Modular Verification of Petri Nets Properties: A Structure-Based Approach

Kais Klai[1], Serge Haddad[2], and Jean-Michel Ilié[3]

[1] LaBRI CNRS UMR 5800, Université de Bordeaux I, Talence, France
kais.klai@labri.fr
[2] LAMSADE CNRS UMR 7024, Université de Paris Dauphine, France
haddad@lamsade.dauphine.fr
[3] LIP6 CNRS UMR 7606, Université de Paris 6, France
Jean-Michel.ilie@lip6.fr

Abstract. In this paper, we address the modular verification problem for a Petri net obtained by composition of two subnets. At first, we show how to transform an asynchronous composition into a synchronous one where the new subnets are augmented from the original ones by means of linear invariants. Then we introduce a non-constraining relation between subnets based on their behaviour. Whenever this relation is satisfied, standard properties like the liveness and the boundedness and generic properties specified by a linear time logic may be checked by examination of the augmented subnets in isolation. Finally, we give a sufficient condition for this relation which can be detected modularly using an efficient algorithm.

Keywords: Abstraction, modular verification, (de)composition, Petri nets.

1 Introduction

The validation of complex distributed systems must come up to the well-known state explosion problem. Thus, numerous validation techniques have been proposed in order to reduce the number of states to be explored.

Among them, the modular verification approaches aim to take benefit from some knowledge about the components of the system and the way they communicate. The synchronous composition between components is very popular in system verification since, from properties of the components, one can deduce those of the system. For instance, the finiteness of the system is directly deduced from the fact that the composed modules are finite. Asynchronous composition usually better corresponds to systems where the modules are distributed and weakly coupled. In such a case, modules communicate asynchronously by message sending. Taking asynchronous communication into account during the validation process is generally a difficult task: for instance the system can be infinite even if the composed modules are finite.

In a verification process, the properties which are validated at first are the standard ones with respect to the model used. For instance, the boundedness or the liveness properties of a Petri net ensure a positive *a priori* on the correctness

F. Wang (Ed.): FORTE 2005, LNCS 3731, pp. 189–203, 2005.

of the design. The finiteness of the system is directly deduced from the fact that it is bounded, while the liveness property indicates that all the pieces of codes within a system remain available whatever the evolution of the system. The validation of the specific properties of a system often requires specification languages like temporal logic, able to express the causality between the state changes. Our work deals with the linear time logic LTL. Such a logic views the system like a set of runs. LTL may be checked on the fly, which means that the state space of the system is constructed step by step as the need of the verification occurs. Moreover whenever the property to be checked is detected false, a run highlighting the problem is exhibited for free to the designer.

Without any restriction on the composition, the efficiency of the modularverification rather depends on the system to be analyzed. For instance in [2], it is possible to minimize the reachable states of each module by hiding the internal moves, before the synchronization of modules. Reachability analysis has been proved to be effective on the resulting structure in [6] and the method has been extended to operate the model checking of LTL-X formulae (LTL without the "next" operator). Anyway, experimental results show that this technique is efficient for some models, but for others the combinatorial explosion is not really attacked.

Other approaches have proposed to restrict the application domain bylaying some construction rules down, either for the modules or their communication medium. The general idea consists in replacing the analysis of the global state space by the analysis of the state spaces of modules. Actually, the verification of system properties consists in checking separately some properties on each module then piecing the results together in order to conclude whether the system is correct. General properties are addressed as well as some sets of temporal properties. In general, the brute force approach which consists in partitioning the system in whatever subnets is bound to fail. Different approaches of composition have been proposed depending in particular, on the way each module can abstract its environment (see [10,9,1]). At first, some general properties of a Petri net were initially considered (boundedness, liveness); henceforth, the model checking problem of temporal formulae was investigated [5,8]. Anyway, rather restrictive conditions are forced, thus reducing drastically the application to concrete systems.

In this paper, we propose a new structure-based modular approach startingfrom a non-constraining relation between components. We start from a specification of the system in Petri nets without restriction and address the verification of both standard properties and linear time temporal logics (LTL-X). We then decompose the Petri net in two components such that their (common) interface only contains transitions. In order to abstract the environment of a component, we propose to take benefit from the existence of linear positive place-invariants. Such invariants which often exist in well-specified system, are used to enrich each subnet by some abstraction of the other subnet. However, to check the system properties in isolation on a component, one may need to check whether the other component does not constrain its behaviour. Thus we develop a modular

test of this constraining relation by analyzing in isolation the behaviour of the component which must be non-constraining. The principal contribution of our modular approach w.r.t. the existent works is the combination of structural and behavioural aspects. From the structural point of view, we furnish a general composition model where the system invariants are originally exploited in order to abstract modules. While, from the behavioural point of view, and in opposition to some existent techniques (see [11,2,6]), the synchronized product between the system components is avoided.

The paper is organized as follows: in section 2, we introduce ourdecomposition scheme showing how to handle asynchronous communication and deducing some useful properties. In section 3, we define the non-constraining relation between components, we propose a sufficient condition, show how to check it efficiently and bring out our compositionality results. At last, concluding remarks and perspectives are given in section 4.

2 Decomposition Scheme

2.1 Preliminaries and Notations

In this section we recall the definition of a Petri net and some basic notions of Petri net theory. In order to decompose Petri nets, we also formalize the notion of subnets.

Vectors and matrices. Let v be a vector or a matrix, then v^T denotes its transpose. So if v, v' are two vectors then $v^T.v'$ corresponds to their scalar product. Let v be a vector of $\mathbb{N}^P$ then the support of v, denoted $||v||$, is defined by $||v|| = \{p \in P \mid v(p) > 0\}$.

Petri nets. Let P and T be disjoint finite sets of places and transitions respectively, the elements of $P \cup T$ are called nodes. A net is a tuple $N = \langle P, T, Pre, Post\rangle$ with the backward and forward incidence matrices Pre and $Post$ defined by Pre (resp. $Post$): $(P \times T) \longrightarrow \mathbb{N}$. We denote by $Pre(t)$ (resp. $Post(t)$) the column vector indexed by t of the matrix Pre (resp. $Post$).
$W = Post - Pre$ is the incidence matrix of N. The preset of a place p (resp. a transition t) is defined as ${}^\bullet p = \{t \in T \mid Post(p,t) > 0\}$ (resp. ${}^\bullet t = \{p \in P \mid Pre(p,t) > 0\}$), and its postset as $p^\bullet = \{t \in T \mid Pre(p,t) > 0\}$ (resp. $t^\bullet = \{p \in P \mid Post(p,t) > 0\}$). The preset (resp. postset) of a set X of nodes is given by the union of the presets (resp. postsets) of all nodes in X. ${}^\bullet X^\bullet$ denotes the union of the preset and the postset of X.

