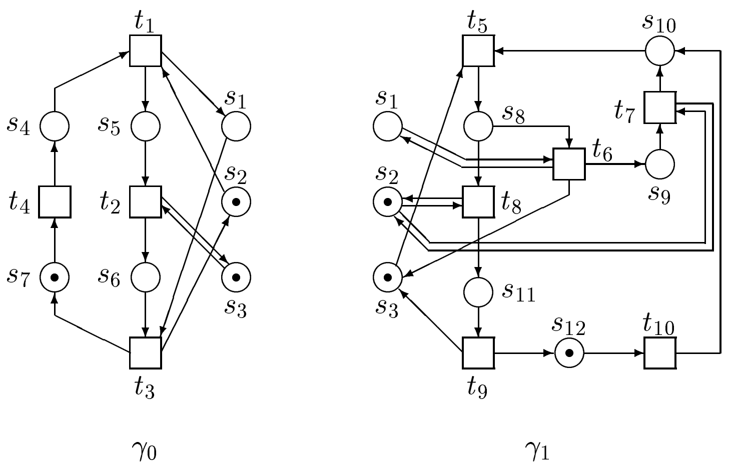


Figure 2.3

3 Process Decomposition w.r.t. Asynchronous Composition

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

1. What is the structure of processes of $\mathcal{M}_0 \circ \mathcal{M}_1$? Can we decompose them into processes of $\mathcal{M}_0$ and $\mathcal{M}_1$? Can we get them by composing arbitrary processes of $\mathcal{M}_0$ and $\mathcal{M}_1$?

2. Can we generate "fragments" of processes of $\mathcal{M}_0 \circ \mathcal{M}_1$ corresponding to $\mathcal{M}_0$ or $\mathcal{M}_1$?

We will give to the first question a complete, and to the second one a partial, answer.

3.1 Process Decomposition

Let us consider the process π pictorially represented in Figure 3.1.1, of the net γ from Figure 2.2. This process can be split into two parts (occurrence nets) π_0 and π_1 as in Figure 3.1.2, according to the decomposition of γ (Figure 2.3). The initial cut of π_0 generates a marking

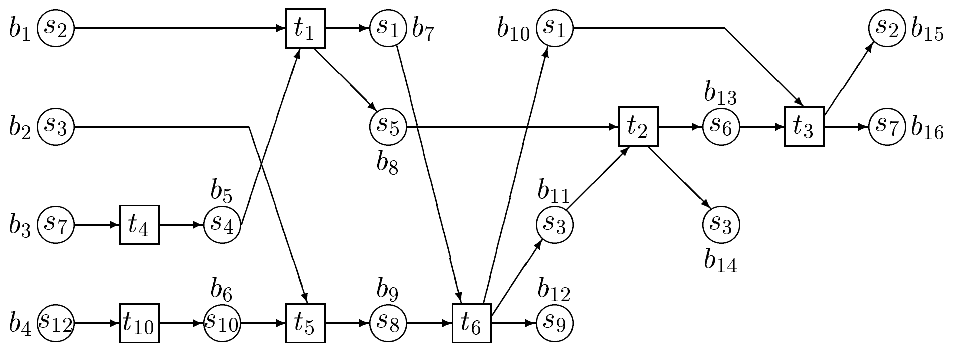


Figure 3.1.1

of γ_0 with one token in s_1 and two tokens in s_3, which is neither reachable in γ nor in γ_0; similarly, the initial cut of π_1 generates a marking of γ_1 with one token in each of the places s_1, s_2 and s_3, 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 s_1 and one in 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 labelled occurrence nets are particular nets whose places are labelled as well. Therefore, we can extend the definition of composition along a set S^c to the case of these nets by requiring supplementary $p_1(s) = p_2(s)$ for all $s \in S^c$ (p_1 and p_2 are the corresponding labelling functions). Then, we adopt one more notation. For a Petri 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 Petri 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 $M \in \mathbf{N}^{S^c}$ we have:

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

Theorem 3.1.1 (Process decomposition theorem)
Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets. For each process $\pi \in \Pi(\gamma_0 \circ \gamma_1)$ 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 two compatible nets and $\pi = (N, l \circ p')$ be a process of $\gamma_0 \circ \gamma_1$, where $N = (B, E, F')$ (see the definition of processes 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 labelled by places in S^c. This set can be partitioned into the following subsets (not necessary each of them non-empty):

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

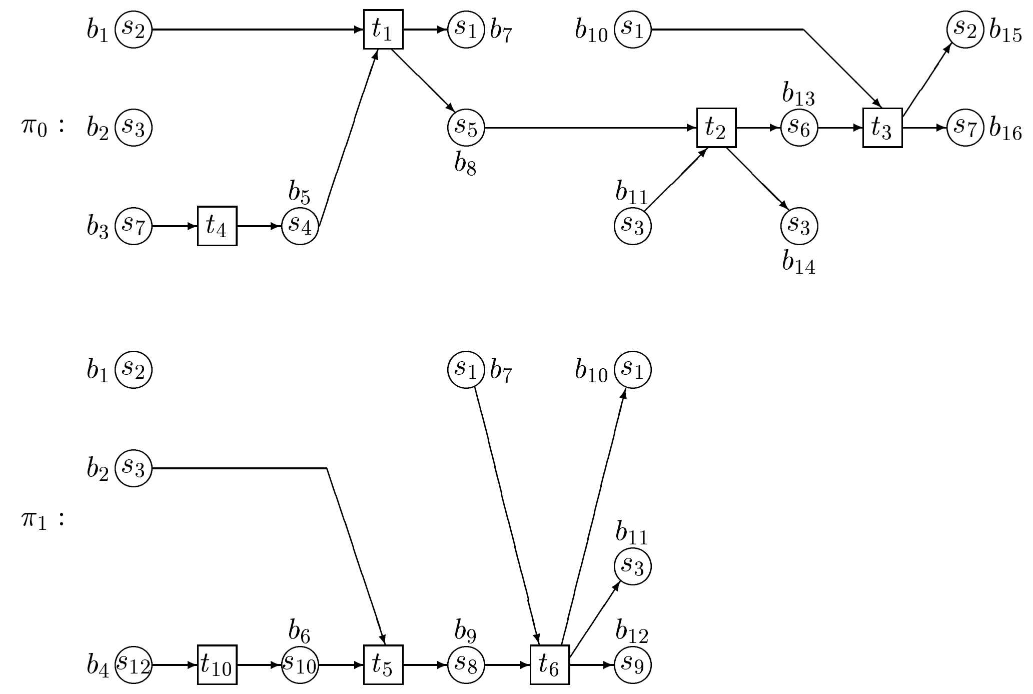


Figure 3.1.2

In Figure 3.1.3(a) are picturially represented conditions from each of the sets defined above (in this order).

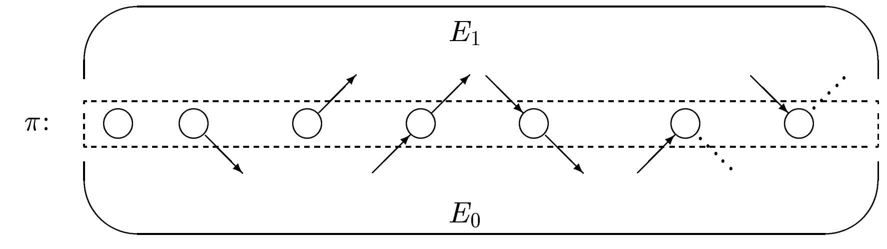
Let π_0 be the subnet of π generated by E_0 togheter with the set of conditions $C_{in} \cup C_{in,1}$, and let π_1 be the subnet of π generated by E_1 togheter with the set of conditions $C_{in} \cup C_{in,0}$ (Figure 3.1.3(b)). Considering the markings $M' = M_{C_{1,0}}$ and $M'' = M_{C_{0,1}}$, one can easily verify that $\pi_0 \in \Pi(\gamma_0 + M')$, $\pi_1 \in \Pi(\gamma_1 + M'')$, and $\pi = \pi_0 \circ \pi_1$ (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). $\Box$

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). The set of all process samples of γ w.r.t. γ_0 will be denoted by $\Pi(\gamma, \gamma_0)$.

