An Efficient Algorithm for Finding Structural Deadlocks in Colored Petri Nets

K. BARKAOUI [*] C. DUTHEILLET [#] S. HADDAD [#]

[*] Laboratoire CEDRIC
Conservatoire National des Arts et Métiers
292 rue Saint-Martin
75003 Paris - FRANCE
e-mail : barkaoui@cnam.cnam.fr

[#] IBP - Laboratoire MASI
Université P. & M. Curie
4 , Place Jussieu
75252 Paris Cedex 05 - FRANCE
e-mail : haddad@masi.ibp.fr

Abstract In this paper, we present an algorithm to compute structural deadlocks in colored nets under specified conditions. Instead of applying the ordinary algorithm on the unfolded Petri net, our algorithm takes advantage of the structure of the color functions. It is obtained by iterative optimizations of the ordinary algorithm. Each optimization is specified by a meta-rule, whose application is detected during the computation of the algorithm. The application of such meta-rules speeds up a step of the algorithm with a factor proportional to the size of a color domain. We illustrate the efficiency of this algorithm compared to the classical approach on a colored net modelling the dining philosophers problem.

1 Introduction

Analysis techniques of Place-Transition nets [1] can be grouped in two broad classes. The first approach consists in building the reachability graph, which gives full information but is usually very expensive [2].

Another approach - *the structural analysis* - consists in getting information about the behavior of the model directly from the structure of its underlying bipartite and valuated digraph, the initial marking being considered as a parameter [3]. Two kinds of structural analysis can be distinguished:

- *The algebraic analysis*, where the structure of the net is represented by the incidence matrix associated with its underlying digraph. It provides results such as conditions for liveness and boundedness of the net, or linear invariants [4] [5].
- *The graph theoretical analysis*, where the behavior of the net is related to the flow relation of subnets generated by remarkable subsets of places, such as structural deadlocks and traps [6]. With such techniques, liveness can be decided in polynomial time for different classes of nets [7] [8] [9].

However, modelling real complex systems yields so large models that it is difficult to handle the complexity of both the structural and the reachability analyses. Thus High-Level nets - Predicate/Transition nets [10] or Colored nets [11] have been introduced to concisely model large systems. In order to take advantage of this high-level description, the analysis of the model must be performed without refering to its equivalent unfolded net, i.e., a place-transition net with the same behavior. Thus, a lot of research is being done to directly apply to High-Level nets analysis techniques similar to those existing for Place-Transition nets.

The present paper is a contribution to graph theoretical analysis for colored nets. On Place-Transition nets, this approach uses the flow relation of subnets generated by structural deadlocks and traps, which are remarkable subsets of places. For these nets, the problem of finding structural deadlocks was attacked from different points of view. The technique of computing strongly connected deadlocks via a positive flow calculus on an expanded net [8] was optimized and extended to the class of Unary Predicate-Transition nets without guards [12]. The application of this method to less restrictive classes requires a generalization of positive flow computation [13], which seems to be a very difficult task [14]. In any case, the problem of obtaining the set of minimal deadlocks by means of flow computation is NP-complete.

In [15], the problem is expressed as a logic programming problem, leading to identify structural deadlocks (traps) with the set of solutions of a Horn-clause satisfiability problem. In colored nets, the flow relation is defined by both the underlying bipartite digraph and the functions labelling the arcs. Hence, a deadlock of a colored net may not correspond to a deadlock of the net's skeleton in the sense of [16], while the converse is true. Yet, for a restricted class of colored nets, the problem of finding deadlocks can be reduced to the problem of finding deadlocks in the skeleton [17], and the technique proposed in [15] could be exploited. But an efficient extension of the method to general colored nets is still an open problem.

In this paper, we develop a method for solving efficiently the following problem `Pb` by reasoning directly on the color functions, i.e., without an effective unfolding.

`Pb`:	For any pair of disjoints subsets of places P_{exc} and P_{inc},
	find a structural deadlock D such that $D \cap P_{exc} = \varnothing$ and $P_{inc} \subset D$,
	or decide that no such deadlock exists.

The paper is organized as follows. Section 2 introduces a property of deadlocks which yields propositional deduction rules. These rules are used in an algorithm that solves problem `Pb` for Place-Transition nets. We then extend this approach to colored Petri nets, rewriting the algorithm in terms of deduction rules of first order logic.