In case of ambiguity the name of the corresponding net is specified: ${}^\bullet X(N)$, $X^\bullet(N)$ and ${}^\bullet X^\bullet(N)$.

A marking of a net is a mapping $P \longrightarrow \mathbb{N}$. We call $\Sigma = \langle N, m_0\rangle$ a net system with initial marking m_0 of N. A marking m enables the transition t ($m \xrightarrow{t}$) if $m(p) \geq Pre(p,t)$ for each $p \in {}^\bullet t$. In this case the transition can occur, leading to the new marking m', given by: $m'(p) = m(p) + W(p,t)$ for every place $p \in P$. We denote this occurrence by $m \xrightarrow{t} m'$. If there exists

a chain $(m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} m_2 \longrightarrow \ldots \xrightarrow{t_n} m_n)$, denoted by $m_0 \xrightarrow{\sigma} m_n$, the sequence $\sigma = t_1 \ldots t_n$ is also called a computation. A computation of infinite length is called a run. We denote by T^* (resp. T^∞) the set of finite (resp. infinite) sequences of T. T^ω denotes the set of all sequences of T ($T^\omega = T^* \cup T^\infty$). The finite (resp. infinite) language of (N, m_0) is the set $L^*(\langle N, m_0\rangle) = \{\sigma \in T^* \mid m_0 \xrightarrow{\sigma}\}$ (resp. $L^\infty(\langle N, m_0\rangle) = \{\sigma \in T^\infty \mid m_0 \xrightarrow{\sigma}\}$), also $L^\omega(\langle N, m_0\rangle) = L^*(\langle N, m_0\rangle) \cup L^\infty(\langle N, m_0\rangle)$. Moreover, $[N, m_0\rangle = \{m \text{ s.t. } \exists \sigma \in T^*,\ m_0 \xrightarrow{\sigma} m\}$ represents the set of reachable markings of $\langle N, m_0\rangle$.

Subnets. Let $N = \langle P, T, Pre, Post\rangle$ be a Petri net. N' is a subnet of N induced by (P', T'), $P' \subseteq P$ and $T' \subseteq T$, if $N' = \langle P', T', Pre', Post'\rangle$ is a Petri net s.t. $\forall (p,t) \in P' \times T'$, $Pre'(p,t) = Pre(p,t)$ and $Post'(p,t) = Post(p,t)$. If m is a marking of N then we define its restriction to places of N' as follows: $\forall p \in P'\ m'(p) = m(p)$. The restriction of m to P' is denoted by $m_{\lfloor P'}$.

Linear invariants. Let v be a vector of $\mathbb{N}^P$, v is a positive linear invariant iff $v.W = 0$. If v is a positive linear invariant and $m \xrightarrow{\sigma} m'$ is a firing sequence then $v^T.m' = v^T.m$.

Sequences. Let σ be a sequence of transitions ($\sigma \in T^\omega$). λ denotes the empty sequence. For a transition t in T, we define $|\sigma|_t$ by:
If t occurs infinitely often in σ then $|\sigma|_t = \infty$ else $|\sigma|_t = k$ where, k is the number of occurrences of t in σ. We extend this notation to subsets X of transitions: $|\sigma|_X = \sum_{t \in X} |\sigma|_t$. Moreover, $|\sigma|$ is the number of transitions in σ. By analogy to the notations introduced on sets of nodes, we define:
${}^\bullet\sigma = \cup_{t,[|\sigma|_t > 0]} {}^\bullet t$, $\sigma^\bullet = \cup_{t,[|\sigma|_t > 0]}\ t^\bullet$ and ${}^\bullet\sigma^\bullet = {}^\bullet\sigma \cup \sigma^\bullet$.
The set of transitions which occur infinitely often in σ is denoted by $inf(\sigma)$.
The projection of a sequence σ on a set of transitions $X \subseteq T$ is the sequence obtained by removing from σ all transitions that do not belong to X. It is defined as follows: $\lfloor : T^\omega \times 2^T \longrightarrow T^\omega$ s.t.:

- $\lambda_{\lfloor X} = \lambda$,
- $\forall \sigma \in T^\omega$ and $t \in T$ if $t \in X$ then $(t.\sigma)_{\lfloor X} = t.\sigma_{\lfloor X}$ else $(t.\sigma)_{\lfloor X} = \sigma_{\lfloor X}$.

The projection function is extended to sets of sequences (i.e. languages) as follows: $\forall \Gamma \subseteq T^\omega$, $\Gamma_{\lfloor X} = \{\sigma_{\lfloor X} \mid \sigma \in \Gamma\}$.

2.2 Synchronous Decomposition

In this section, we define the decomposition of a Petri net N into two subnets N_1 and N_2 through a set of interface transitions T_I.

Definition 1 (Decomposable Petri net). *Let $N = \langle P, T, Pre, Post\rangle$ be a Petri net and T_I a non empty subset of T. N is said to be decomposable into $N_1 = \langle P_1, T_1 = T_{11} \cup T_I, Pre_1, Post_1\rangle$ and $N_2 = \langle P_2, T_2 = T_{21} \cup T_I, Pre_2, Post_2\rangle$ through the interface T_I if:*

- $P = P_1 \cup P_2$ *and* $T = T_1 \cup T_2$,
- $P_1 \cap P_2 = \emptyset$, $T_{11} \cap T_{21} = \emptyset$,

- $\forall i \in \{1,2\}$, $\forall (p,t) \in P_i \times T_i$, $Pre_i(p,t) = Pre(p,t)$ *and* $Post_i(p,t) = Post(p,t)$,
- $\forall i,j \in \{1,2\}$, $i \neq j$, $\forall (p,t) \in P_i \times (T_j \setminus T_I)$, $Pre(p,t) = Post(p,t) = 0$

Notation: From now on, tuple $N_d = \langle N_1, T_I, N_2 \rangle$ denotes the decomposition of the net N into N_1 and N_2 through T_I.