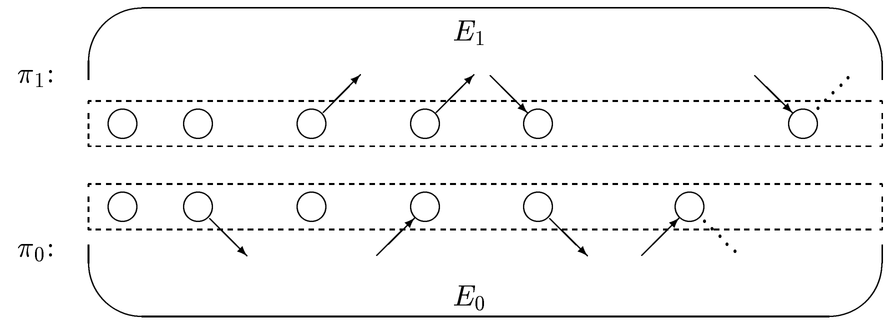
For the process π in Figure 3.1.1 we have (with the same notation as in the proof of Theorem 3.1.1):

$$\begin{array}{lcl} C & = & \{b_1, b_2, b_7, b_{10}, b_{11}, b_{14}, b_{15}\} \\ C_{in} & = & \emptyset \\ C_{in,0} & = & \{b_1\} \\ C_{in,1} & = & \{b_3\} \\ C_{0,1} & = & \{b_7\} \\ C_{1,0} & = & \{b_{10}, b_{11}\} \\ C_0 & = & \{b_{14}, b_{15}\} \\ C_1 & = & \emptyset. \end{array}$$

π_0 and π_1 in Figure 3.1.2 are processes of $(\gamma_0 + (1,0,1))$ and $(\gamma_1 + (1,0,0))$, respectively



(a)



(b)

Figure 3.1.3

(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.

We will consider now the converse of Theorem 3.1.1. 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 labelled by places in S^c. A *p-partition* of $C(\pi)$ is a couple of sets $(A, C(\pi) - A)$ such that:

(i) $A \subseteq C(\pi)$;

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

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 labelled-preserving bijection $f : A_0 \to A_1$ 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*)
$(\not\exists b_0, b_1, b_0', b_1')(f(b_0) = b_1 \;\wedge\; f(b_0') = b_1' \;\wedge\; b_1 \prec_{\pi_1} b_1' \;\wedge\; b_0' \prec_{\pi_0} b_0)$.

It is easy to see that for the processes $\pi_0 \in \Pi(\gamma_0 + M')$ and $\pi_1 \in \Pi(\gamma_1 + M'')$ in the proof of Theorem 3.1.1, 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) $M' = M_{\{b \in Cod(f) \mid |^\bullet b| = 1\}}$ and $M'' = M_{\{b \in Dom(f) \mid |^\bullet b| = 1\}}$, 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 3.1.2 (Process composition theorem)
Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ *be two compatible nets,* $M', M'' \in \mathbf{N}^{S^c}$, $\pi_0 \in \Pi(\gamma_0 + M')$ *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:*

(1) π_1' *is obtained from* π_1 *by renaming its elements (but not the labels)* [5];

(2) $\pi_0 \circ \pi_1' \in \Pi(\gamma_0 \circ \gamma_1)$ *(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 by b_0, where $f(b_0) = b_1$.

Let π_1' be the process such obtained. From the axiomatic definition of processes it follows easily that $\pi_0 \circ \pi_1' \in \Pi(\gamma_0 \circ \gamma_1)$, where the composition of process is along A_0. $\Box$

Theorems 3.1.1 and 3.1.2 give a complete answer to the first question in Section 3. Moreover, they lead to a compositional semantics of modules, as follows. For a net $\gamma \in PN(S^c, M_0^c)$ define

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

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 $\pi_1 \in \Pi(\gamma_1 + M'')$, $\pi_0 \circ_{M_0^c} \pi_1$ is the set 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.

Corollary 3.1.1 *Let* $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ *be two compatible nets. Then,*

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

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

(i) $p_1(x) = p_1'(\varphi(x))$, for all $x \in B_1 \cup E_1$;

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

(p_1 and p_1' are the labelling 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 4.1 we will introduce a more general concept of isomorphism – see also the concept of a (j, λ)-*isomorphism* in the next section.

Proof Directly from definitions, composition and decomposition theorems. □

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 $\pi_1 \in \Pi(\gamma_1 + M'')$, $\pi_0 \circ_{\geq M_0^c} \pi_1$ is the set 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''. Extend then this operation, by union, to sets of processes.

Corollary 3.1.2 *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 each marking M on S^c we have:

$$\begin{array}{rcll}
\Pi((\gamma_0 \circ \gamma_1) + M) & = & \Pi((\gamma_0 + M) \circ (\gamma_1 + M)) & \\
 & = & \Pi_m(\gamma_0 + M) \circ_{M_0^c + M} \Pi_m(\gamma_1 + M) & \text{(Corollary 3.1.1)} \\
 & \subseteq & \Pi_m(\gamma_0) \circ_{\geq M_0^c} \Pi_m(\gamma_1) & \text{(definition of } \Pi_m \text{ and } \circ_{\geq M_0^c}\text{)}
\end{array}$$

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

3.2 Process Sample Generation

Given a net $\gamma = \gamma_0 \circ \gamma_1$, where $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ are compatible nets, we may ask how to generate all the process samples of γ w.r.t. γ_0. Of course, we may generate all the processes of γ and then split these processes as in the proof of Theorem 3.1.1. We shall describe in the sequel an alternative method based on e-modules which allows a direct generation of process samples. First, let us look again to the nets in Figure 3.1.2. From the 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 with the condition b_7 (labelled by s_1). More precisely, γ_0 gives to γ_1 a condition labelled by s_1 and receives two conditions labelled 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 produced configuration is the same with the initial one, but this is not the case in general).

Considering the set R_0 of all such pairs, we obtain an e-module (γ_0, R_0) whose behaviour captures the "interactive" behaviour of γ_0 with γ_1, making abstraction of the internal behaviour of γ_1. We shall prove that the set of all process samples of γ w.r.t. γ_0 can be easily obtained from the set of processes of (γ_0, R_0), which is to be defined. First of all we recall the concept of a *jumping (Petri) net* in a slightly different way than in [31] and [35]; this concept is a natural generalization of the concept of an e-module by allowing more sets of interface places.

Definition 3.2.1 *A* jumping net (marked jumping net, labelled marked jumping net) *is a couple $\mathcal{J} = (\gamma, R)$, where γ is a net (marked net, labelled marked net) and R is a finite union of sets, each of which being a binary relation on $\mathbf{N}^{S'}$ for some $S' \subseteq S$.*

The elements of R are called *jumps* of $\mathcal{J}$. A jump (M, M'), where $M, M' \in \mathbf{N}^{S'}$ and $S' \subseteq S$, is called *local on* S'. For 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 markings $M, M' \in \mathbf{N}^S$, where $S' \subseteq S$ and $(M|_{S'}, M'|_{S'})$ is a local jump on S'.

