Syntactical Colored Petri Nets Reductions

S. Evangelista[1], S. Haddad[2], and J.-F. Pradat-Peyre[1]

[1] CEDRIC - CNAM Paris, 292, rue St Martin, 75003 Paris
[2] LAMSADE-CNRS UMR 7024 Université Paris-Dauphine,
Place du Maréchal de Lattre de Tassigny, 75775, Paris Cedex 16

Abstract. In this paper, we develop a syntactical version of elaborated reductions for high-level Petri nets. These reductions simplify the model by merging some sequential transitions into an atomic one. Their conditions combine local structural ones (e.g. related to the actions of a thread) and global algebraic ones (e.g. related to the threads synchronization). We show that these conditions are performed in a syntactical way, when a syntax of the color mappings is given. We show also how our method outperforms previous ones on a recent case study with regard both to the reduction ratio and the automatization of their application.

1 Introduction

The concurrent programming paradigm is a powerful tool for the implementation of complex software. However it may lead to applications where the interaction between threads or processes produces subtle behaviours that are difficult to predict. In this context, it is necessary to include in the application development life cycle a complete and systematic verification step.

Two kinds of verification techniques are usually performed: state enumeration based methods and structural methods. The state enumeration based methods lead to a complete verification of the modeled system but the analysis is restricted by the combinatory explosion factor (i.e. the number of control states may grow exponentially w.r.t. the number of threads and the size of the application). The structural methods are generally efficient but they do not ensure the complete correctness of the system. Thus an attractive trade-off is to combine both methods.

In this context, an efficient strategy is to examine the structure of the model for reducing the number of execution traces that are to be analyzed. The obtained reduction ratio depends on the kind of considered properties. The more specific are the properties, the greater is the reduction.

Again, two distinct approaches can be followed to obtain such a reduction. On the one hand, it's possible to apply on-the-fly techniques when building the state graph. These techniques are based on the detection that in a given state,

– some enabled actions may be forgotten since they lead to an already visited state [GW93],
– some enabled actions may be safely delayed [Val93],
– some enabled actions may be executed simultaneously [VM97].

On the other hand, one can work at the model level in order to simplify it before building a reduced state graph. In this framework, a frequent approach is the merging of consecutive statements into a virtual atomic one whose effect is the composition of the effects of these statements. Such a transformation presents the following advantages:

- the combinatory explosion is drastically reduced by the elimination of the intermediate states,
- the induced overhead computation is negligible w.r.t. the cost of the state graph building,
- this abstraction is potentially applicable to "parameterized" programs (e.g. independent of the number of instances of a process class) and **this feature is not covered by the on-the-fly techniques**.

We have chosen to explore this approach by proposing new colored Petri nets reductions that simplify the model by merging some sequential transitions [EHPP04]. These reductions enlarge earlier ones by weakening application conditions but also by defining precisely which conditions are sufficient to preserve some specific properties (i.e. liveness and linear temporal formulae defined on maximal or infinite sequences) We show here, that given a syntax for colored net, application conditions and transformation rules of these reductions can performed with only fully automatic syntactical operations.

The paper is organized as follows. In Section 2 we recall the definition of colored nets and the definition of the colored post-agglomeration. Section 3 shows how to define syntactical conditions enabling the manipulation of colored mapping with a well chosen syntax for colored Petri nets. Section 4 highlights the interest of our reductions by applying them on a recently published case study. Before concluding, we present in Section 5 related works.

2 Colored Petri Nets and Agglomerations

We assume that the reader is familiar with Petri nets and usual mathematics notions such as multisets or powersets. We denote by $Bag(S) = \mathbb{N}^S$ the set of multisets over the finite set S, and by $\mathcal{P}(S) = \{true, false\}^S$ the set of powersets over S.

We first give in this Section some definitions related to colored Petri nets. Then, we recall some definitions concerning color mappings properties and handling. At last, we detail the application conditions of the post-agglomeration. For space constraint, the pre-agglomeration will not be mentioned in this paper. The pre-agglomeration is treated in [EHPP04].

2.1 Colored Petri Nets

Definition 1. *A colored Petri net (CPN for short) is a tuple $N = \langle P, T, \Sigma, C, W^-, W^+ \rangle$ where P is a finite set of* **places***; T is a finite set of* **transitions***, with $P \cap T = \emptyset$; Σ, the* **colors set***, is a finite set of finite and non empty sets; C, the* **color domain** *application, is a mapping from $P \cup T$ to Σ; W^- and W^+, the* **backward and forward incidence matrixes** *associate to each $(p,t) \in P \times T$ a color mapping from $C(t)$ to $Bag(C(p))$.*

A couple (e,c) with $e \in P \cup T$ and $c \in C(e)$ is called an instance of e. In the remainder of the paper mappings from C to $Bag(C')$ will be extended to mappings from $Bag(C)$ to $Bag(C')$ by the two following rules : $f(\lambda.c) = \lambda.f(c)$ and $f(c_1+c_2) = f(c_1)+f(c_2)$. Given a place p, the sets $^\bullet p$ and $p^\bullet$ are defined as usual as $\{t \in T | W^+(p,t) \neq 0\}$ and $\{t \in T | W^-(p,t) \neq 0\}$. The same notations for a transition can be given in a straightforward way. A marking of a CPN associates to each of its places a multi-set over its color domain. The firing rule defines the dynamic of the net.

Definition 2. *Let $N = \langle P,T,\Sigma,C,W^-,W^+,m_0\rangle$ be a CPN. A* **marking** *of N is a mapping which associates each $p \in P$ to an element of $Bag(C(p))$. The set of markings of a net N is denoted by $\mathbb{M}_N$. A* **colored marked net** *is a couple $\langle N,m_0\rangle$ with N a CPN and $m_0 \in \mathbb{M}_N$ the initial marking of the net.*

Definition 3. *Let $N = \langle P,T,\Sigma,C,W^-,W^+\rangle$ be a CPN, $t \in T$, $c_t \in C(t)$, and $m \in \mathbb{M}_N$. The instance (t,c_t) is* **firable** *at m, denoted by $m[(t,c_t)\rangle$, if and only if $\forall p \in P, m(p) \geq W^-(p,t)(c_t)$. The* **firing** *of (t,c_t) at m leads to the marking m', denoted by $m[(t,c_t)\rangle m'$, defined by: $\forall p \in P, m'(p) = m(p) + W(p,t)(c_t)$. The* **reachability set** *of $\langle N,m_0\rangle$, denoted by $Reach(N,m_0)$, is the set $\{m_0\} \cup \{m \in \mathbb{M}_N | \exists m' \in Reach(N,m_0), t \in T, c_t \in C(t) | m'[(t,c_t)\rangle m\}$.*

Application conditions of agglomerations rely on the existence on some flows that induce invariants.