Note that the composition of subnets by fusion of transitions occurs in a large class of Petri net models. Even if this kind of interface is especially used to model synchronous composition, we will see that our modular technique allows one to handle asynchronous composition as well, thanks to the exploitation of the positive linear invariants of the system. Figure 1 illustrates an example of a decomposable Petri net model. It models a simplified *client-server* system. The server switches between states *Passive* and *Active* on reception of *On* and *Off* signals respectively. On the other side, the client is initially *Idle*. When it wants to send a message, it waits for the server to be *Active* (place *Wserv*). Then, it sends its message and waits for a positive or negative acknowledgement (place *Wack*). In the case of a positive acknowledgement, it becomes again *Idle*. Otherwise, it tries to retransmit the message (place *Fail*). On reception of a message, the server analyzes it and sends a positive or negative acknowledgement (place *Analyze*).

The considered set of transitions T_I is $\{Send, Cons, Ncons\}$ (the full transitions of Figure 1). Here the subnet of the server (generated by the bold places of Figure 1) is unbounded due to the place *Mess* and any other choice of interface between the client and the server will lead to similar problems. A correct modular approach should analyze a component of the system completed by an abstraction of its environment. In the next subsection, we show how to exploit system invariants in order to automatically construct such an abstraction.

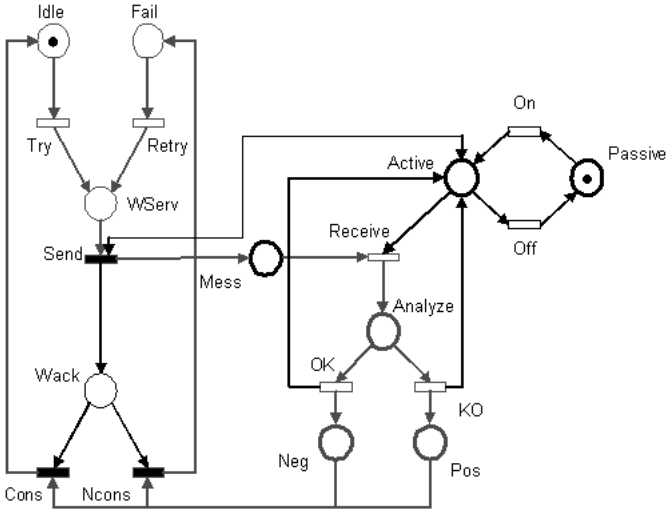

Fig. 1. A decomposable Petri net

2.3 Exploiting the Linear Invariants

A linear invariant of a Petri net corresponds to a safety property of the modelled system. Due to the equation defining such invariants, their computation is reduced to find a generative family of positive solutions of a linear equation system. Although the worst case time complexity of this computation is not polynomial, in practice the algorithm behaves efficiently and its usual time complexity is negligible w.r.t. the reachability graph construction. Thus, this approach is widely used for analysis of Petri nets and integrated in numerous softwares.

Here, we propose to use a linear invariant as a witness of thesynchronization between two subnets. Consequently, we look for linear invariants whose support intersects the places of the two subnets. Let V_{dec} be the subset of positive place-invariants which fulfill the above condition, picked from a generative family of a decomposable Petri net N. With each item $v \in V_{dec}$, we associate two places $a_1^{(v)}, a_2^{(v)}$ where $a_i^{(v)}$ is added to the subnet N_j in such a way that its current marking summarizes the information given by the positive place-invariant$v$. The obtained net is called *component subnet* and denoted from now on by $\widehat{N_i}$. V_{dec} will be called the set of global invariants. Given a place$p$, the vector 1_p in the following definition denotes the vector of $\mathbb{N}^P$ where each element is zero except the one indexed by the place p.

Definition 2 (*Component subnet*). *Let $N_d = \langle N_1, T_I, N_2\rangle$ be a decomposition of a Petri net N. The component subnet related to $N_i = \langle P_i, T_i, Pre_i, Post_i\rangle$ ($\{i,j\} = \{1,2\}$) is a Petri net $\widehat{N_i} = \langle \widehat{P_i}, \widehat{T_i}, \widehat{Pre_i}, \widehat{Post_i}\rangle$ such that:*

- $\widehat{T_i} = T_i$,
- $\widehat{P_i} = P_i \cup A_j$, *with* $A_j = \{a_j^{(v)} | v \in V_{dec}\}$ *the set of abstraction places.*

Let Φ be a mapping from $P \cup A_1 \cup A_2$ to $\mathbb{N}^{P \cup A_1 \cup A_2}$ defined by:
$\forall p \in P, \Phi(p) = 1_p$ *and* $\forall a_i^{(v)} \in A_i, \Phi(a_i^{(v)}) = \Sigma_{p \in P_i} v(p).1_p$

- $\forall p \in \widehat{P_i}, \forall t \in \widehat{T_i}$, $\widehat{Pre_i}(p,t) = Pre(t)^T.\Phi(p)$ *and* $\widehat{Post_i}(p,t) = Post(t)^T.\Phi(p)$

We illustrate the concept of *component subnets* on the client-server model of Figure 1. This model has the following generative family of invariants:

1. *Idle*+*Fail*+*Wserv*+*Wack*
2. *Active*+*Passive*+*Analyze*
3. *Idle*+*Fail*+*Wserv*+*Mess*+*Analyze*+*Pos*+*Neg*

The first two invariants are local to one subnet. Thus, only the third one, which covers both subnets, is used for the *component subnets* construction, leading to the components described in Figure 2. These subnets have been enlarged with two abstraction places, called here Abs_1 and Abs_2. Let us explain for instance the underlying meaning of the abstraction place Abs_1. Since $\Phi(Abs_1) = 1_{Idle} + 1_{Fail} + 1_{Wserv} + 1_{Mess}$, this place contains the sum of tokens of the four previous places. As *Wserv* is an input place of the transition *Send* and the three others ones aren't, $Pre(Abs_1, Send) = 1$. The other arcs are similarly deduced.

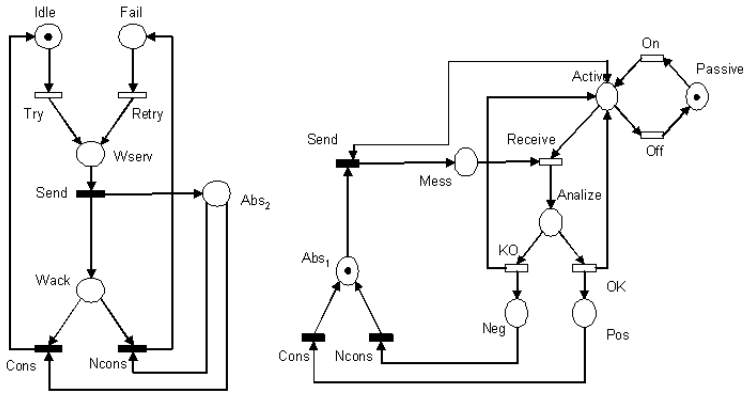

Fig. 2. The two component subnets

According to this interpretation, the following definition defines the mapping from a *global* marking (a marking of the original net) to markings of the component subnets.