A computation step in a jumping net $\mathcal{J} = (\gamma, R)$ is performed either by a transition, in the usual way, or by a jump. That is,

$$M[x\rangle M' \quad \Leftrightarrow \quad \text{either } x \in T \text{ and } M[x\rangle_\gamma M', \text{ or } x \in R \text{ and } M\, R\, M'.$$

In the case $R \subseteq \mathbf{N}^{S'} \times \mathbf{N}^{S'}$ for some $S' \subseteq S$, labelled marked jumping nets correspond exactly to e-modules (with the set S' of interface places).

Local jumps which do not affect all places of a net can occur concurrently with each other or concurrently with transition occurrences. This is formally reflected in the following definition of processes of jumping nets (for convenience we will adopt an inductive definition). Let $\mathcal{J} = (\gamma, R)$ be a marked jumping Petri net, t a transition and $r = (M, M')$ a local jump on a subset $S' \subseteq S$. Then:

- an *elementary occurrence net* associated to t is a labelled occurrence net $\pi = (N, p)$ with the properties: π contains only one event e labelled by t, $W(s, t)$ preconditions and $W(t, s)$ postconditions of e labelled by s, for all $s \in S$, and no other element;
- an *elementary occurrence net* associated to r is a labelled occurrence net $\pi = (N, p)$ with the properties: π contains only one event e labelled by r, $M(s)$ preconditions and $M'(s)$ postconditions of e labelled 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 an occurrence net (N, p) which does not contain any event and, for each $s \in S$, it contains exactly $M_0(s)$ conditions labelled by s.

Definition 3.2.2 *Let $\mathcal{J} = (\gamma, R)$ 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 possible, 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 a subset S' of places 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 possible, is in $\Pi(\mathcal{J})$.

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

It is clear that for every $\pi \in \Pi(\mathcal{J})$ there is at least 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 3.2.2, for all $0 \leq i \leq m-1$. Processes of labelled jumping nets are obtained as for labelled nets, by relabelling 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. That is, $\Pi(\gamma) \subseteq \Pi(\mathcal{J})$, for every jumping net $\mathcal{J} = (\gamma, R)$. Moreover, the Definition 3.2.2 leads also to the inductive definition of processes of nets ([5]).

Example 3.2.1 *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 2.2, and R_0 and R_1 are relations on $\mathbf{N}^{S^c}$ containing r_0 and $r_1 = ((0,1,1),(1,0,1))$, respectively ($S^c = \{s_1, s_2, s_3\}$ and r_0 is the pair given at the beginning of this section). Then, π_0 (π_1) in Figure 3.2.1 is a process of $\mathcal{J}_0$ ($\mathcal{J}_1$).*

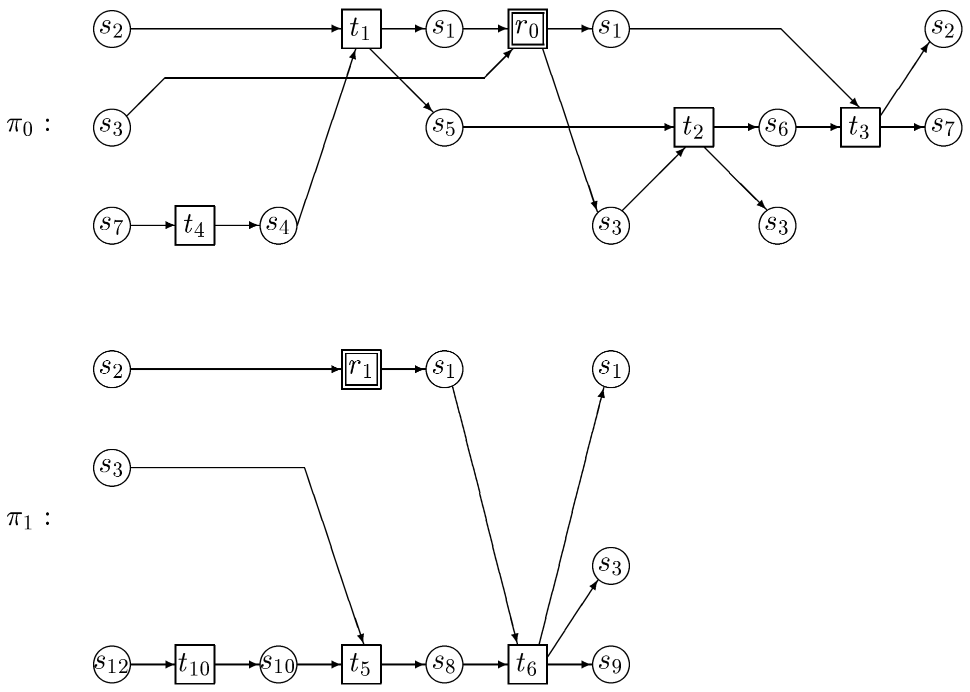


Figure 3.2.1

Now let us turn our attention to the generation of process samples by jumping nets. First let us note that a subnet γ_1 of a net γ may *induce* some jumps for the net $\gamma - \gamma_1$. Moreover, it may induce the same jump at different markings of γ. If the jumps induced by γ_1 do not depend on the internal configuration of γ_1, but just on the marking on the interface places, then γ is called *γ_1-context free.*

Definition 3.2.3 *Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets, $\gamma = \gamma_0 \circ \gamma_1$, and let M be a reachable marking in γ.*

(1) A jump $(M^c, \overline{M}^c)$ *on* S^c *is* induced by γ_1 at M in γ *if there is a marking* $\overline{M} \in [M_0\rangle_\gamma$ *such that:*

(i) $M|_{S^c} = M^c$ *and* $\overline{M}|_{S^c} = \overline{M}^c$*;*

(ii) $\overline{M}$ *is reachable from* M *only by occurrences of transitions in* γ_1*, but at least by one occurrence (and so* $\overline{M}|_{S_0^i} = M|_{S_0^i}$*).*

(2) γ *is called* γ_1-context free *if for every jump* $(M^c, \overline{M}^c)$ *induced by* γ_1 *and for every reachable marking* M *in* γ*, if* $M|_{S^c} = M^c$ *then* γ_1 *can induce* $(M^c, \overline{M}^c)$ *at* M*.*

It can be easily verified that the net γ in Figure 2.2 is both γ_0- and γ_1-context free, γ_0 and γ_1 being the nets in Figure 2.3.

Let $\mathcal{J} = (\gamma, R)$ be a jumping net. For each process π of $\mathcal{J}$ we define a new occurrence net by removing all the events labelled by jumps (and the corresponding arcs). Let $\Pi'(\mathcal{J})$ be the set of all these occurrence nets. The occurrence nets in Figure 3.1.2 are obtained, as described above, from the processes in Figure 3.2.1. We note that for every $\mathcal{J} = (\gamma, R)$, $\Pi(\gamma) \subseteq \Pi'(\mathcal{J})$.

Theorem 3.2.1 (Process sample generation theorem)
Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ *be two compatible nets, and* $\gamma = \gamma_0 \circ \gamma_1$*. If* γ *is* γ_1*-context free then* $\Pi(\gamma, \gamma_0) = \Pi'(\mathcal{J})$*, where* $\mathcal{J} = (\gamma_0, R_0)$ *and* R_0 *is the set of jumps induced by* γ_1 *in* γ*.*

Proof Let π be a process of γ and $\pi_1, \ldots, \pi_n = \pi$ be an inductive definition of it. Define a sequence of occurrence nets $\pi'_1, \ldots, \pi'_n = \pi'$, as follows:

- $\pi'_1 = \pi_1$;
- assume $\pi'_1, \ldots, \pi'_i$ have been defined, for $i < n$. If π_{i+1} is obtained by extending π_i to the right by an event without preconditions and postconditions labelled by elements in S^c, then extend π'_i by the same event and in the same way; let π'_{i+1} be the result.