In Section 3, we optimize the applications of the previous rules by defining meta-rules. These meta-rules transform a set of rules so that one application of a transformed rule corresponds to the application of several initial rules.

In Section 4, we present the optimized algorithm that exploits the effects of these meta-rules, and we apply it on a model of the dining philosophers. Then we compare its execution on a colored net with the execution of the ordinary algorithm on the equivalent unfolded net. Section 5 contains the perspectives of this work.

2 Structural Deadlocks in Colored Nets

In the first part of this section, we recall the basic notations of Petri nets, together with the definition of a deadlock. We then extend this definition to colored Petri nets. Finally, we show on an example how the color functions of a model could be exploited to improve the efficiency of deadlock characterization algorithms.

2.1 Structural Deadlocks in Ordinary Petri Nets

Definition 2.1 *A Petri net N is a 4-tuple $<P, T, W^-, W^+>$ where*

P *is a finite set of places,*

T *is a finite set of transitions,*

W^- *(resp. W^+) : $P \times T \rightarrow \mathbb{N}$ is the input (resp. output) function.*

We also define the input set and output set of a subset of places, which contain respectively the input and output transitions of the places under consideration.

Definition 2.2 *Let $p \in P$. The input (resp. output) set of p, denoted by $\bullet p$ (resp. $p\bullet$) is defined as:*

$$\bullet p = \{t \in T \mid W^+(p, t) \neq 0\}$$

$$(resp. \quad p\bullet = \{t \in T \mid W^-(p, t) \neq 0\})$$

This definition can be extended to a subset D of places:

$$\bullet D = \bigcup_{p \in D} \bullet p \qquad (resp. \quad D\bullet = \bigcup_{p \in D} p\bullet)$$

A similar definition exists for transitions.

Definition 2.3 *Let $t \in T$. The input (resp. output) set of t, denoted by $\bullet t$ (resp. $t\bullet$) is defined as :*

$$\bullet t = \{p \in P \mid W^-(p, t) \neq 0\} \qquad (resp. \quad t\bullet = \{p \in P \mid W^+(p, t) \neq 0\})$$

We now give the definition of a structural deadlock:

Definition 2.4 *Let D be a non-empty subset of places. D is a structural deadlock iff $\bullet D \subset D\bullet$.*

The following property states that if no input place of a transition belongs to a deadlock, then the output places of this transition neither belong to the deadlock. Such a property is introduced since it leads to the construction of maximal deadlocks.

Property 2.1 *Let D be a non-empty subset of places. We have:*

$$[\ \forall t \in T, \quad \bullet t \subset (P \setminus D) \ \Rightarrow\ t\bullet \subset (P \setminus D)\ ] \ \Leftrightarrow\ D \text{ is a structural deadlock}$$

Proof: If we consider the negation of the left-hand part of the property, we can write:

$[\forall\, t \in T,\ \ \exists\, p \in D, p \in t\bullet \Rightarrow \exists\, p' \in D, p' \in \bullet t\,] \Leftrightarrow$ D is a deadlock.

but we also have $\exists\, p \in D,\ p \in t\bullet \Leftrightarrow t \in \bullet D$

and $\exists\, p' \in D,\ p' \in \bullet t \Leftrightarrow t \in D\bullet$

Hence, the left-hand part of the property is equivalent to:

$$\forall\, t \in T, \qquad t \in \bullet D \Rightarrow t \in D\bullet$$

which is the definition of a structural deadlock.

Actually, Property 2.1 defines removal rules according to which places that do not belong to a deadlock can be eliminated from the net. We know that a place cannot belong to a deadlock if it is an output place of a transition such that no input place of this transition belongs to the deadlock.

Definition 2.5

- *Let $t \in T$. Then R(t) is the removal rule associated with t and is written as:*

$$R(t)\ :\quad \bullet t \quad \Rightarrow \quad t\bullet$$

- *Let E be a subset of P. Then R(t) is applicable on E iff $\bullet t \subset E$. The application of t on E is given by $E := E \cup t\bullet$.*

We will call hypothesis the left-hand part of the rule, whereas the right-hand part will be called conclusion. Using Definition 2.5, we can write a generic algorithm for finding a structural deadlock satisfying two constraints:

- places contained in a set called P_{exc} must not belong to the deadlock,
- places contained in a set called P_{inc} must belong to the deadlock.