Definition 3. *Let $N_d = \langle N_1, T_I, N_2 \rangle$ be the decomposition of a Petri net N and let $\widehat{N_i}$ $(i = 1, 2)$ be the induced component subnets. For each marking m of N, Φ_i the projection mapping on $\widehat{N_i}$ is defined by:* $\forall p \in P_i, \Phi_i(m)(p) = m^T.\Phi(p)$

The following proposition and corollary summarize what can be directly deduced from this decomposition about the relative behaviours of the net and its component subnets.

Proposition 1. *Let $N_d = \langle N_1, T_I, N_2 \rangle$ be a decomposition of a marked Petri net $\langle N, m \rangle$ and let $\widehat{N_i}$ $(i = 1, 2)$ be the induced component subnets. Then, the following assertion holds:*
$\forall \sigma \in T^*$, *if* $m \xrightarrow{\sigma} m'$ *then* $\Phi_i(m) \xrightarrow{\sigma_{\downarrow \widehat{T_i}}} \Phi_i(m')$

Proof. We prove the proposition for $\sigma = t$ being a single transition. The proposition follows by a straightforward induction. We consider the following cases.

case 1: $t \in \widehat{T_i}$
$m \xrightarrow{\sigma} m' \Rightarrow m \geq Pre(t) \Rightarrow \forall p \in \widehat{P_i}, m^T.\Phi(p) \geq Pre(t)^T.\Phi(p)$ (by positivity of $\Phi(p)$) $\Leftrightarrow \forall p \in \widehat{P_i}, \Phi_i(m)(p) \geq \widehat{Pre_i}(p,t)$. Thus $\Phi_i(m) \xrightarrow{t}$.
$m' = m - Pre(t) + Post(t) \Rightarrow \forall p \in \widehat{P_i}, {m'}^T.\Phi(p) = m^T.\Phi(p) - Pre(t)^T.\Phi(p) + Post(t)^T.\Phi(p) \Leftrightarrow \forall p \in \widehat{P_i}, \Phi_i(m')(p) = \Phi_i(m)(p) - \widehat{Pre_i}(p,t) + \widehat{Post_i}(p,t)$.
Thus $\Phi_i(m) \xrightarrow{t} \Phi_i(m')$.
case 2: $t \notin \widehat{T_i}$
$\forall p \in P_i, p \notin {}^\bullet t^\bullet$. Thus $\Phi_i(m')(p) = m'(p) = m(p) = \Phi_i(m)(p)$
Let $v \in V_{dec}$, since v is a flow: ${m'}^T.(\Phi(a_i^{(v)}) + \Phi(a_j^{(v)})) = m^T.(\Phi(a_i^{(v)}) + \Phi(a_j^{(v)}))$

Thus, $\Phi_i(m')(a_j^{(v)}) - \Phi_i(m)(a_j^{(v)}) = m'^T.\Phi(a_j^{(v)}) - m^T.\Phi(a_j^{(v)})$
$= m'^T.\Phi(a_i^{(v)}) - m^T.\Phi(a_i^{(v)}) = \Sigma_{p\in P_i} v(p).(m'(p) - m(p)) = 0$
(since any such $p \notin {}^\bullet t^\bullet$)
Thus $\Phi_i(m') = \Phi_i(m)$.

The assertions given in the following corollary are immediate consequences of the above proposition.

Corollary 1. *Let $N_d = \langle N_1, T_I, N_2\rangle$ be a decomposition of a marked Petri net $\langle N, m\rangle$ and let $\widehat{N_i}$ $(i = 1, 2)$ be the induced component subnets. Then, the following assertions hold:*

- $L^\omega(N,m)_{\lfloor \widehat{T_i}} \subset L^\omega(\widehat{N_i}, \Phi_i(m))$
- $\{\sigma_{\lfloor \widehat{T_i}} \mid \sigma \in L^\infty(N,m)$ *and* $Inf(\sigma) \cap \widehat{T_i} \neq \emptyset\} \subset L^\infty(\widehat{N_i}, \Phi_i(m))$
- (N,m) *is unbounded* $\Rightarrow \exists i\, (\widehat{N_i}, \Phi_i(m))$ *is unbounded*

3 Preservation of Properties

3.1 The Non-constraining Relation

In this section, we define the *non-constraining relation*: an asymmetric property to be checked between two given marked *component subnets* obtained from a decomposition of a net: $(\widehat{N_2}, \widehat{m_2})$ does not constrain $(\widehat{N_1}, \widehat{m_1})$ if for any firing sequence enabled from $(\widehat{N_1}, \widehat{m_1})$, there exists a firing sequence enabled from $(\widehat{N_2}, \widehat{m_2})$, which both have the same projection on the interface transitions. Under such a relation, we prove that the firing sequences enabled in the non constrained component exactly represent the firing sequences of the global net, up to the projection on the transition interface.

Definition 4 (Non-constraining relation). *Let $\langle \widehat{N_1}, \widehat{m_1}\rangle$ and $\langle \widehat{N_2}, \widehat{m_2}\rangle$ be the two* component subnets *induced by a decomposition of a Petri net $\langle N, m\rangle$: $\langle \widehat{N_2}, \widehat{m_2}\rangle$ does not constrain $\langle \widehat{N_1}, \widehat{m_1}\rangle$ iff $L^\omega_{\lfloor T_I}(\langle \widehat{N_1}, \widehat{m_1}\rangle) \subseteq L^\omega_{\lfloor T_I}(\langle \widehat{N_2}, \widehat{m_2}\rangle)$.*

When each component doesn't constrain the other one we say that they are *mutually non-constraining*.