  If π_{i+1} is obtained by extending π_i to the right by an event with preconditions or postconditions labelled by elements in S^c, then extend π'_i by the same event and in the same way, and relabel this event by the jump $(M^c, \overline{M}^c)$, where M^c is the marking on S^c induced by $(\pi'_i)^\circ$ and $\overline{M}^c$ is the marking on S^c produced by the occurrence of this event; let π'_{i+1} be the result.

As we can see, π'_i is obtained from π_i by relabelling all the events associated to transitions in γ_1 which induce jumps on S^c, for all i.

The process π can be decomposed along the set B^c of conditions as in Theorem 3.1.2. Let $\pi = \alpha \circ \beta$. We have $\alpha \in \Pi(\gamma, \gamma_0)$. Similarly, we can decompose π' along B^c into α' and β', but with the difference that all the events labelled by jumps that are connected to some conditions in B^c, and the corresponding connecting arcs, are also kept in α' (the other conditions connected to such events, and the corresponding arcs, are removed). Assume that α' corresponds to γ_0, in the sense above. Then, if we remove from α' the events labelled by jumps and the corresponding arcs, we get α. Moreover, $\alpha' \in \Pi(\mathcal{J})$, which shows that $\alpha \in \Pi'(\mathcal{J})$.

Conversely, let π be a process of $\mathcal{J}$. There is an inductive definition of π,

$$\pi_1, \ldots, \pi_n = \pi,$$

such that, for each i, the occurrence net π'_i obtained from π_i by removing all the events labelled by jumps (and the corresponding arcs) is an element of $\Pi'(\mathcal{J})$. Define now a sequence of processes of γ,

$$\overline{\pi}_1, \ldots, \overline{\pi}_n,$$

such that, for each i, $\overline{\pi}_i = \pi'_i \circ \pi''_i$ for some π''_i, and show that π'_i is a process samples of γ w.r.t. γ_0. We will obtain in this way that π' is a process sample of γ w.r.t. γ_0.

π_1 is an initial occurrence net of $\mathcal{J}$ and $\pi'_1 = \pi$. Extend π_1 to an initial occurrence net $\overline{\pi}_1$ of γ. Clearly, $\overline{\pi}_1 = \pi'_1 \circ \pi''_1$ for some π''_1, and π'_1 is a process samples of γ w.r.t. γ_0.

Assume we have defined $\overline{\pi}_1, \ldots, \overline{\pi}_i$, for $1 \leq i < n$. There are two cases to be considered.

Case 1: π_{i+1} is obtained by extending π_i to the right by an elementary occurrence net associated to a transition t in γ_0. Extend to the right the process $\overline{\pi}_i$ by the same elementary occurrence net associated to t, and in the same way; let $\overline{\pi}_{i+1}$ be the result. Then, it is easy to see that $\overline{\pi}_{i+1} = \pi'_{i+1} \circ \pi''_{i+1}$ for some π'_{i+1}, and π'_{i+1} is a process samples of γ w.r.t. γ_0.

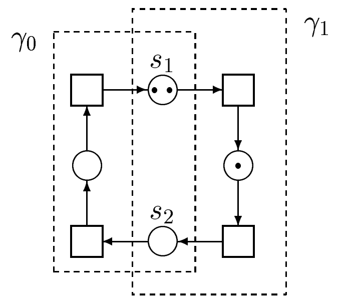
Case 2: π_{i+1} is obtained by extending π_i to the right by an elementary occurrence net associated to a jump $r = (M^c, \overline{M}^c)$. Consider the marking M_i induced by $\overline{\pi}_i^\circ$. Since γ is γ_1-context free it follows that γ_1 can induce the jump r at the marking M_i, and let $w = t_1 \cdots t_m$ be a nonempty sequence of transitions of γ_1 inducing this jump (note that for every $s \in S^c$, $M(s) = M^c(s)$). Extend recursively to the right the process $\overline{\pi}_i$ by elementary occurrence nets associated to $t_1, \ldots, t_m$ (in this order), with the next remarks:

- the elementary occurrence net associated to t_1 has the same preconditions labelled by places in S^c as the elementary occurence net associated to r;
- the elementary occurrence net associated to t_m has the same postconditions labelled by places in S^c as the elementary occurence net associated to r.

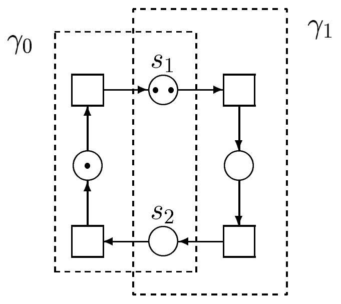
The process $\overline{\pi}_{i+1}$ obtained in this way verifies $\overline{\pi}_{i+1} = \pi'_{i+1} \circ \pi''_{i+1}$ for some π'_{i+1}, and π'_{i+1} is a process samples of γ w.r.t. γ_0. $\square$

Theorem 3.2.1 gives us a specification tool for all the process samples of a net γ w.r.t. a subnet γ_1 generated by a subset of transitions, in case that γ is γ_1-context free.

The context freeness property with respect to a subnet is a strong property. "Very simple" nets do not have this property, as the net γ is Figure 3.2.2 is. However, there are important



γ_1 induces the jump $((2,0),(2,1))$ at $(2,0)$ on $\{s_1, s_2\}$



γ_1 cannot induce the jump $((2,0),(2,1))$ at $(2,0)$ on $\{s_1, s_2\}$

Figure 3.2.2

classes of nets with this property. For example, the net version of the Owicki-Lamport's

Mutex algorithm, which we already discussed, has this property. The difference between this net and the net in Figure 3.2.2 can be expressed as follows.

Let $\gamma_0, \gamma_1 \in PN(S^c, M_0^c)$ be two compatible nets, and $\gamma = \gamma_0 \circ \gamma_1$. For a marking M^c on S^c define the set

$$\begin{aligned} min_\gamma(\gamma_1, M^c) \;=\; & \{M|_{S_1} | M \in [M_0\rangle_\gamma \;\wedge\; M|_{S^c} = M^c \;\wedge \\ & (\forall M' \in [M_0\rangle_\gamma)(M'|_{S^c} = M^c \;\Rightarrow\; \neg(M' < M))\}. \end{aligned}$$

Then, it is easily seen that γ is γ_1-context free iff whenever γ_1 can induce a jump $(M^c, \overline{M}^c)$ then for every marking $M \in min_\gamma(\gamma_1, M^c)$ there is a marking M' reachable from M in γ_1 and which agrees with $\overline{M}^c$ on S^c (in other words, the jump $(M^c, \overline{M}^c)$ can be induced at every marking in $min_\gamma(\gamma_1, M^c)$).

Now, we can see that the net γ_1 in Figure 3.2.2 induces the jump $((2,0),(2,1))$, but it can not induce this jump at $(2,0,0) \in min_\gamma(\gamma_1,(2,0))$. In contrast, the net in Figure 2.2 satisfies the criterion above (we can easily check that due to the fact that the subnets γ_0 and γ_1 have very particular internal markings).

In [31] it has been proved that jumping nets are strictly more powerful than nets from the interleaving semantics point of view, even if finite sets of jumps are considered. Clearly, this result holds true for the case of process semantics as well. In some particular cases, as we shall see below, jumping nets can be simulated by nets.

Definition 3.2.4 *Let S be a finite non-empty set and R be a finite union of sets, each of which being a binary relation on $\mathbf{N}^{S'}$ for some set $S' \subseteq S$. We say that R is* Δ-finite *if there is a finite set V of vectors with integer components such that for every $(M, M') \in R$ we have $M' - M \in V$.*

All jumping nets obtained by decomposing safe nets are Δ-finite.