Definition 4. *A* **colored flow** *$\mathcal{F}$, on the color domain $C_{\mathcal{F}}$, is a vector over P, denoted by the formal sum $\mathcal{F} = \sum_{p\in P} \lambda_p.\mathcal{F}_p.p$, where $\forall p \in P$, $\lambda_p \in \mathbb{Z}$ and $\mathcal{F}_p$ is a mapping from $Bag(C(p))$ to $Bag(C_{\mathcal{F}})$ such that: $\forall t \in T, \sum_{p\in P} \lambda_p.\mathcal{F}_p \circ W(p,t) = 0$ [1]. The colored flow $\mathcal{F}$ is* **positive** *if $\forall p \in P$, $\lambda_p \geq 0$.*

Definition 5. *A colored flow $\mathcal{F}$, on the domain $C_{\mathcal{F}}$, induces the invariant:*
$\forall m \in Reach(N,m_0), \sum_{p\in P} \lambda_p.\mathcal{F}_p(m(p)) = \sum_{p\in P} \lambda_p.\mathcal{F}_p(m_0(p))$. This invariant is a **binary** *invariant if $\forall c \in C_{\mathcal{F}}, \sum_{p\in P} \lambda_p.\mathcal{F}_p(m_0(p))(c) = 1$; a* **synchronization** *invariant if $\forall c \in C_{\mathcal{F}}, \sum_{p\in P} \lambda_p.\mathcal{F}_p(m_0(p))(c) = 0$. When no confusion is possible (i.e. the initial marking is given), we will not distinguish a colored flow and its induced invariant.*

For instance, given the right model of figure 1(b) we can automatically compute the positive flow (on the domain ε) $\mathcal{F} = mb + \langle All_C\rangle.q2$ which induces the binary invariant $\forall m \in Acc(N,m_0)$, $m(mb) + \sum_{c\in C} m(q2(c)) = 1$. This invariant ensures that when place mb is marked then place $q2$ is not, and then, that transition $d1d2$ is not fireable when $q2$ is marked.

2.2 Color Mappings Properties and Handling

Reductions techniques for Petri nets are characterized by : (1) some application conditions, (2) the characterization of the reduced net, (3) the properties preserved. For ordinary Petri nets, the application conditions rely on the structure of the net, i.e., the physical links between the places and transitions of the net, and on algebraic conditions

[1] 0 denotes here the null mapping from $C(t)$ to $Bag(C_{\mathcal{F}})$.

given by the invariants of the net. When reasoning with colored nets, the color mappings also have to be considered since the structure of the colored net does not necessarily reflect the structure of the underlying ordinary Petri net. There is then two possibilities : unfold the net, apply reductions on the unfolded net and fold it back; or define conditions on the colored net that ensure correct ordinary agglomerations in the underlying net. Thus operations on natural numbers, e.g., $W^-(p,t).W^+(q,t) > 0$ becomes operations on color mappings, e.g., $W^-(p,t) \circ {}^t(W^+(q,t)) \neq 0_{Bag(C(q)) \rightarrow Bag(C(p))}$.

Firstly, we define a set of properties that are used in agglomerations definition. These properties can give some precious hints on the structure on the underlying net. For instance, a quasi-one-to-one mapping that labels an arc between a place p and a transition t, implies that two different instances of t cannot be linked to the same instance of p.

Definition 6. *Let f be a mapping from C to $Bag(C')$. f is*

- **unitary** *when* $\forall c \in C, f(c)(c') \leq 1$
- **orthonormal** *when* $\forall c \in C \exists c' \in C'$ *such that* $f(c)(c') = 1$ *and* $\forall c' \in C' \exists c \in C$ *such that* $f(c)(c') = 1$
- **ortho-projection** *when* $f = g \circ h$ *with h an orthonormal mapping from C to $Bag(C)$ and g an orthonormal mapping from C to $Bag(C')$*
- **quasi-one-to-one** *when* $\forall c_1, c_2 \in C, c' \in C', f(c_1)(c') = 0 \vee f(c_2)(c') = 0$
- **quasi-onto** *when* $\forall c' \in C', \exists c \in C$ *such that* $f(c)(c') > 0$

Secondly, a frequent need is to symbolically follow a path in the underlying net. That can be achieved by using the transposition and composition operators. The transposition is used to find the instances of a place linked to a transition instance, e.g., ${}^tW^+(p,h)$. The composition enables to find, for example, the instances of a transition linked to another one by an intermediary place instance, e.g., ${}^tW^+(p,h) \circ W^-(p,f)$.

Definition 7 (Transposition and composition). *If f is a mapping from $Bag(C'')$ to $Bag(C')$, and g is a mapping from $Bag(C)$ to $Bag(C'')$ then $f \circ g$ is a mapping from $Bag(C)$ to $Bag(C')$ defined by $\forall c \in C, c' \in C', (f \circ g)(c)(c') = \sum_{c'' \in C''} f(c'')(c') \cdot g(c)(c'')$. If h is a mapping from $Bag(C)$ to $Bag(C')$, then th is a mapping from $Bag(C')$ to $Bag(C)$ defined by $\forall c \in C, c' \in C', {}^th(c')(c) = h(c)(c')$.*

At last, we are usually not interested in the exact numbers of tokens produced in a place, but rather in the fact that tokens are actually produced. The $\overline{f}$ operation can be used for this purpose.

Definition 8. *Let $f \in S \rightarrow Bag(S')$. $\overline{f} \in S \rightarrow \mathcal{P}(S')$ is defined by: $\forall s \in S, \overline{f}(s) = \{s' \in S' \mid f(s)(s') > 0\}$.*

It is also useful to check if the image of a mapping f from $\mathcal{P}(C)$ to $\mathcal{P}(C')$ is included in the image of a mapping g from $\mathcal{P}(C)$ to $\mathcal{P}(C')$; we denote this by $f \sqsubseteq g$ and it is defined by $f \sqsubseteq g$ iff $\forall c \in C, \overline{f}(c) \subseteq \overline{g}(c)$.

2.3 The Post-Agglomeration Reduction

We recall now application conditions of the post-agglomeration. The transformation rule and the definition of the other transitions agglomeration (the pre-agglomeration) are presented in [EHPP04].