The principle of the algorithm is simple: a set R is initialized with all the rules of the net, a set Removed is initialized with the set P_{exc} and we try to apply on Removed the different rules of R, i.e., to remove the places that do not belong to the deadlock. When no more rule is applicable, we verify that P_{inc} is included in the complementary of Removed. If so, the algorithm has produced the maximal deadlock satisfying the constraints. Else, there is no deadlock satisfying these constraints.

Abstract Algorithm 1

```
Removed := $P_{exc}$
While ∃ t such that R(t) is applicable on Removed do
     Apply R(t) on Removed
     Delete R(t) in R
done;
Deadlock := P \ Removed
If $P_{inc}$ ⊂ Deadlock and Deadlock ≠ ∅ then return (success,Deadlock)
else return(failure)
```

An optimized implementation of this algorithm [15] requires an execution time proportional to the size of the net (expressed as the sum of the number of nodes and arcs). In the next section we extend the definitions and the algorithm to colored nets.

2.2 Structural Deadlocks in Colored Nets

Before giving a formal definition of a structural deadlock in a colored net, we recall the notations associated with this model.

Definition 2.6 *A colored Petri net N is a 5-tuple $<P, T, C, W^-, W^+>$ where*

- P *is a finite set of places,*
- T *is a finite set of transitions,*
- C *is the color function, mapping $P \cup T$ onto Ω, where Ω is some finite set of finite and non-empty sets. $C(s)$ is called the color set of s.*
- W^- *(resp. W^+) is the input (resp. output) function defined on $P \times T$, where $W^-(p, t)$ (resp. $W^+(p, t)$) is a function from $C(t)_{MS}$ to $C(p)_{MS}$.*

In this definition, E_{MS} denotes the set of multisets over a set E. Informally, an element of E_{MS} is a subset of E where the same element can appear several times.

Notation:

In the rest of this section, we will use PC to denote the set $[\bigcup_{p \in P} \{p\} \times C(p)]$, i.e., the set containing couples of places and associated color instances.

Definitions 2.7 and 2.8 are the extension to colored nets of Definitions 2.2 and 2.3.

Definition 2.7 *Let p be a place, and $c \in C(p)$ be a color instance of p. The input (resp. output) set of (p, c) is defined by the set $\bullet(p, c)$ (resp. $(p, c)\bullet$)*

$$\bullet(p, c) = \{(t, c') \mid W^+(p, t)(c')(c) > 0\}$$

$$(\textit{resp.} \quad (p, c)\bullet = \{(t, c') \mid W^-(p, t)(c')(c) > 0\})$$

This definition can be extended to a subset $D \subset PC$, i.e., a subset D of places with associated color instances:

$$\bullet D = \bigcup_{(p,\, c) \in D} \bullet(p, c) \qquad (\textit{resp.} \quad D\bullet = \bigcup_{(p,\, c) \in D} (p, c)\bullet\ )$$

Definition 2.8 *Let t be a transition, and $c \in C(t)$ be a color instance of t. The input (resp. output) set of (t, c) is defined by the set $\bullet(t, c)$ (resp. $(t, c)\bullet$)*

$$\bullet(t, c) = \{(p, c') \mid W^-(p, t)(c)(c') > 0\}$$

$$(\textit{resp.} \quad (t, c)\bullet = \{(p, c') \mid W^+(p, t)(c)(c') > 0\})$$

Now the formal expression of the definition of a structural deadlock is quite similar to the definition of a deadlock in an ordinary Petri net.

Definition 2.9 *Let D be a non-empty subset of PC. D is a structural deadlock iff* $\bullet D \subset D\bullet$.

In the same way as we did for ordinary Petri nets, we now give a property that is equivalent to the definition of a deadlock.

Property 2.2 *Let D be a non-empty subset of PC. We have:*

$$[\ \forall (t, c) \in [\bigcup_{t \in T} \{t\} \times C(t)],\quad \bullet(t, c) \subset (PC \setminus D) \ \Rightarrow\ (t, c)\bullet \subset (PC \setminus D)\ ]$$
$$\Leftrightarrow\ D \text{ is a structural deadlock}$$

The proof of this property is quite similar to the proof of Property 2.1.

In order to extend the algorithm computing the maximal structural deadlock that verifies some conditions, we extend the definition of removal rules to colored nets. Considering a transition t and an associated color c, we know from Property 2.2 that if no input place of (t, c) in the unfolded net belongs to a deadlock, then also none of the output places belongs to this deadlock. Hence, we introduce a function that gives for each place p the set of colors of p that are input (resp. output) of (t, c). These functions map the color domain of t onto the powerset of the color domain of p, i.e., the set of subsets of C(p), that we denote by P[C(p)] .