Proposition 2. *Let $\langle \widehat{N_1}, \widehat{m_1}\rangle$ and $\langle \widehat{N_2}, \widehat{m_2}\rangle$ be the two marked component subnets induced by a decomposition of a marked Petri net $\langle N, m\rangle$. If $\langle \widehat{N_2}, \widehat{m_2}\rangle$ does not constrain $\langle \widehat{N_1}, \widehat{m_1}\rangle$ then the following assertion holds:*
$\forall \sigma_1 \in \widehat{T_1}^*$ *:* $\widehat{m_1} \xrightarrow{\sigma_1} \widehat{m_1'} \Rightarrow \exists \sigma \in T^*$ *and* $\exists m' \in \mathbb{N}^P$ *s.t.:*
$\sigma_{\lfloor T_1} = \sigma_1$, $m \xrightarrow{\sigma} m'$ *(and* $\Phi_1(m') = \widehat{m_1'}$ *by proposition 1).*

Proof. Let $\sigma_1 = \sigma_1^0.t^1.\cdots.t^k.\sigma_1^k$ with $\forall m, \sigma_1^m \in T_{11}$ and $t^m \in T_I$.
By hypothesis, there exists a firing sequence σ_2 in $(\widehat{N_2}, \widehat{m_2})$:
$\sigma_2 = \sigma_2^0.t^1.\cdots.t^k.\sigma_2^k$ with $\forall m, \sigma_2^m \in T_{21}$.
We claim that $\sigma = \sigma_1^0.\sigma_2^0.t^1.\cdots.t^k.\sigma_1^k.\sigma_2^k$ is the required sequence. We prove it by induction on the prefixes of σ.

Let $\sigma'.t$ be a prefix of σ such that σ' is a firing sequence, i.e. $m \xrightarrow{\sigma'} m''$.
By proposition 1, $\forall i \in \{1,2\}, \widehat{m_i} = \Phi_i(m) \xrightarrow{\sigma'_{\lfloor \widehat{T_i}}} \Phi_i(m'')$
By construction $\forall i \in \{1,2\}, (\sigma'.t)_{\lfloor \widehat{T_i}}$ is a prefix of σ_i. Thus $\Phi_i(m'') \xrightarrow{t_{\lfloor \widehat{T_i}}}$

Case 1: $t \in T_I$
Let $p \in {}^\bullet t$, $p \in P_i$ for some $i \in \{1,2\}$,
since $\Phi_i(m'') \xrightarrow{t}$, $m''(p) = \Phi_i(m'')(p) \geq Pre(p,t)$, we conclude that $m'' \xrightarrow{t}$.
Case 2: $t \in T_{i1}$ for some $i \in \{1,2\}$
Let $p \in {}^\bullet t$ since $t \in T_{i1}$ then $p \in P_i$,
since $\Phi_i(m'') \xrightarrow{t}$, $m''(p) = \Phi_i(m'')(p) \geq Pre(p,t)$, we conclude that $m'' \xrightarrow{t}$.

The non-constraining relation can be regarded as an inclusion relation between the languages of the components subnets, once projected on the shared transition interface. Checking such a property represents the main difficulty of our approach. A naive test of this relation would result in building the synchronized product of the components subnets reachability graphs, which could drastically limit the interests of our methods.

Here, we propose a new approach based on an abstraction of the system, namely, the *interface component subnet*, which allows one to represent the language of the global net compactly, up to a projection on the transition interface. It is obtained by connecting the interface transitions to the abstraction places of both components subnets. Figure 3 represents the interface component subnet of the net depicted in Figure 1.

Definition 5 (*Interface component subnet*). *Let $N_d = \langle N_1, T_I, N_2 \rangle$ be the decomposition of a Petri net N and let $\widehat{N_i}$ $(i = 1,2)$ be the induced component subnets. The* interface component subnet *related to N_d is a Petri net $\widehat{N_{int}} = \langle \widehat{P_{int}}, \widehat{T_{int}}, \widehat{Pre_{int}}, \widehat{Post_{int}} \rangle$ such that, for $i,j \in \{1,2\}$ and $i \neq j$:*
- $\widehat{T_{int}} = T_I$,
- $\widehat{P_{int}} = A_1 \cup A_2$, *with* $A_i = \{a_i^{(v)} | v \in V_{dec}\}$ *the set of abstraction places of* $\widehat{N_i}$,
- $\forall a \in A_i$, $\forall t \in \widehat{T_{int}}$, $\widehat{Pre_{int}}(a,t) = \widehat{Pre_j}(a,t)$ *and* $\widehat{Post_{int}}(a,t) = \widehat{Post_j}(a,t)$.

Using Proposition 1, one can immediately state the following: $\forall i \in \{1,2\}$, $L^\omega_{\lfloor T_I}(\widehat{N_i}, \widehat{m_i}) \subseteq L^\omega(\widehat{N_{int}}, \widehat{m_{int}})$.

Proposition 3. *Let $N_d = \langle N_1, T_I, N_2 \rangle$ be a decomposition of a marked Petri net $\langle N, m \rangle$ and let $\widehat{N_i}$ $(i = 1,2)$ and $\widehat{N_{int}}$ be the induced component subnets.*

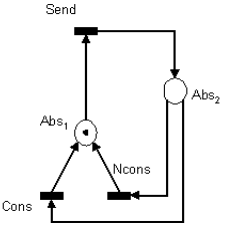

Fig. 3. The Client-Server interface component subnet

Then, the following assertion holds:
$\langle \widehat{N_i}, \widehat{m_i} \rangle$ *is non-constraining for* $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle \Rightarrow \langle \widehat{N_i}, \widehat{m_i} \rangle$ *is non-constraining for* $\langle \widehat{N_j}, \widehat{m_j} \rangle$ *(*$j \in \{1,2\}$ *and* $j \neq i$*)*

The proof is obvious since from proposition 1, one can immediately state that $L^{\omega}_{\lfloor T_I}(\widehat{N_i}, \widehat{m_i}) \subseteq L^{\omega}(\widehat{N_{int}}, \widehat{m_{int}})$ (for $i \in \{1,2\}$). This proposition will be exploited in the next subsection, in order to restrain the test of the non-constraining relation between two component subnets, to a lighter relation between a component subnet and the interface component subnet. The non-constraining relation can thus be checked modularly, since one component subnet is considered at a time. It is worth noting that the component subnet is computed only once even if a mutual non-constraining relation is checked.