The basic hypothesis of the post-agglomeration is that the set of transitions of the net is partitioned as : $T = T_0 \uplus H \uplus F$. The underlying idea of this decomposition is that transitions of H and transitions of F are causally dependent : an occurrence of $f \in F$ in a sequence of firings may always be related to a previous occurrence of some $h \in H$ in this sequence. Thus, in the reduced net, one fires f immediately after the firing of some $h \in H$.

The definition of the net obtained by the agglomeration of H with F is straightforward. Thus for space constraint we will not give it in this paper. For the same reasons, we will not detail the properties preserved by the post-agglomeration. We simply recall that if all the mentioned conditions are verified by the net, both the reduced and the original net will be equivalent in terms of Petri net liveness and languages of maximal and infinite sequences which do not observe transitions of F. Indeed, since the idea of the post-agglomeration is to advance the firing of any transition $f \in F$, the sequence remains equivalent with respect to transitions which are not in F. The curious readers may find these additional informations in [EHPP04].

The post-agglomeration is based on four hypotheses : the **potential agglomerability**, the F**-independence**, the F**-continuation** and the HF**-interchangeability**.

Firstly we impose that the net is **potentially agglomerable**. This hypothesis ensures that a transition $f \in F$ in a sequence of firings is related to a previous occurrence of some $h \in H$ in this sequence. The first point ensures that the place p models an intermediate state between the firing of a transition in H and the firing of a transition in F. The second one ensures that the firing of a transition h produces only one token in place p (conditions on H) and that two different tokens in place p cannot be consumed by a same firing of f (conditions on F).

Definition 9. *A colored net is **potentially agglomerable** (p-agglomerable) if $\exists p \in P$ such that*

1. $^\bullet p = H$*,* $p^\bullet = F$ *and* $m_0(p) = 0$*;*
2. $\forall f \in F$*,* $C(f) = C(p) \times C_f$*,* $W^-(p,f)$ *is an ortho-projection from* $C(f)$ *to* $C(p)$ *and* $\forall h \in H$*,* $C(h) = C(p)$*,* $W^+(p,h)$ *is orthonormal*

The F**-independence** hypothesis ensures that when the place p is marked, no transition that can produce tokens useful for the firing of a transition in F can be fired. Given $c \in C(p)$, $\phi(c)$ is exactly the set of firing instances of t producing tokens useful for the firing of (f,c). Similarly, given $c \in C(p)$, $\psi(c)$ is exactly the set of firing instances of t which can not be fired when a token colored by c is present in p. Additionally, the strong independence ensures that the place p is safe, i.e. there is at most one token per color present in p.

Definition 10. *A p-agglomerable colored net is F**-independent** if $\forall f \in F$, $\forall q \in ({}^\bullet f \setminus \{p\})$, $\forall t \in {}^\bullet q \setminus F$, $\exists p_t \in {}^\bullet t$, $\exists \mathcal{F} = \sum_{r \in P} \mathcal{F}_r.r$ a binary positive flow on a domain D such that if $\phi = \overline{{}^tW^+(q,t) \circ W^-(q,f) \circ {}^tW^-(p,f)}$ and $\psi = \overline{{}^tW^-(p_t,t) \circ {}^t\mathcal{F}_{p_t} \circ \mathcal{F}_p}$ then $\phi \sqsubseteq \psi$. Furthermore, if there exists a binary positive invariant $\mathcal{F}'$ on the domain $C(p)$ such that ${}^t\mathcal{F}'_p$ is a quasi-onto mapping then the net is* **strongly** *F-independent.*

The F**-continuation** hypothesis means that an excess of occurrences of $h \in H$ can always be reduced by subsequent firings of transitions of F, i.e. when the place p is marked, a transition of F is necessarily fireable.

Definition 11. *A p-agglomerable colored net is F-**continuable** if $\exists f \in F$ such that $^\bullet f = \{p\}$* **or** *$\exists\ F_s \subset F$ such that:*

1. $\forall f \in F_s, \exists p_f \neq p \in P$, $^\bullet f = \{p, p_f\}$,
2. $\forall f \in F_s$, ${}^tW^-(p_f, f)$ *is an unitary quasi-one-to-one mapping;*
3. *there exists a flow on $C(p)$, $\mathcal{F} = \sum_{f \in F_s} \mathcal{F}_{p_f}.p_f - \lambda.X_{C(p)}.p$ with* $\forall f \in F_s$, $\overline{{}^t\mathcal{F}_{p_f}} \sqsubseteq \overline{W^-(p_f, f) \circ {}^tW^-(p, f)}$ *and such that*
   (a) *either $\lambda = 0$ and $\mathcal{F}$ induces a binary positive invariant*
   (b) *or $\lambda = 1$ and $\mathcal{F}$ induces a synchronization invariant*

At last, the HF**-interchangeability** hypothesis mainly restricts either the set H or F to be a singleton in order to avoid the case where $h \in H$ and $f \in F$ are live in the original net whereas the transition hf is not live in the reduced net.

Definition 12. *A p-agglomerable colored net is HF-**interchangeable** if either $H = \{h\}$ or $F = \{f\}$, $C(f) = C(p)$ (thus $W^-(p, f)$ is orthonormal since it is p-agglomerable)*

3 Syntactical Rules for Agglomerations Implementation

Computing the transposition of a color mapping, or the composition of two color mappings is impossible for general colored Petri nets, i.e., with unstructured color domains or color mappings. In order to enable to perform such computations in a symbolic way without unfolding the net, we first define in this section a restricted class of colored Petri nets with well defined color domains and mappings. In the second and third part of the section, we will see that this class allows us to check basic mapping properties such as orthonormality in a straightforward way, and to perform operations on color mappings in a syntactic manner.

3.1 Quasi Well Formed Colored Nets (QWNs)

Quasi well formed nets are a restriction of the well-known well formed nets class. QWNs are characterized by a good structuring of color domains and mappings.

At first, to such a net is associated a set of finite color classes (e.g. a set of processes) that will be denoted $Cl = \{C_1, \ldots, \mathbf{C}_N\}$. The sizes of these classes are the parameters of the net (denoted n_i for C_i). Each class can be enumerated starting from any color with the help of a successor mapping *succ*, i.e. $\forall c \in C_i$, $C_i = \{c, succ(c), \ldots, succ^{n_i - 1}(c)\}$.

A color domain C is a cartesian product of color classes. For the sake of simplicity, we assume that these domains are built upon the order of Cl, i.e., each color domain C can be written as $C = C_1 \times \cdots \times C_1 \times \cdots \times C_N \times \cdots \times C_N$. Since we allow repeated occurrences of a class, e_i denotes the number of occurrences of C_i in C.