Definition 3.2.5 *Let $\mathcal{J}$ be a jumping net. Its set of jumps R is called* complete *if for every reachable marking M in γ and for every jump $(M_1, M_2) \in R$, if (M_1, M_2) is local on $S' \subseteq S$ and $M|_{S'} \geq M_1$ then there is a marking $M' \in \mathbf{N}^{S'}$ such that $(M|_{S'}, M') \in R$ 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 3.2.6 *A process $\pi_1 = (N_1, p_1)$ of a labelled jumping net is* (j, λ)-isomorphic *to a process $\pi_2 = (N_2, p_2)$ of a labelled 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 (see the footnote in Section 3.1): it is an isomorphism of occurrence nets preserving all the condition-labels and all the non-jump event-labels; the events labelled by jumps are mapped into events labelled by λ (this justifies our terminology of (j, λ)-isomorphism).

Theorem 3.2.2 *Let $\mathcal{J} = (\gamma, R)$ be a labelled marked jumping net. If R is Δ-finite and complete then there is a labelled marked net γ' such that each process of $\mathcal{J}$ is (j, λ)-isomorphic to a process of γ', and vice-versa.*

Proof Since R is Δ-finite, there are finitely many vectors $V_1, \ldots, V_k$ on $\mathbf{Z}$, $k \geq 1$, as in Definition 3.2.4. Consider

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

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 $(M, M') \in R$ and $M' - M = V_i$.

To each i and $M \in min(U_i)$ we associate a transition $t_{i,M}$ such that $W'(s, t_{i,M}) = M(s)$ 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 labelled by λ.

Let $\gamma' = (\Sigma', M_0, l')$, where:

- $\Sigma' = (S, T', F', W')$;
- $T' = T \cup \{t_{i,M} | 1 \leq i \leq k, \ M \in min(U_i)\}$;
- $l'(t)$ 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: $M \, R \, M'$ 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 $M \, R \, M'$ by this jump.

Using the above facts one can easily complete the proof of the theorem. □

4 Some Applications

In this section we will present some applications of the results developed in Section 3.1. We will discuss first a technique of replacement and then some applications of it in proving the correctness of Petri net structural (modular) transformations. Finally, a methodology for validation of Petri net models, based on partially ordered runs, is proposed.

4.1 Concurrent Behaviour and Replacements

Many transformations of Petri nets may be simply described by "replace the subnet γ_1 of γ by the net γ_2"; that 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 behaviour and equivalence relations preserving refinement (as above) have been proposed ([38], [29], [23], [40], [13], [18], [3], [4], [41], [6], [33]).

In this section we consider a technique of replacement based on subnets generated by subsets of transitions. That is, only such subnets will be replaced. As a conclusion, the replacement operation can be simply expressed 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. The equivalence relations we consider are based on the concepts of *isomorphism of processes* and *partial word*.

Definition 4.1.1 *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 $\varphi : B_1 \cup E_1 \to B_2 \cup E_2$ 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 3.1): 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 an equivalence relation (transitivity is not assured, as the processes in Figure 4.1.1 shows us: $\pi_1 \cong \pi_2$, $\pi_2 \cong \pi_3$ but π_1 and π_3 are not isomorphic), but this is not important for the results we will obtain further.

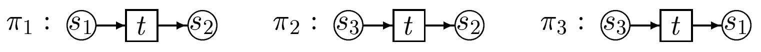


Figure 4.1.1

The reason for choosing such a notion of isomorphism can be explained shortly as follows (it will become very clear in Section 4.2 where some Petri net transformations are discussed). Generally, 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 behaviour of the net γ (that is, we want to have $\gamma \approx \gamma[\gamma_1 \leftarrow \gamma_2]$, for some equivalence relation $\approx$). But, for many transformations, γ_2 introduces new places or transitions which are copies of some places and transitions of γ_1, having in common with γ_1 only a subset of places. Usually, copies of a transition are labelled as the transition is labelled; places are not labelled but it is not important at all whether we use a place s or a copy of it. In such a case, the notion of isomorphism we adopted seems to be suitable enough.

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 labelled by λ. Let $\pi = (N, p)$ be a process of a net γ. An *abstraction* of π is any labelled partially ordered set (E', A, p'), where [6]:

- $E' = \{e \in E | p(e) \neq \lambda\}$;
- $(e, e') \in A$ iff there is a path in π leading from e to e';
- $p' = p|_{E'}$.

The equivalence class with respect to isomorphism (of labelled partialy ordered sets [7]) induced by (E', A, p'), denoted by $PW(\pi)$, is called the *partial word* associated to π. The set of all partial words of γ is denoted by $PW(\gamma)$.

Definition 4.1.2

(1) Two nets 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) Two nets 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 modelled by conditions labelled by interface places. Thus, we introduce the concept of an S^c-abstraction of a process as follows. Let γ be a net, $S^c \subseteq S$, and π a process of γ. An *S^c-abstraction* of π is any labelled partially ordered set $(B' \cup E', A, p')$, where:

- $B' \subseteq \{b \in B | p(b) \in S^c \ \wedge \ (|{}^\bullet b| = 0 \ \vee \ |b^\bullet| = 0)\}$, and $E' = \{e \in E | p(e) \neq \lambda\}$;

[6]A labelled 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.

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

(i) $(\forall x, y \in X_1)(x \leq_1 y \Leftrightarrow f(x) \leq_2 f(y))$;

(ii) $(\forall x \in X_1)(p_1(x) = p_2(f(x)))$.

– $(x, y) \in A$ iff there is a path in π leading from x to y;

– $p' = p|_{B' \cup E'}$.

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-$PW(\gamma)$.

Definition 4.1.3 *Let $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$.*

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

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

This definition says that no matter how the initial marking on S^c is increased that 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 writing 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 $M \in \mathbf{N}^{S^c}$ 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 4.1.1

(1) $\approx_m \subseteq \approx$;

(2) $\approx_m \subseteq \overset{M}{\approx}_m \subseteq \overset{M'}{\approx}_m$, for all $M, M' \in \mathbf{N}^{S^c}$ with $M \leq M'$.

It is 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 4.1.1 *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 give 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 hyphotesis, 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 exactly the conditions in B^c). One can easily prove that $\pi' \in \Pi(\gamma_1 \circ \gamma_2)$ and $PW(\pi) = PW(\pi')$. □

Corollary 4.1.1 *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 4.1.1(2) it follows that

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

and by Theorem 4.1.1 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 such sets A. □

Corollary 4.1.2 *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 $\gamma_2 \in PN(S_1^c, M_0^c)$ 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, $\gamma_0, \gamma_1, \gamma_2 \in PN(S_1^c, M_0^c)$, 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 4.1.1. □

From a practical point of view we are interested in Corollary 4.1.2. 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 favourable particular case would be when a marking $M \in \mathbf{N}^{S^c}$ exists such that the processes of the nets $(\gamma + M')$, 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 labelled 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 labelled occurrence net obtained from π by removing all the conditions in C.

Definition 4.1.4 *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 succesors and labelled by places in S^c, then we get two isomorphic labelled occurrence nets.

Lemma 4.1.1 *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 4.1.5 *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 each 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 4.2.3(a)(b) are process stable w.r.t. $M = (0, \ldots, 0)$, where $S^c = \{s_1, \ldots, s_k\}$. We have:

Theorem 4.1.2 *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 (γ_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 2.2.1 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 4.1.1 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, $\pi_1' \cong \pi_2'$. $\Box$

4.2 Proving Correctness of Petri Net Structural Transformations

The Corollary and Theorem 4.1.2 may be used to prove the correctness of some Petri net transformations. 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 4.1.2 and Theorem 4.1.2. In what follows we will exemplify our discussion by considering some 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 [24], which takes many journal pages and uses techniques of graph colouring, with our proofs of Theorem 4.2.1 and 4.2.2).