3.2 The Non-constraining Test Algorithm

Now we explain how to check whether a given somponent subnet $\langle \widehat{N_i}, \widehat{m_i} \rangle$ is non-constraining for $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$. The proposed algorithm 3.2 works on the fly and focusses on the behaviour of the $\langle \widehat{N_i}, \widehat{m_i} \rangle$ around the interface. Its local moves induced by the local transitions are hence abstracted (unobserved) since they are not directly involved in the inclusion test. This allows us to reuse the concept of observation graph proposed in [4] to represent a reachability graph compactly. Here, the observed transitions are those of the interface transitions. The observation graph of $(\widehat{N_i}, \widehat{m_i})$ is a graph where each node is a set of markings linked by local (unobserved) transitions and each arc is labelled with an interface transition. Nodes of the observation graph are called meta-states and may be represented and managed efficiently by using *decision diagram techniques* (BDD for instance). In practice, the efficiency of this approach is obtained whenever the number of observed transitions is small with respect to the total number of transitions ([4], [7]). In order to check the non-constraining relation, the observation graph of $(\widehat{N_i}, \widehat{m_i})$ is synchronized against the reachability graph of the interface component subnet $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$. However, the required synchronized product is widely reduced comparing to the general one. In fact, each reachable meta-state of $\langle \widehat{N_i}, \widehat{m_i} \rangle$ leads, by construction, to a unique reachable state of $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$. In other words, a meta-state is never synchronized with two different states of the interface component subnet. Obviously, the reciprocal doesn't hold: a state of $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$ could be synchronized with many meta-states of $\langle \widehat{N_i}, \widehat{m_i} \rangle$. Thus, in the worst case, the complexity of the non-constraining test is given by the number of reachable meta-states of $\langle \widehat{N_i}, \widehat{m_i} \rangle$ instead of (classically) the size of the synchronized product. The data structures used by Algorithm 3.2 are the following ones:

- a set H_{int} represents a heap to store the states of $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$ that are visited,
- a table *Synch* is used to associate a subset of meta-states (set of states) with each state of H_{int}. For any state s in H_{int}, we ensure that the meta-states of $Synch[s]$ are incomparable.

– a stack st, the items of which are tuples $\langle S, s, f\rangle$ composed of a meta-state of $\langle \widehat{N_i}, \widehat{m_i}\rangle$, a state of $\langle \widehat{N_{int}}, \widehat{m_{int}}\rangle$ and a set of interface transitions enabled from both nodes.

Algorithm 3.1 Non-constraining of $(\langle \widehat{N_i}, \widehat{m_i}\rangle$ w.r.t. $\langle \widehat{N_{int}}, \widehat{m_{int}}\rangle)$

state $s_{int} = \widehat{m_{int}}$, s'_{int};
Events f^i, f^{int}, $Obs = T_I$, $Unobs = \widehat{T_i} \setminus Obs$;
Set S_i, S'_i, H_{int}; **Set of Set** $Synch$;
stack $st(\langle Set, state, Events\rangle)$;
$S_i = Saturate(\{\widehat{m_i}\}, Unobs)$;
$f^{int} = firable(\{s_{int}\}, Obs)$; $f^i = firable(S_i, Obs)$;
if $(\neg(f^i \supseteq f^{int}))$ **then**
  return false
end if
$H_{int} = \{s_{int}\}$; $Synch[s_{int}] = \{S\}$;
st.**Push**$(\langle S_i, s_{int}, f^{int}\rangle)$;
repeat
  st.**Pop**$(\langle S_i, s_{int}, f^{int}\rangle)$;
  for $t \in f^{int}$ **do**
    $S'_i = Img(S_i, t)$; $S'_i =$ **Saturate**$(S'_i, Unobs)$;
    $s'_{int} = Img(\{s_{int}\}, t)$
    if $s'_{int} \notin H_{int}$ **then**
      $H_{int} = H_{int} \cup \{s'_{int}\}$; $Synch[s'_{int}] = \emptyset$
    end if
    if $\nexists\ S \in Sync[s'_{int}]$ s.t. $S \subseteq S'_i$ **then**
      for each $S \in Synch[s'_{int}]$ s.t. $S'_i \subseteq S$ **do**
        $Sync[s'_{int}] = Sync[s'_{int}] \setminus \{S\}$;
      end for
      $Sync[s'_{int}] = Sync[s'_{int}] \cup \{S'_i\}$;
      $f^i = firable(S'_i, Obs))$; $f^{int} = firable(s'_{int}, Obs)$
      if $(\neg(f^i \supseteq f^{int}))$ **then**
        return false
      end if
      st.**Push**$(\langle S'_i, s'_{int}, f^{int}\rangle)$;
    end if
  end for
until $st == \emptyset$;
return true

Three builder functions are used (all can be implemented symbolically using BDD notations) : $img(S, t)$ returns the immediate successors of the set of states S, through the firings of the transition t. $firable(S, o)$ is defined from a set of states S and a set of transitions o. It returns the subset of transitions in o that are enabled from a state of S (not necessarily the same). $saturate(S, u)$ returns a meta-state from a subset of states S and a set of (unobserved) transitions u.

Starting from each state of S, it infers all possible firings of the u transitions until a fix point is reached. The resulting meta-state consists of the states of S and all the reached states w.r.t. u.

The first stage of Algorithm 3.2 allows one to compute the first elements of the data structures, in particular the first items of the H_{int}, $Synch$ and st. For that, the initial meta-state S_i of $\langle \widehat{N_i}, \widehat{m_i} \rangle$ is computed. The transitions f^{int} enabled from the initial state s_{int} of $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$ are evaluated. If the enabled set of observable transitions from S_i doesn't contain the enabled set f^{int} the non-constraining test is stopped with a negative answer. Otherwise, the tuple $\langle S_i, s_{int}, f^{int} \rangle$ is pushed on the stack. Then, the algorithm iterates to synchronize successors items from an element of the stack, e.g. $\langle S_i, s_{int}, f^{int} \rangle$. According to each transition t in f^{int}, it computes and processes the successor s'_{int} from s_{int} in $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$ and the successor S'_i from S_i in $\langle \widehat{N_i}, \widehat{m_i} \rangle$.

In order to be efficient, we propose to decrease the number of (successors) tuples to be pushed on the stack. This is why we maintain the set $Sync[s'_{int}]$ of meta-states of $\langle \widehat{N_i}, \widehat{m_i} \rangle$ for each visited state s'_{int} of $\langle \widehat{N_{int}}, \widehat{m_{int}} \rangle$. Actually, a newly computed meta-state S'_i synchronized with s'_{int} can be discarded if a larger meta-state already exists in $Sync[s'_{int}]$. Otherwise, the meta-states in $Sync[s'_{int}]$, the set of states of which properly contains S'_i, are removed.