Color mappings are built using simpler mappings called elementary mappings. Four kinds of elementary mappings are allowed: the projection, the successor (or predecessor), the constant mapping, and the broadcast mapping.

Definition 13. *Let C be a color domain, and C_i be a color class. The set of* **elementary color mappings** *from C to $Bag(C_i)$ is the set*
$\{X_i^j\}_{j \in [1..e_i]} \cup \{X_i^j \oplus n\}_{j \in [1..e_i], n \in \mathbb{N}} \cup \{All_i\} \cup \{\mathbb{1}_{c_i}\}_{c_i \in C_i}$

It is defined by $\forall c = \langle c_1^1, \ldots, c_N^{e_N} \rangle \in C$:

- $X_i^j(c) = \{c_i^j\}$ (*a* **projection** *mapping*)
- $X_i^j \oplus n(c) = X_i^j \ominus -n(c) = \{succ^n(c_i^j)\}$ (*a* **successor** *mapping*)
- $All_i(c) = \sum_{c_i \in C_i} c_i$ (*the* **broadcast** *mapping*)
- $\mathbb{1}_{c_i}(c) = \{c_i\}$ (*a* **constant** *mapping*)

Definition 14. *Let C be a color domain and* $c = \langle c_1^1, \ldots, c_N^{e_N} \rangle \in C$ *then an* **elementary guard** *G on C, a mapping from C to* $\mathbb{B} = \{true, false\}$, *is:*

- *either* $(X_i^j = X_i^{j'} \oplus n)$ *defined by* $(X_i^j = X_i^{j'} \oplus n)(c) = (c_i^j = succ^n(c_i^{j'}))$,
- *or* $(X_i^j = \mathbb{1}_{c_i})$ *with* $c_i \in C_i$ *defined by* $(X_i^j = \mathbb{1}_{c_i})(c) = (c_i^j = c_i)$

A general guard is a boolean combination of elementary guards.

The general syntax of color mappings is based on elementary mappings and guards with the help of three constructors: the tuple constructor, the product of a tuple by a scalar and the sum of tuples.

Definition 15. *Let C and C′ be two color domains. The syntax of a* **QWN color mapping** *f from C to* $Bag(C')$ *is:*
$f = \sum_{k=1}^{K} \alpha_k.[G_k]\langle f_{1,k}^1, \ldots, f_{N,k}^{e'_N} \rangle$ *with* $\forall k \in [1..K], i \in [1..N], j \in [1..e'_i]$, $\alpha_k > 0$, G_k *a guard on C and* $f_{i,k}^j$ *is an elementary color mapping from C to* $Bag(C_i)$. *It is defined by:*

$$f(c) = \sum_{k=1}^{K} \alpha_k \sum_{\{c \in C \mid G_k(c)\}} \langle f_{1,k}^1, \ldots, f_{N,k}^{e'_N} \rangle(c)$$

3.2 Checking Color Mappings Properties

The good structuring of quasi well formed nets allows us to easily check color mapping properties. The straightforward proof of these propositions can be found in [Eva04]. In all the propositions, we consider a quasi well formed color mapping f from $Bag(C)$ to $Bag(C')$.

A simple condition to ensure that f is unitary is to impose that it is composed of a single tuple of which valuation is 1.

Proposition 1. *If* $f = [G]\langle f_1^1, \ldots, f_N^{e'_N} \rangle$ *then f is* **unitary**.

To ensure orthonormality, we must have $C = C'$, f composed of a single non guarded tuple and each variable of the transition must appear in this tuple.

Proposition 2. *If* $C = C'$, $f = \langle f_1^1, \ldots, f_N^{e'_N} \rangle$ *and* $\forall i \in [1..N], j \in [1..e_i], \exists j' \in [1..e_i], n \in \mathbb{N}$ *such that* $f_i^j = X_i^{j'} \oplus n$ *then f is* **orthonormal**.

If f is a single non guarded tuple in which only variables appear and such that the same variable can not appear at two different positions in the tuple then it is an ortho-projection.

Proposition 3. *If* $f = \langle f_1^1, \ldots, f_N^{e'_N} \rangle$ *and* $\forall i \in [1..N], j \in [1..e'_i], f_i^j = X_i^{j'} \oplus n$ *and* $\nexists j'' \in [1..e'_i]$ *such that* $f_i^{j''} = X_i^{j'} \oplus m$ *then f is an* **ortho-projection**.

A single tuple in which all the variables of the transition appear is quasi-one-to-one.

Proposition 4. *If $f = \alpha.[G]\langle f_1^1,\ldots,f_N^{e'_N}\rangle$ and $\forall i \in [1..N], j \in [1..e_i], \exists j' \in [1..e'_i], n \in \mathbb{N}$ such that $f_i^{j'} = X_i^j \oplus n$ then f is* **quasi-one-to-one**.

At last, f is quasi-onto if there is a non guarded tuple in it which is such that no constant appear in it and no variable can appear at two different positions in it.

Proposition 5. *If $f = \alpha.\langle f_1^1,\ldots,f_N^{e'_N}\rangle + g$ such that these two conditions are fulfilled*

1. $\forall i \in [1..N], j \in [1..e'_i], f_i^j = All_i$ *or* $f_i^j = X_i^{j'} \oplus n$
2. $\nexists i \in [1..N], j \in [1..e'_i], j' \in [1..e'_i]$ *such that* $f_i^j = X_i^{j''} \oplus n$ *and* $f_i^{j'} = X_i^{j''} \oplus m$

then f is **quasi-onto**.

3.3 Computing Structural Relations

Syntactic Transposition. We now focus on the specification of a symbolic transposition. At first, we notice that the transpose of a linear combination of mappings is the linear combination of the transposes. Thus we restrict ourselves to a guarded tuple. We now focus on the guard. We remark that if $f = [G]\langle f_1^1,\ldots,f_N^{e'_N}\rangle$ then f can be viewed as the following composition: $f = \langle f_1^1,\ldots,f_N^{e'_N}\rangle \circ [G]\langle X_1^1,\ldots,X_1^{e_1},\ldots,X_N^1,\ldots,X_N^{e_N}\rangle$. Thus ${}^tf = {}^t[G]\langle X_1^1,\ldots,X_1^{e_1},\ldots,X_N^1,\ldots,X_N^{e_N}\rangle \circ {}^t\langle f_1^1,\ldots,f_N^{e'_N}\rangle$. By a straightforward evaluation, one remarks that:

$$ {}^t[G]\langle X_1^1,\ldots,X_1^{e_1},\ldots,X_N^1,\ldots,X_N^{e_N}\rangle = [G]\langle X_1^1,\ldots,X_1^{e_1},\ldots,X_N^1,\ldots,X_N^{e_N}\rangle $$

Hence supposing that the composition can be handled, we restrict the symbolic transposition to non guarded tuples.