Recall first that processes of Petri nets can be defined inductively in terms of composition of initial occurrence nets and elementary occurrence nets associated to transitions (Definition 3.2.2, for the case of an empty set of jumps).

According to [24], a labelled marked Petri net is called *normalized* if the weight function and the initial marking take values into $\{0, 1\}$. In [24] it was shown that every λ-free labelled marked Petri net (that is, λ cannot be a label) is partial word equivalent with 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. Also the initial marking is needed to be processed in the first step. 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 of the size (in terms of places, transitions, and arcs) of the produced net. In [32] has been pointed out that the normalization algorithm can be also applied to labelled nets and, further, in [33] has been noted that the normalization preserves the processes as well if one consider the notion of isomorphism we already adopted. Moreover, in [36] another solution for processing the initial marking was proposed. It consists of a distribution of the initial marking into the old places. No place and transition is needed more. 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 [36] for processing the initial marking.

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:

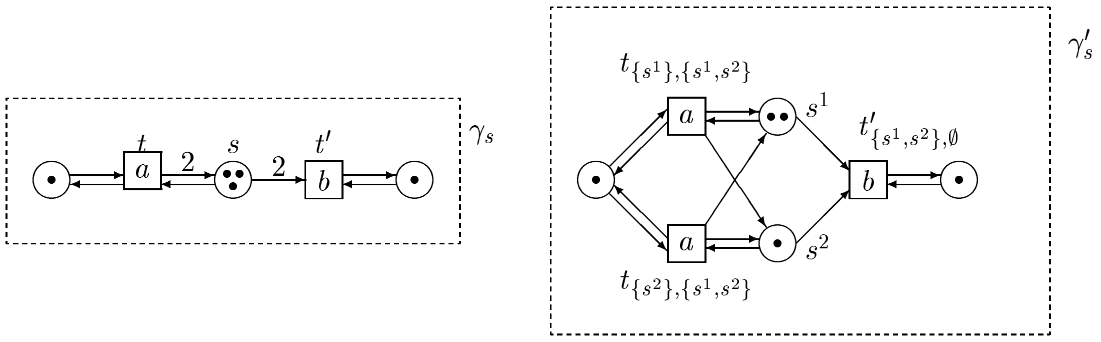
- $\gamma'_s = (\Sigma'_s, M'_s, l'_s)$, $\Sigma'_s = (S'_s, T'_s, F'_s, W'_s)$;
- $S'_s = (S_s - \{s\}) \cup C(s)$, where $C(s) = \{s^1, \ldots, s^{n_s}\}$ is a set of n_s new places (copies of the place s);
- $T'_s := \bigcup_{t \in {}^\bullet s^\bullet} C(t)$, where $C(t) = \{t_{A,B} | A, B \subseteq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

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

- $M'_s|_{C(s)}$ 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 $s' \in S_s - \{s\}$;



t
a
2
s
2
t'
b
γ_s
$t_{\{s^1\},\{s^1,s^2\}}$
a
s^1
$t'_{\{s^1,s^2\},\emptyset}$
b
a
s^2
$t_{\{s^2\},\{s^1,s^2\}}$
γ'_s

Figure 4.2.1

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

This transformation is exemplified in Figure 4.2.1.

Theorem 4.2.1 ([36])
The net γ' yielded by Transformation-A on the input γ satisfies $\gamma \approx_P \gamma'$.

Proof In the view of Corollary 4.1.2 we have to prove that $\gamma_s \approx_{mP} \gamma'_s$, for all s with $n_s > 1$. To prove this it is enough to show that, for every marking M on the interface places, we have:

- 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.

But these facts follow directly from the definition of γ'_s. $\Box$

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 $t^i \in T'_s$.

This transformation is exemplified in Figure 4.2.2 for the case of the place s_1. Clearly, the net yielded by Transformation-B is normalized.

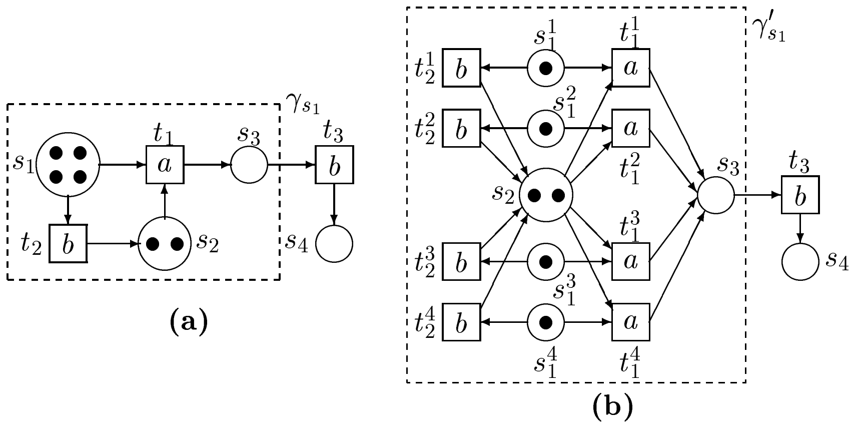


Figure 4.2.2

Theorem 4.2.2 ([24])
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 works in this case too. □

The above two theorems assure the correctness of the normalization algorithm.

In [28] 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 [32] 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-labelled

Petri nets, the normal form was improved with respect to the finite transition sequence behaviour. More precisely, it was shown that every λ-labelled Petri net is equivalent to a $(2,2)$-transition restricted net (with respect to the finite transition sequence behaviour). This result was extended in [34] 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 labelled net. In the view of the results above we may assume that γ is normalized ([32], [34]). 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 4.2.3(a) then Σ'_t is the net in Figure 4.2.3(b);
- if Σ_t is the net in Figure 4.2.3(c) then Σ'_t is the net in Figure 4.2.3(d);
- the initial marking of γ'_t on the places $s_1, \ldots, s_k$ is the same as the initial marking of γ on these places;
- the labeling is that specified in diagrams.

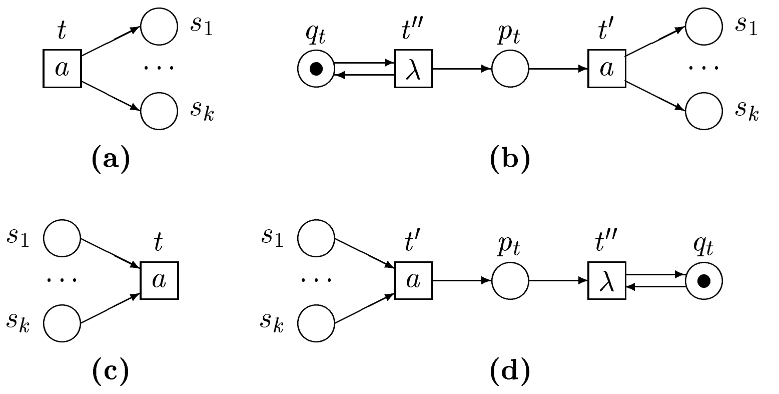


Figure 4.2.3

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 4.2.3 ([34])
The net γ' yielded by Transformation-C on the input γ satisfies $\gamma \approx_{PW} \gamma'$.