The algorithm will return false as soon as the sets of enabled interface transitions from both sides don't match with each other, meaning that the component subnet is constraining for the interface component subnet. Conversely, the algorithm will return true at the end of the synchronization product (the stack is empty), meaning that the component subnet does not constrain the interface component subnet.

3.3 Compositionality Results

With respect to the decomposition, we study two kinds of systems properties: generic properties like liveness and boundedness, and specific properties expressed by action-based temporal logics (logics using actions as atomic propositions) based on infinite observed sequences (sequences where some observed transitions occur infinitely often).

Preservation of generic properties. In this part, we prove that given a decomposable Petri net N, and under a mutual non-constraining condition between the corresponding *component subnets* $\widehat{N}_1$ and $\widehat{N}_2$, liveness (resp. boundedness) of N is completely characterized by the liveness (resp. boundedness) of $\widehat{N}_1$ and $\widehat{N}_2$.

Proposition 4. *If $\langle \widehat{N}_1, \widehat{m}_1 \rangle$ and $\langle \widehat{N}_2, \widehat{m}_2 \rangle$ are mutually non-constraining, then the following assertion holds:*
$\langle N, m \rangle$ is live $\Leftrightarrow$ $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and $\langle \widehat{N_2}, \widehat{m_2} \rangle$ are live.

Proof. ($\Rightarrow$) Assume that $\langle N, m \rangle$ is live. Let $\widehat{m_1}'$ be a reachable marking in $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and t a transition of $\widehat{T_1}$. Let us prove that there exists a sequence end-

ing by t which is enabled by the marking $\widehat{m_1}'$. Since $\langle \widehat{N_2}, \widehat{m_2} \rangle$ is non-constraining for $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and according to the proposition 2, there exists a sequence σ and a marking m' such that $m \xrightarrow{\sigma} m'$ and $\Phi_1(m') = \widehat{m_1}'$. On the other hand, since the marked net $\langle N, m \rangle$ is live, there exists a sequence σ' having t as the last transition and which is enabled by the marking m'. Let m'' be the marking reached by this sequence.

Let us now consider the sequence $\sigma\sigma'$, enabled by $\langle N, m \rangle$, according to the proposition 1, the projected sequence $\sigma\sigma'_{\lfloor \widehat{T_1}}$ is enabled by $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and the marking reached is equal to $\Phi_1(m'')$. We conclude that $\sigma'_{\lfloor \widehat{T_1}}$ (having t as a last transition) is enabled by $\widehat{m_1}'$.

By symmetry, we prove that $\langle \widehat{N_2}, \widehat{m_2} \rangle$ is live.

($\Leftarrow$) Assume that que $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and $\langle \widehat{N_2}, \widehat{m_2} \rangle$ are mutually non-constraining and live. Let m' be a reachable marking in $\langle N, m \rangle$ and t a transition of T, let us prove that there exists a sequence, having t as the last transition, which is enabled by marking m'.

Due to the symmetry of the problem, we assume that the transition t belongs to $\widehat{T_1}$. According to proposition 1, there exists a sequence σ_1 which is enabled by $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and leading to the marking $\widehat{m'_1} = m'_{\lfloor \widehat{T_1}}$. On the other hand, there exists a sequence σ'_1 having t as the last transition and which is enabled by m'_1 (because $\langle \widehat{N_1}, \widehat{m_1} \rangle$ is live). Moreover, since $\langle \widehat{N_2}, \widehat{m_2} \rangle$ is non-constraining for $\langle \widehat{N_1}, \widehat{m_1} \rangle$, we deduce the existence of a sequence σ', which is enabled by $\langle N, m' \rangle$, such that $\sigma'_{\lfloor \widehat{T_1}} = \sigma'_1$. The sequence satisfying the former condition has t as the last transition. Thus, we deduce that $\langle N, m \rangle$ is live.

Proposition 5. *Let $\langle N, m \rangle$ be a Petri net and let $N_d = \langle N_1, T_I, N_2 \rangle$ be a decomposition of N leading to the component subnets $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and $\langle \widehat{N_2}, \widehat{m_2} \rangle$. If $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and $\langle \widehat{N_2}, \widehat{m_2} \rangle$ are mutually non-constraining, then:*
$\langle N, m \rangle$ is bounded $\Leftrightarrow$ $\langle \widehat{N_1}, \widehat{m_1} \rangle$ and $\langle \widehat{N_2}, \widehat{m_2} \rangle$ are bounded.

Proof. $\Leftarrow$ (Corollary 1)
$\Rightarrow$ Assume that $\langle N, m \rangle$ is bounded. We suppose that $\langle \widehat{N_i}, \widehat{m_i} \rangle$ is unbounded, for $i \in \{1, 2\}$. This means that there exists a run (an infinite computation) $\xi_i = \widehat{m_i}^0 \xrightarrow{t_1^i} \widehat{m_i}^1 \xrightarrow{t_2^i} \dots$ and a place $p \in \widehat{P_i}$ such that $\forall\ \widehat{m_i}^k \in \xi_i$, $\exists\ \widehat{m_i}^l \in \xi_i$ with $l > k$ s.t. $\widehat{m_i}^k(p) < \widehat{m_i}^l(p)$. Since $\langle \widehat{N_j}, \widehat{m_j} \rangle$ ($j \in \{1, 2\}$, $j \neq i$) doesn't constrain $\langle \widehat{N_i}, \widehat{m_i} \rangle$, there exists a run $\xi = m_0 \xrightarrow{\sigma_1} m_1 \xrightarrow{\sigma_2} \dots$ in $\langle N, m \rangle$ s.t. $\xi_{\lfloor \widehat{T_i}} = \xi_i$ and $\forall k = 1, 2, \dots\ \widehat{m_i}^k = \Phi_i(m_k)$.

Case 1: $p \notin A_j$
In this case, $p \in P$ and $\forall\ m_k \in \xi$, $\exists\ m_l \in \xi$ with $l > k$ s.t. $m_k(p) < m_l(p)$ which means that $\langle N, m \rangle$ is unbounded and then contradicts the hypothesis.
Case 2: $p \in A_j$, let v be the corresponding positive invariant.
In this case, $\forall\ m_k \in \xi$, $\exists\ m_l \in \xi$ with $l > k$ s.t. $v^T.m_k < v^T.m_l$. This means that there exists, at least, one place in $||v||$ for which the marking can be infinitely

increased through ξ (v is a positive invariant). This contradicts the hypothesis that $\langle N, m\rangle$ is bounded.