We make a new observation. Let g be a mapping from C to C' and h be a mapping from D to D'. Suppose that f is a mapping from $C \times D$ to $C' \times D'$ defined by $f(c,d) = \langle g(c), h(d)\rangle$. Then ${}^tf(c',d') = \langle {}^tg(c'), {}^th(d')\rangle$. As it is the case for QWN mappings (that can be viewed as tuple of mappings from C^{e_i} to $C^{e'_i}$), we can restrict ourselves to mappings where a single class occurs (with possible repetitions) in its domain and its codomain.

Proposition 6 (Symbolic transposition). *Let $C = C_i^e$ and $C' = C_i^{e'}$ be two color domains. Since we deal with a single class C_i, we omit in the sequel the subscript i.*

Let $f = \langle f^1,\ldots,f^{e'}\rangle$ be a mapping from $Bag(C)$ to $Bag(C')$ with $f^{k'}$ being either All, $X^{\mu(k')} \oplus m_{k'}$ or $\mathbb{1}_n$. The transposition of f is defined by:

$$ {}^tf = [\bigwedge_{k=1}^{e} G_k \bigwedge_{k'=1}^{e'} H_{k'}]\langle g^1,\ldots,g^e\rangle] $$

where:

- **if** $\mu^{-1}(k) = \emptyset$ **then** $g^k = All$ *and* $G_k = true$
- **else** $(\mu^{-1}(k) = \{j_1,\ldots,j_q\})$ $g^k = X^{j_1} \ominus m_{j_1}$ *and* $G_k = \bigwedge_{k'=2}^{q}(X^{j_{k'}} \ominus n_{j_{k'}} = X^{j_1} \ominus m_{j_1})$

and

- **if** $f^{k'} = \mathbb{1}_n$ **then** $H_{k'} = (X^{k'} = \mathbb{1}_n)$
- **else** $H_{k'} = true$

Example 1. Let us compute the transpose of the mapping f from $Bag(C_1 \times C_1 \times C_2 \times C_3 \times C_3)$ to $Bag(C_1 \times C_1 \times C_2 \times C_2 \times C_3)$ defined by

$$f = \langle X_1^1 \ominus 3, X_1^1 \oplus 1, X_2^1, All_2, \mathbb{1}_{3,1} \rangle$$

- We first consider the mapping f_1 from $Bag(C_1 \times C_1)$ to $Bag(C_1 \times C_1)$ defined by $f_1 = \langle X_1^1 \ominus 1, X_1^1 \oplus 1 \rangle$.
  Using previous notations, $\mu_1(1) = 1$ and $\mu_1(2) = 2$. So,
  - $\mu_1^{-1}(1) = \{1,2\}$ and then $g_1^1 = X_1^1 \oplus 3$ and $G_1 = (X_1^2 \ominus 1 = X_1^1 \oplus 3)$
  - $\mu_1^{-1}(2) = \emptyset$ and then $g_1^2 = All_1$
  - No constant appears in f_1 so $H_1 = true$.
- We consider then the mapping from $Bag(C_2)$ to $Bag(C_2 \times C_2)$ $f_2 = \langle X_2^1, All_2 \rangle$.
  - $\mu_2^{-1}(1) = \{1\}$ and then $g_2^1 = X_2^1$ and $G_2 = true$
  - No constant appears in f_2 so $H_2 = true$.
- At last consider the mapping from $Bag(C_3 \times C_3)$ to $Bag(C_3)$ $f_3 = \langle \mathbb{1}_{3,1} \rangle$. We obtain
  - $\mu_3^{-1}(1) = \mu^{-1}(2) = \emptyset$ so $g_3^1 = g_3^2 = All_3$.
  - As $f_3^1 = \mathbb{1}_{3,1}$ then $H_3 = (X_3^1 = \mathbb{1}_{3,1})$.

So we obtain as result to our calculus:

$$^t f = [(X_1^2 \ominus 1 = X_1^1 \oplus 3) \wedge (X_3^1 = \mathbb{1}_{3,1})] \langle X_1^1 \oplus 3, All_1, X_2^1, All_3, All_3 \rangle$$

Syntactic Composition. Computing the composition $f \circ g$ of two mappings can raise syntactical problems. For instance, if g is a mapping from $Bag(\varepsilon)$ to $Bag(C_1)$ and f a mapping from $Bag(C_1)$ to $Bag(C_1 \times C_1)$ defined by $f = \langle X_1^1, X_1^1 \rangle$ and $g = \langle All_1 \rangle$, we clearly have $f \circ g = \sum_{c \in C_1} \langle c, c \rangle$ which cannot be expressed in our syntax. In the same way, $\langle All_i \rangle \circ \langle All_i \rangle = n_i.\langle All_i \rangle$ which is also not allowed in our syntax. So we impose for the computation of $f \circ g$ that

$$\forall i, k, g_i^k = All_i \Rightarrow (\exists! j, n \text{ such that } f_i^j = X_i^j \oplus n) \text{ and} (\forall j, f_i^j \neq All_i)$$

With this additional constraint, we can compute the transposition of any two tuples as follows (the proof can again be found in [EHPP04]): Using linearity of QWN mappings we restrict our self to the composition of tuples of elementary mappings.

Proposition 7 (Symbolic tuples composition). *Let $g = \langle g_1^1, \ldots, g_N^{e''_N} \rangle$ from $Bag(C)$ to $Bag(C'')$ and $f = \langle f_1^1, \ldots, f_N^{e'_N} \rangle$ from $Bag(C'')$ to $Bag(C')$ be two QWN mappings. Then $h = f \circ g = \langle h_1^1, \ldots, h_N^{e'_N} \rangle$ is defined by :*

$$\forall i, j \in [1..e'_i], h_i^j = \begin{cases} \textbf{if } f_i^j = X_i^{j'} \oplus n \textbf{ then } \begin{cases} \textbf{if } g_i^{j'} = X_i^{j''} \oplus m & \textbf{then } X_i^{j''} \oplus (n+m) \\ \textbf{if } g_i^{j'} = All_i & \textbf{then } All_i \\ \textbf{if } g_i^{j'} = \mathbb{1}_m & \textbf{then } \mathbb{1}_{n+m} \end{cases} \\ \textbf{if } f_i^j = All_i \quad \textbf{then } All_i \\ \textbf{if } f_i^j = \mathbb{1}_n \quad \textbf{then } \mathbb{1}_n \end{cases}$$

Example 2. Let $f = \langle X_1^1, X_1^1 \oplus 1, X_2^1 \rangle$ from $Bag(C_1 \times C_2)$ to $Bag(C_1 \times C_1 \times C_2)$ and $g = \langle X_1^3, All_2 \rangle$ from $Bag(C_1 \times C_1 \times C_1 \times C_2)$ to $Bag(C_1 \times C_2)$. The mapping $h = f \circ g$ from $Bag(C_1 \times C_1 \times C_1 \times C_2)$ to $Bag(C_1 \times C_1 \times C_2)$ is $h = \langle X_1^3, X_1^3 \oplus 1, All_2 \rangle$.