Proof In the view of Corollary 4.1.2 we have to prove that $\gamma_t \approx_{mPW} \gamma'_t$ for all t with the property $|{}^\bullet t| = 0$ or $|t^\bullet| = 0$. But this job is as simple as minute it is, and therefore it is omitted (for the nets in Figure 4.2.3(a)(b) we may use Theorem 4.1.2). $\Box$

Transformation-D: Let γ be a normalized net γ 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 4.2.4, but with the next remarks:

- in the case $n = 1$ or $n = 2$ the places s_1 or s_1 and s_2 respectively are directly connected to t^n;
- in the case $m = 1$ or $m = 2$ the only successors of t^n are s_{n+1} or s_{n+1} and s_{n+2} respectively;
- 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 4.2.4 is labelled 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).

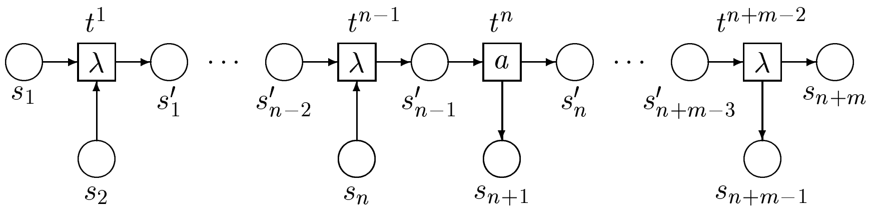


Figure 4.2.4

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 4.2.4 ([34])
The net γ' yielded by Transformation-D on the input γ satisfies $\gamma \approx_{PW} \gamma'$.

We want to stress again on the simplicity (and elegance, in our view) of the correctness proofs of these transformations in comparison with the original ones. As one could expect, not any Petri net transformation can be proved, about its correctness, by our method. We will show that by considering a transformation proposed in [4] (**Transformation-E** below). This transformation is in connection with *Petri nets with λ-transitions*, abbreviated *λ-PTN*, which are labelled Petri nets whose labelling function l has the property: for each transition t, $l(t)$ equals t or λ (in [4] such nets were called *strictly labelled 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 labelled by a. Replace recursively the subnets γ_A generated by A, where A is as above, by γ'_A defined as follows:

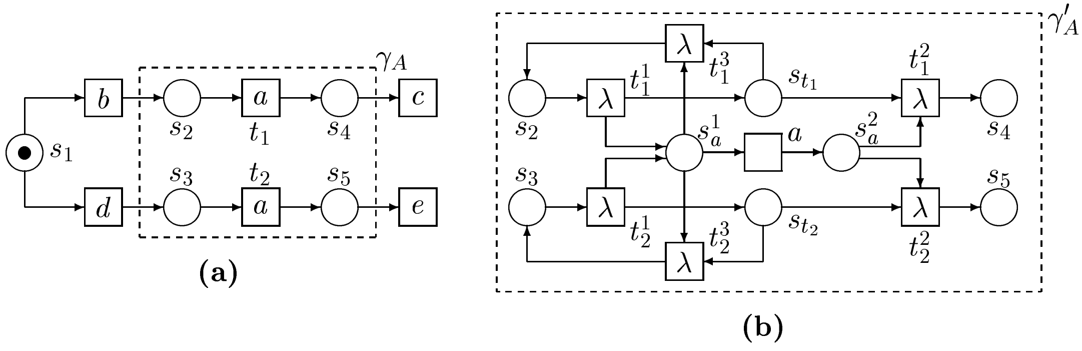
- $\gamma'_A = (\Sigma', M'_0, l')$, $\Sigma' = (S', T', F', W')$;
- $S' = S_A \cup \{s^1_a, s^2_a\} \cup \{s_t | t \in A\}$, where S_A is the set of places of γ_A;
- $T' = \{a\} \cup \{t^1, t^2, t^3 | t \in A\}$;
- $W'(s, t^1) = W(s, t)$, for all $s \in S_A$ and $t \in A$,
  $W'(t^2, s) = W(t, s)$, 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 $s' = s_a^1$, for all $t \in A$,
$W'(s', t^2) = 1$, if $s' = s_t$ or $s' = s_a^2$, for all $t \in A$,
$W'(s_a^1, a) = W'(a, s_a^2) = 1$,
$W'(s_a^1, t^3) = W'(s_t, t^3) = 1$, 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 Example 4.2.4(a)(b). In [4] it was shown that if γ is



$$\begin{tikzcd} s_2 \arrow[r] & a \arrow[r] & s_5 \\ s_3 \arrow[r] & a \arrow[r] & s_4 \end{tikzcd}$$

(c) an S^c-partial word of γ'_A

Figure 4.2.4

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 bisimular (γ' being the net yielded by Transformation-E); therefore, $\gamma \approx_{PW} \gamma'$ (see [4] for more details).

The correctness of Transformation-E cannot be proved by Corollary 4.1.2. Indeed, the nets γ_A and γ'_A in Figure 4.2.4(a)(b) (γ_A is without auto-concurrency and multiple need) are not $\approx_{mPW}$-related because the labelled partially ordered set in Figure 4.2.4(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 kind of transformations should be achieved. Concerning Petri nets with λ transitions we can prove that, in general, there is no transformation of labelled nets into partial word equivalent λ-PTN's.

Proposition 4.2.1 *There is a labelled net which is not partial word equivalent to any net with λ transitions.*

Proof Consider the labelled net γ in Figure 4.2.4(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' labelled 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 labelled 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. □

4.3 A Methodology for Validation of Petri Net Models

One aim of simulation is to *validate* the model w.r.t. some desired behavioural 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 [11]). 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 3 can be used for validation of Petri net models.

Suppose that a net γ can be decomposed as follows:

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

(the sets of interface places in brackets). Formally, we may write

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

We suppose that the net γ_n is of reazonable 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 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 possibe 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 the set of jumps.

A partial answer to the first question is done by the process sample generation theorem (Theorem 3.2.1). The answer to the second question seems to be more complicated. However, 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, togheter 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').

A related topic to this methodology of validation can be found in the next section.

5 Modular Model Checking

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 ([7], [8]), and an alternative approach based on showing inclusion between ω-automata was later devised by R. Kurshan ([21]). Unfortunately, temporal logic model checking procedures sufferes 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 section 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 5.4). Since for safe 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 behaviour";
- the system K_2 may abstract from some parts of the behaviour 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, [14], [8], [20]).

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.*

5.1 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 [8]. 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 [8]. 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

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

[8]This restriction is necessary in order to be able to transfer properties from systems with "more behaviour" to systems smaller in the simulation preorder, which have "less behaviour" (see Section 5.2).

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 $\mathit{inf}(\sigma) \cap A \neq \emptyset$, 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 $\overline{U}$.

5.2 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 5.2.1 *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 5.2.1(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 5.2.1. 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 5.2.2 *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 [14] (except for the fact that we use fairness constraints given as Büchi but not as Streett acceptance conditions).

Proposition 5.2.1 *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 5.2.1 ($i_0, i_1, \ldots$ 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 5.2.1. 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 5.2.1 *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,

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

   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 5.2.1 is the following result.

Corollary 5.2.1 *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 $K_1 \models \hat{\overline{\varphi}}$.*

5.3 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, whitout 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 behaviour (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 5.3.1 *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) $Q_0 = \{q \in Q | q|_{V_1} \in Q_0^1 \ \wedge \ q|_{V_2} Q_0^2\}$;

(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 [16] and [17], 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 $\overline{\mathcal{F}}_2 = \{\{q|_{V_1} | q \in \mathit{Reach}(K_1 \circ K_2) \ \wedge \ q|_{V_2} \in A_2\} | A_2 \in \mathcal{F}_2\}$.

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

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

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

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

Theorem 5.3.1 *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 $\mathit{inf}(\sigma) \cap A' \neq \emptyset$, and so there is $q \in Q \cap \mathit{inf}(\sigma)$. 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 5.3.1 *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 5.3.1 and Corollary 5.2.1. □

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 [8] and [17] for more comments). On the other side, to have a good abstraction it is important to produce exactly $\rho_{1,2}^e$, or to produce approximations sufficiently closed to $\rho_{1,2}^e$ so that we can still verify interesting properties of the system.