Preservation of action-based temporal properties Let us consider a Petri net N and an action-based temporal logic formula f relative to the infinite observed sequences of N. Assuming that the transitions occurring in f belong to one component subnet, we show how to exploit our approach in order to modularly check f. First, the following proposition states that, checking if a particular transition $t \in \widehat{T_1}$ (for instance) appears infinitely often in a firing sequence of a decomposable Petri net $\langle N, m\rangle$ is reduced to the analysis of firing sequences of $(\widehat{N_1}, \widehat{m_1})$ if it is not constrained by $(\widehat{N_2}, \widehat{m_2})$.

Proposition 6. *Let $N_d = \langle N_1, T_I, N_2\rangle$ be a decomposition of a Petri net $\langle N, m\rangle$ leading to the component subnets $\langle \widehat{N_1}, \widehat{m_1}\rangle$ and $\langle \widehat{N_2}, \widehat{m_2}\rangle$. Let t be a transition in $\widehat{T_1}$. If $\langle \widehat{N_2}, \widehat{m_2}\rangle$ is non-constraining for $\langle \widehat{N_1}, \widehat{m_1}\rangle$, then:*
$\exists \sigma \in L^\omega(N, m)$ *s.t.* $t \in inf(\sigma) \iff \exists \sigma_1 \in L^\omega(\widehat{N_1}, \widehat{m_1})$ *s.t.* $t \in inf(\sigma_1)$

Proof. ($\Longrightarrow$) Let $\sigma \in L^\omega(N, m)$ a firing sequence in $\langle N, m\rangle$ such that $t \in inf(\sigma)$ and $\sigma_1 = \sigma_{\lfloor \widehat{T_1}}$. Following the proposition 1, we deduce that $\widehat{m_1} \xrightarrow{\sigma_1}$. Because $t \in \widehat{T_1}$, one concludes that $t \in inf(\sigma_1)$.
($\Longleftarrow$) Let $\sigma_1 \in L^\omega(\widehat{N_1}, \widehat{m_1})$ such that $\widehat{m_1} \xrightarrow{\sigma_1} \widehat{m_1}'$ and $t \in inf(\sigma_1)$. According to proposition 2, there exists a firing sequence $\sigma \in L^\omega(N, m)$ such that $\sigma_1 = \sigma_{\lfloor \widehat{T_1}}$. Since $t \in inf(\sigma_1)$, one deduces that $t \in inf(\sigma)$.

Proposition 6 leads to a modular model checking approach dealing with infinite observed sequences. Given a decomposable Petri net N and a formula f such that the set *Occ(f)* (set of transitions occurring in f) is a subset of $\widehat{T_i}$ ($i \in \{1, 2\}$), checking f on N can be reduced to check f on $\widehat{N_i}$ in the two following cases:

- f holds on $\widehat{N_i}$,
- f doesn't hold on $\widehat{N_i}$ and $\widehat{N_i}$ is non constrained by $\widehat{N_j}$.

4 Conclusion

In this paper, we have presented a decomposition approach which allows the modular verification of Petri nets properties. The liveness and boundedness of the system components can be used to check these properties for the system. This is also the case for any linear time property whenever its checking relates to the infinite observed executions of some component of the system. In contrast to previous techniques, we do not force any specific (restrictive) structure at the interface of the modules, but we exploit the linear invariants (that are usually common in well-specified system models) of the system. Our main contribution is the definition of a sufficient condition to test the non-constraining relation w.r.t. a component subnet. In order to be general, it is tested behaviourally but modularly with respect to a component subnet.

The limit of the presented work occurs when the non-constraining relationis required for a component subnet but does not hold. Concerning LTL properties, our first solution is an iterative technique presented in [3]. Starting from the smallest component subnet to check the satisfaction of an LTL property, it automatically enlarges the component subnet whenever the property is detected false and the environment is constraining.

From a practical point of view, we have shown how the observation-basedapproach presented in [4] can be adapted to reduce the representation of the reachability graph of a component subnet (using symbolic decision diagram techniques) leading to an efficient modular test of the non-constraining relation. We are currently developping a tool in order to test our method on real case studies.

References

1. Mohamed-Lyes Benalycherif and Claude Girault. Behavioural and structural composition rules preserving liveness by synchronisation for colored FIFO nets. In *Lecture Notes in Computer Science; Proc. 17th International Conference in Application and Theory of Petri Nets (ICATPN'96), Osaka, Japan*, volume 1091, pages 73–92. Springer-Verlag, June 1996.
2. S. Christensen and L. Petrucci. Modular analysis of petri nets. *Computer Journal*, 43(3):224–242, 2000.
3. S. Haddad, J.-M. Ilié, and K. Klai. An incremental verification technique using decomposition of petri net. In *proc. of the IEEE SMC'02 - Systems, Man and Cybernetics, Hammamet, Tunisia*, 2002.
4. S. Haddad, J.-M. Ilié, and K. Klai. Design and evaluation of a symbolic and abstraction-based model checker. In *Proc. of Automated Technology for Verification and Analysis: Second International Conference, ATVA 2004, Taipei, Taiwan, ROC, October 31-November 3*, 2004.
5. J.-M. Ilié, K. Klai, and B. Zouari. A modular verification methodology for d-nri petri nets. In *in proc. of the International Conference ACS/IEEE 2003 on Computer Systems and Applications (AICCSA-03), Tunis, Tunisia*, pages 14–18, 2003.
6. T. Latvala and M. Makela. Ltl model checking for modular Petri nets. In*in proc. of ICATPN'04*, pages 298–311, 2004.
7. V. Noord. Treatment of epsilon moves in subset construction. In *Computational Linguistics, MIT Press for the Association for Computational Linguistics*, volume 26. 2000.
8. A. Santone. Compositionality for Improving Model Checking. In *In proc. of FORTE'00*, in proc. of Formal Methods for Distributed System Development, October 2000.
9. C. Sibertin-Blanc. A client-server protocol for composition of Petri nets. In*in proc. of ICATPN'93*, LNCS, June 1993.
10. Y. Souissi and G. Memmi. Compositions of nets via a communication medium. *LNCS*, 483:457–470, 1991.
11. A. Valmari. Compositional state space generation. In *in proc. of ICATPN'90*, LNCS, May 1990.