Others complications appear when computing the composition of two guarded tuples $[G_f]f \circ [G_g]g$ when there is a predicate $[X_i^j \oplus n = X_i^{j'} \oplus n']$ in G_f and when $g_i^j = g_i^{j'} = All_i$. For instance, if $f = [X_1^1 = X_1^2]\langle X_1^1, All_1 \rangle$ and $g = \langle All_1, All_1 \rangle$ (from $Bag(C_1)$ to $Bag(C1 \times C_1)$) then $(f \circ g) = \sum_{c \in C_1} \langle c, c \rangle$ which is not a quasi well formed mapping. Thus we have to introduce a second constraint for the computation:

$$G_f = [(X_i^j = X_i^{j'} \oplus n)] \Rightarrow (g_i^j \neq All_i \text{ or } g_i^{j'} \neq All_i)$$

Proposition 8 (Symbolic guarded tuples composition). *Let $g = \langle g_1^1, \ldots, g_N^{e''_N} \rangle$ from $Bag(C)$ to $Bag(C'')$ and f from $Bag(C'')$ to $Bag(C')$ be two QWN mappings. Then $h = [G_f]f \circ [G_g]g = [G \wedge G_g]f \circ g'$ where g' is defined by $g'^j_i = g_i^j$ except for some indices where a substitution occurs and where G is defined as follows (note that due to symmetry of guards we just consider non symmetrical cases for g_i^j and $g_i^{j'}$, and that the negation of a guard is easily defined from these constructions):*

– *G_f is $(X_i^j = X_i^{j'} \oplus n)$:*

$$\textbf{if } g_i^j = X_i^k \oplus m \textbf{ then } \begin{cases} \textit{if } g_i^{j'} = X_i^{k'} \oplus m' & \textit{then} \quad G = (X_i^k \oplus m = X_i^{k'} \oplus (n+m')) \\ \textit{if } g_i^{j'} = All_i & \textit{then} \quad G = G_f \\ \textit{if } g_i^{j'} = \mathbb{1}_m & \textit{then} \quad G = (X_i^k = \mathbb{1}_{n+m}) \end{cases}$$

$$\textbf{if } g_i^j = \mathbb{1}_n \quad \textbf{then } \begin{cases} \textit{if } g_i^{j'} = All_i \ \textit{then} \quad G = true \textbf{ and } g'^{j'}_i = \mathbb{1}_n \\ \textit{if } g_i^{j'} = \mathbb{1}_{n'} \ \textit{then} \quad G = (n = n') \end{cases}$$

– *G_f is $(X_i^j = \mathbb{1}_n)$:*

if $g_i^j = X_i^k \oplus m$ **then** $G = (X_i^k \oplus m = \mathbb{1}_n)$
if $g_i^j = All_i$ **then** $G = true$ **and** $g'^{j'}_i = \mathbb{1}_n$
if $g_i^j = \mathbb{1}_{n'}$ **then** $G = (n = n')$

Mapping Inclusion. Our last need is to be able to check that two QWN color mappings f and g are such that $\overline{f} \sqsubseteq \overline{g}$. This is the case if for every tuple tup_f of f there is a tuple tup_g of g which is such that the guard of tup_g is true and at each position in the co-domain of f and g, the elementary mapping in tup_g is either the broadcast mapping either the same mapping as in tup_f.

Proposition 9. *Let f and g be two QWN color mappings from C to C' such that*

– $f = \sum_{k=1}^{K_f} \alpha_{k,f} . [G_{k,f}] \langle f_{1,k}^1, \ldots, f_{N,k}^{e'_N} \rangle$
– $g = \sum_{k=1}^{K_g} \alpha_{k,g} . [G_{k,g}] \langle g_{1,k}^1, \ldots, g_{N,k}^{e'_N} \rangle$

If $\forall k \in [1..K_f], \exists k' \in [1..K_g]$ such that $G_{k',g} = true$, and $\forall i \in [1..|Cl|], j \in [1..e'_N]$, either $g_{i,k'}^j = All_i$, either $g_{i,k'}^j = f_{i,k}^j$ then $f \sqsubseteq g$.

4 Cases Studies

Flanagan and Qadeer proposed in [FQ03a] the following example where a counter *count* can be either read, incremented or decremented. Two shared variables, *a* and *b*, keep track of the number of increments or decrements performed on the counter.

```
int a, b, ma, mb, count = 0;

void incr(){             void decr(){             void read(){
  acquire (ma);           acquire (mb);           acquire (ma); int x = a;
  int x = a ;             int y = b;              acquire (mb); int y = b;
  count++;                count--;                release (mb);
  a = x+1;                b = y+1;                release (ma);
  release (ma); }         release (mb); }         return (tx-ty); }
```

The corresponding colored Petri net is depicted Fig. 1(a) and, for simplicity, we have duplicated some places on the figure (`ma`, `mb`, `a`, `b` and `count`). Note that the value of the local variables (`x` and `y`) is modeled by the coloration of the token contained in the places `p2`,...,`p4`, `q2`,...,`q4` and `u3`,...,`u5`. One aims to reduce this net, to check that a property that does not observe the five variables declared holds.