5.4 Modular Model Checking of Petri Nets

To each safe 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 componentwise 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 [22], 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 [9].

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

- for a safe 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 safe 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 safe 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 5.3.

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 safe e-modules whose underlying nets are elements of $PN(S^c, M_0^c)$, $\mathcal{A}$ is a set of commom 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 5.3.1 Further, consider the fair e-module $(\mathcal{J}_{1,2}, \mathcal{F}_{1,2})$, where:

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

[9]It was pointed out in [12] that in the case of 1-safe 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 safe nets.

– $\mathcal{F}_{1,2} = \mathcal{F}_1 \cup \overline{\mathcal{F}}_2$, 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 5.4.1 *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 safe, 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 safe, then $\mathcal{J}_1$ and $\mathcal{J}_2$ are safe. 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 5.3.1.

(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 5.4.1 *Let $\gamma_1, \gamma_2 \in PN(S^c, M_0^c)$ be two compatible nets. If $\gamma_1 \circ \gamma_2$ is safe, 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 $\gamma_1 \circ \gamma_2$ (Definition 3.2.3).*

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.

5.5 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

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

where, for a path σ,

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

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 5.2 hold also for such of fairness requirements.

The composition of two structures (under these fairness constraints) may be defined as in Section 5.3 but with the difference:

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

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 $\overline{\mathcal{F}}_2^e = \{A'_2 | A_2 \in \overline{\mathcal{F}}_2$ 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 5.3) 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 $\overline{\mathcal{F}}_2^i = \{A'_2 | A_2 \in \overline{\mathcal{F}}_2$ 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 5.3.1 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: $A'_2 \in \mathcal{F}_1$, $A'_2 \in \overline{\mathcal{F}}_2^e$, and $A_2 \in \overline{\mathcal{F}}_2^i$.

The first two can be treated in the same way as the case 1 in the proof of Theorem 5.3.1. 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') occurrs 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, $infs(\sigma') \cap A'_2 \neq \emptyset$, and so the path σ' is fair.

The results in Section 5.4 may now be easily reformulated for the case of step fairness constraints.

Conclusions and Related Work

Nets equiped with subsets of interface places (modules, in our paper) appear naturally when a distributed systems is modelled 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 [6], the modules (called there *open interface nets*) are non-labelled and endowed with a set of markings on the internal places (called stable states). Our modules are exactly those from [41] (called there *host nets*) or [39] (called there *net components*), with the difference that in [39] they are not labelled; the asynchronous composition for modules we considered is like in [41] (called there *place composition*). The set of interface places may be partitioned (as we have already said in Section 2) into subsets of input and output places as in [19]. 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 [41] (by adding two transitions t_s^- and t_s^+, for each interface place s), in [39] (by means of *actions*, which are jumps in our paper), and in [19], 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 modelling reactive systems which may interact with each other.

Section 3 considers 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 is 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 operation of composition we considered, but with a little effort we can obtain a compositional one (Corollary 3.1.2); it is totally diferrent than the semantics considered in the papers cited above. For example, the process semantics in [19] was suitable modified such that the compositional property was achieved, and the CFFD-semantics in [39] is a conjunction of stable failures, divergence traces, and infinite traces (which lead to compositionality). The second part of Section 3 takes into consideration the generation of process samples of a module w.r.t. some submodules. The generation is done via e-modules which are particular cases of jumping nets (as we have mentioned, [39] consideres actions which are jumps in our terminology, and which are used as an abstraction mechanism but in a different way than us; the concept of a jump and its main use as an abstraction mechanism goes back to [30] and [31]).

The main line we follow in Section 4.1 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 propose 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 the Section 4.2 points out.

Section 5 considers temporal logic model checking in connection with modular design. Since we do not restrict the environment action on modules, we cannot transfer, in a direct way, properties from components to the entire system; however, some delayed versions of them may be transferred. The logic we use is $\forall CTL^*$ which has enough expressive power.

Finding efficient methods to describe or approximate the set of jumps induced by sub-modules is of great importance for practical applications. Certainly, there is no general method to that; we have to restrict ourself to subclasses of (safe) nets, and this one could be an interesting subject of study. By taking into consideration partial information about the internal structure of the module and/or environment (as it has been already mentioned in Section 2) we can reduce the size of the set of jumps. On the other side, using semaphor variables (as partial information), the interface places may be regarded both as input and output places, without a specific distinction. When some semaphor variable is on, the module and the environment may work together; when the variable is off, only the environment may work. The environment is setting this variable; the module has just to use read arcs ([42]) in order to know when it has the right to work.

Acknowledgements

We thank Walter Vogler in Augsburg (Germany) for some comments on a preliminary version of this paper.

References

[1] R. Alur, Th.A. Henziger: *Reactive Modules*, in: Proc. of the 11th IEEE Symposium on Logic in Computer Science LICS, 1996, 207–218.

[2] 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.

[3] 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.

[4] E. Best, R. Devillers, A. Kiehn, L. Pomelo: *Concurrent Bisimulations in Petri Nets*, Acta Informatica 28, 1991, 231–264.

[5] E. Best, C. Fernandez: *Nonsequential Processes. A Petri Net Point of View*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1988.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] J. Desel: *Validation of System Models Using Partially Ordered Runs*, to appear 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.

[12] J. Esparza, S. Melzer: *Model Checking LTL Using Constraint Programming*, Technical Report, Technische Universität München, 1997.

[13] 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.

[14] O. Grumberg, D.E. Long: *Model Checking and Modular Verification*, ACM Transactions on Programming Languages and Systems 16, 1994, 843–871.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Z. Manna, A. Pnueli: *The Temporal Logic of Reactive and Concurrent Systems. Specification*, Springer-Verlag, 1992.

[23] K. Müller: *Constructable Petri Nets*, Journal of Information Processing and Cybernetics EIK 21, 1985, 171–199.

[24] 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).

[25] J.L. Peterson: *Petri Net Theory and the Modelling of Systems*, Prentice-Hall, 1981.

[26] W. Reisig: *Petri Nets. An Introduction*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1985.

[27] W. Reisig: *Elements of Distributed Algorithms. Modeling and Analysis with Petri Nets*, Springer-Verlag, 1998.

[28] 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.

[29] I. Suzuki, T. Murata: *A Method for Stepwise Refinement and Abstraction of Petri Nets*, Journal of Computer and System Science 27, 1983, 51–76.

[30] F.L. Ţiplea: *Contributions to the Language Theory of Petri Nets*, Ph.D. Thesis, "Al.I.Cuza" University of Iasi (Romania), 1993.

[31] F.L. Ţiplea, T. Jucan: *Jumping Petri Nets*, Foundations of Computing and Decision Sciences 19, 1994, 319–332.

[32] 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.

[33] 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.

[34] F.L. Ţiplea, M. Katsura, M. Ito: *On a Normal Form of Petri Nets*, Acta Cybernetica 12, 1996, 295–308.

[35] F.L. Ţiplea, E. Mäkinen: *Jumping Petri Nets. Specific Properties*, Fundamenta Informaticae 32, 1997, 373–392.

[36] 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.

[37] 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.

[38] R. Valette: *Analysis of Petri Nets by Stepwise Refinement*, Journal of Computer and System Science 18, 1979, 35–46.

[39] 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.

[40] 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.

[41] W. Vogler: *Modular Construction and Partial Order Semantics of Petri Nets*, Lecture Notes in Computer Science 625, Springer-Verlag, 1992.

[42] 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).