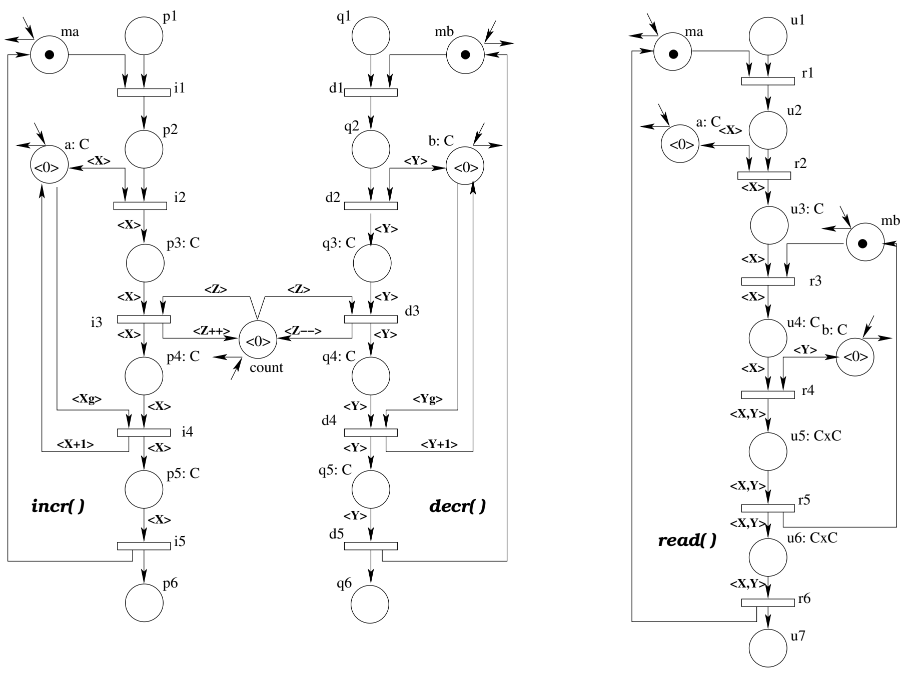
We first perform four post-agglomerations (`i4` with `i5`, `d4` with `d5`, `r5` with `r6` and `r4` with `r5r6` (the result of the agglomeration of `r5` with `r6`). These agglomerations are possible mainly because the transitions corresponding to f in these reductions have a single input: the place corresponding to p in the agglomeration scheme.

Then we perform three post-agglomerations (`i1` with `i2`, `d1` with `d2`, `r1` with `r2` and `r3` with `r4r5r6`). Let us detail the post-agglomeration of $i1$ with $i2$:

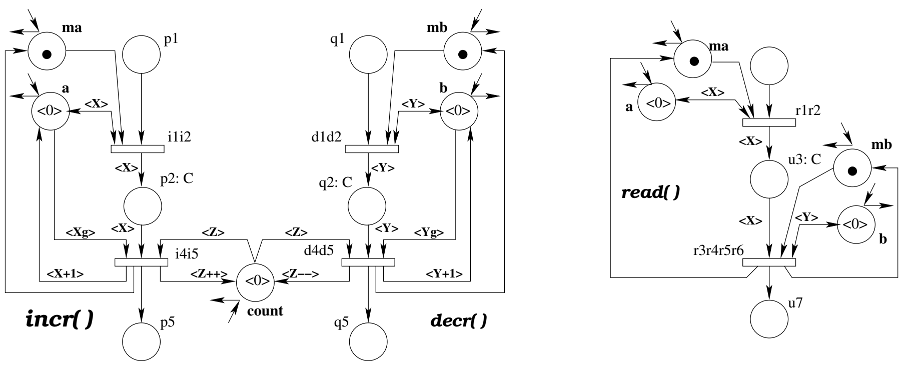
1. the p-agglomerability is obviously fulfilled;
2. concerning the F-independence, we use the following positive flow (on the domain ε) $\mathcal{F} = ma + p2 + \langle All_C\rangle.p3 + \langle All_C\rangle.p4 + u2 + \langle All_C\rangle.u3 + \langle All_C\rangle.u4$ which induces the binary positive invariant $\forall m \in Acc(N, m_0), m(ma) + \sum_{x\in C}(m(p2)(x) + m(p3)(x) + m(p4)(x)) + \sum_{x\in C}(m(u3)(x) + m(u4)(x)) = 1$. With the help of this invariant, we check for instance that (using notations of definition 10) for $q = a$, $t = i4$, $p_t = p4$ then $\Phi = \overline{{}^t(\langle X+1\rangle)} \circ \overline{(\langle X\rangle)} \circ \overline{{}^t(\langle \Pi_\varepsilon\rangle)} = All_C$ and $\Psi = \overline{{}^t(\langle X\rangle)} \circ \overline{{}^t(\langle All_C\rangle)} \circ \overline{\langle X_\varepsilon\rangle} = All_C$ and then $\phi \sqsubseteq \psi$ (note again that these computations are performed using only the syntax of the mappings).
3. the F-continuation is verified with the help of the flow $\langle All_C\rangle.a$ which induces the binary invariant $\forall m \in Acc(N, m_0), \sum_{x\in C} m(a)(x) = 1$.
4. the HF-interchangeability is ensured since $|H| = \{i1\}$

The reduced net is depicted Fig. 1(b).

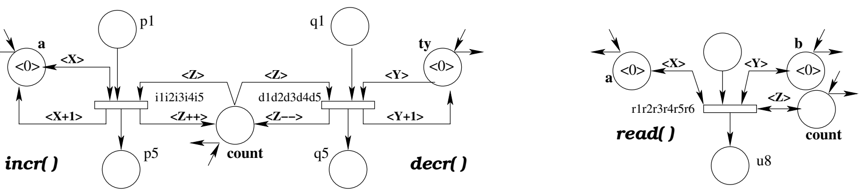
In this last model, we perform a pre-agglomeration of `d1d2` with `d4d5`. This reduction is possible since `r3r4r5r6` is a neutral transition and since `mb` induces a binary invariant ensuring that `d4d5` is not fireable when `d1d2` is fireable. Then we suppress the place `mb` (now an implicit place see [Had90]) and we apply a parallel pre-agglomeration of `r1r2` with `r3r4r5r6`, and `i1i2` with `i4i5`. So we suppress *ma* (now an implicit place) and we obtain a net reduced to three transitions (Fig.1(c)). Note that contrary to the results proposed in [FQ03a], our reduction process is fully automatic and, in addition to serializability, it preserves Petri nets liveness, deadlocks, as well as other properties expressed with the help of maximal or infinite sequences.



(a) The initial model



(b) After several agglomerations



(c) Fully reduced

Fig. 1. The Flanagan and Qadeer's example

5 Related Works

The first theoretical work concerning reduction of sequences into atomic actions for simplification purpose was performed by Lipton in [Lip75]. Lipton focused only on deadlock property preservation. Using parallel program notations of Dijkstra he defined "left" and "right" movers. Roughly speaking, a "left" (resp. "right") mover is a local process statement that can be moved forward (resp. delayed) w.r.t. statements of others processes without modifying the halting property. Lipton then demonstrated that, in principle, the statement `P(S)`, where `S` is a semaphore, is a "left" mover and `V(s)` is a "right" mover. Then Lipton proved that some parallel program are deadlock free by moving `P(S)` and `V(S)` statements and by suppressing atomic statements that have no effect on variables. However, two difficulties arise: the reduction preserves only the existence of deadlocks and the application conditions are difficult to be automatically checked. Thus, this work has been extended and adapted to different formalisms [CL98], [SC03], or to programming languages [SC03], [FQ03b], [FQ03a].

In Petri nets formalism, the first works concerning reductions have been performed by Berthelot [BRV80, Ber85]. The author focused only the preservation of specific Petri net properties such as liveness or boundedness. The link between transition agglomerations (the most effective reductions) and general properties, expressed in LTL formalism, is done in [PPP00]. However, these reductions rely on "pure" structural application conditions, which are, on the one hand, very efficient, but in the other hand, lead to a quite narrow application area.

Esparza and Schröter [ES01], simplify one point in the original pre-agglomeration conditions. However, they consider only 1-safe Petri nets (each place is bounded by 1), the application conditions remain purely structural, and as the authors focus only on infinite sequences preservation, their reductions do not even preserve the existence deadlock[2].

Orthogonally, Schnoebelen and Sidorova [SS00] characterize reductions by means of bisimulation. The interest in this approach is that one only needs to consider a particular subset of the markings and thus, can obtain a very abstract model. On the other hand, the applicability of these reductions is quite limited.

Recently we proposed in [HPP04] new Petri nets reductions that cover a large range of patterns by introducing algebraic conditions whereas the previously defined ones rely solely on structural conditions. We adapt them to colored nets in [EHPP04] and we show here how to automatize their applications with a well chosen syntax for colored Petri nets. Note that the expressiveness of colored Petri nets is sufficient to model concurrent software. Thus, these colored Petri nets reductions are a very efficient supplementary material for simplifying software model checking.

6 Conclusion

We have presented in this paper how new colored Petri nets reductions can be automatically performed using a precise syntax of colored net. In particular, we showed that a precise syntax of colored nets allows us to transform functional calculus into syntactical operations.

[2] Note moreover that being 1-safe is not a stable characteristic w.r.t. reductions.

We have illustrate on a recent and significant example that the use of these reductions leads to a very effective way to simplify model (and thus concurrent programs) while preserving general properties of the model (expressed for instance with an action-based linear time temporal logic).

The next step in our researches in this area will be to define directly in high-level languages (such as Ada or Java) equivalent conditions allowing to automatically infer transactions for simplifying software model checking.

References

[Ber85] G. Berthelot. Checking properties of nets using transformations. In G. Rozenberg, editor, *Advances in Petri nets*, volume No. 222 of *LNCS*. Springer-Verlag, 1985.

[BRV80] G. Berthelot, G. Roucairol, and R. Valk. Reduction of nets and parallel programs. In Brauer, W., editor, *LNCS: Net Theory and Applications*, volume 84, pages 277–290, Berlin, Heidelberg, New York, 1980. Springer-Verlag.

[CL98] Ernie Cohen and Leslie Lamport. Reduction in TLA. In *International Conference on Concurrency Theory*, pages 317–331, 1998.

[EHPP04] S. Evangelista, S. Haddad, and J.F. Pradat-Peyre. Colored Petri nets reductions for concurrent software validation. Technical report, CEDRIC, CNAM, Paris, 2004.

[EKPPR03] S. Evangelista, C. Kaiser, J. F. Pradat-Peyre, and P. Rousseau. Quasar: a new tool for analysing concurrent programs. In *Reliable Software Technologies - Ada-Europe 2003*, volume 2655 of *LNCS*. Springer-Verlag, 2003.

[ES01] J. Esparza and C. Schröter. Net Reductions for LTL Model-Checking. In T. Margaria and T. Melham, editors, *Correct Hardware Design and Verification Methods (CHARME'01)*, volume 2144 of *Lecture Notes in Computer Science*, pages 310–324. Springer-Verlag, 2001.

[Eva04] S. Evangelista. Syntactical rules for colored Petri nets manipulation. Technical Report 641, CEDRIC, CNAM, Paris, 2004.

[FQ03a] Cormac Flanagan and Shaz Qadeer. Transactions for software model checking. In Byron Cook, Scott Stoller, and Willem Visser, editors, *Electronic Notes in Theoretical Computer Science*, volume 89. Elsevier, 2003.

[FQ03b] Cormac Flanagan and Shaz Qadeer. A type and effect system for atomicity. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 338–349. ACM Press, 2003.

[GW93] Patrice Godefroid and Pierre Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Form. Methods Syst.*, 2(2):149–164, 1993.

[Had90] S. Haddad. A reduction theory for colored nets. In Jensen and Rozenberg, editors, *High-level Petri Nets, Theory and Application*, volume 424 of *LNCS*, pages 399–425. Springer-Verlag, 1990.

[HPP04] S. Haddad and J.F. Pradat-Peyre. Efficient reductions for LTL formulae verification. Technical report, CEDRIC, CNAM, Paris, 2004.

[Lip75] Richard J. Lipton. Reduction: a method of proving properties of parallel programs. *Commun. ACM*, 18(12):717–721, 1975.

[PPP00] D. Poitrenaud and J.F. Pradat-Peyre. Pre and post-agglomerations for *LTL* model checking. In M. Nielsen and D Simpson, editors, *High-level Petri Nets, Theory and Application*, number 1825 in LNCS, pages 387–408. Springer-Verlag, 2000.

[SC03] Scott D. Stoller and Ernie Cohen. Optimistic synchronization-based state-space reduction. In H. Garavel and J. Hatcliff, editors, *TACAS'03*, volume 2619 of *Lecture Notes in Computer Science*, pages 489–504. Springer-Verlag, April 2003.

[SS00] P. Schnoebelen and N. Sidorova. Bisimulation and the reduction of petri nets. In M. Nielsen and D Simpson, editors, *High-level Petri Nets, Theory and Application*, number 1825 in LNCS, pages 409–423. Springer-Verlag, 2000.

[Val93] Antti Valmari. On-the-fly verification with stubborn sets. In *Proceedings of the 5th International Conference on Computer Aided Verification*, pages 397–408. Springer-Verlag, 1993.

[VM97] François Vernadat and François Michel. Covering step graph preserving failure semantics. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pages 253–270. Springer-Verlag, 1997.