Definition 2.10 *Let p be a place and t be a transition. The function $\Gamma^-(p, t) : C(t) \rightarrow P[C(p)]$ defines for every color instance c of t the corresponding input colors in p:*

$$\Gamma^-(p, t)(c) = \{c' \mid W^-(p, t)(c)(c') > 0\}$$

The output colors are defined by $\Gamma^+(p, t) : C(t) \rightarrow P[C(p)]$

$$\Gamma^+(p, t)(c) = \{c' \mid W^+(p, t)(c)(c') > 0\}$$

However, we already know that if there is no arc between p and t (resp. t and p), the set $\Gamma^-(p, t)(c)$ (resp. $\Gamma^+(p, t)(c)$) will be empty for any value of c. Hence, we need to determine which places and transitions are connected, disregarding the color functions. The sets •t and t• that we define now are the sets that would be calculated on the ordinary Petri net obtained when ignoring the color functions on the arcs.

Definition 2.11 *Let t be a transition. The input (resp. output) places of t are defined by the set •t (resp. t•):*

$$\bullet t = \{p \in P \mid \exists c \in C(t),\ \Gamma^-(p, t)(c) \neq \emptyset\}$$
$$(\textit{resp.} \qquad t\bullet = \{p \in P \mid \exists c \in C(t),\ \Gamma^+(p, t)(c) \neq \emptyset\})$$

The following definition introduces new notations that allow a more compact expression of set of places and associated colors.

Definition 2.12 *Let p and q be two places, E and F be subsets of C(p) and C(q) respectively. We define:*

- $[p, E] \ = \ \{(p, c) \mid c \in E\}$
- $[p, E] \ \wedge [q, F] \ = \ [p, E] \cup [q, F]$

Now we have all the elements that allow us to define the rules, whose application will determine if a place belongs to a deadlock or not.

Definition 2.13

- *Let $t \in T$. Then $R(t)$ is the removal rule associated to t and is written as:*

  $$R(t)\ :\quad \bigwedge_{p \in {}^\bullet t} [p,\ \Gamma^-(p,\ t)] \quad \Rightarrow \quad \bigwedge_{p \in t^\bullet} [p,\ \Gamma^+(p,\ t)]$$

- *The color domain $C[R(t)]$ of the rule is the color domain of the transition.*
- *Let E be a subset of PC, and $c \in C[R(t)]$. $R(t)$ is applicable for c on E iff*

  $$\bigwedge_{p \in {}^\bullet t} [p,\ \Gamma^-(p,\ t)(c)] \ \subseteq E$$

- *The application of R(t) for c on E is given by*

  $$E := E \cup \bigwedge_{p \in t^\bullet} [p,\ \Gamma^+(p,\ t)(c)]$$

We use removal rules to write an algorithm that computes structural deadlocks. We first define a set P_{exc} (resp. P_{inc}) of places and associated color instances that are excluded from the deadlock (resp. that must belong to the deadlock). We initialize a set R with the rules of the net, a set Removed with P_{exc} and we try to apply the removal rules. If a rule R(t) is applicable for a color c, we add the output places of (t, c) to Removed, i.e., we remove these places because they cannot belong to the deadlock, and we also remove c from the color domain of the rule. When this color domain is empty, we remove the rule from R. When no more rule is applicable, places that have not been included in Removed form the maximal deadlock under the constraints of P_{exc}. If P_{inc} is included in the complementary of Removed, then the research has been successful.

```
Abstract Algorithm 2
Removed := $P_{exc}$;
While ∃ t and c such that R(t) is applicable for c on Removed
do
     Apply R(t) for c on Removed ;
     C[R(t)] := C[R(t)] \ {c};
     If C[R(t)] = ∅ then Delete R(t) in R;
done;
Deadlock := PC \ Removed;
If $P_{inc}$ ⊂ Deadlock and Deadlock ≠ ∅ then return (success,
Deadlock)
else return (failure)
```

Clearly, the complexity of this algorithm depends on the size of the unfolded net, even if we extend the optimized version that exists for ordinary Petri nets. But such an algorithm does not exploit at all the structure of the color functions, which corresponds to particular structures of the graph of the unfolded net. Actually, these functions often have a regular structure that can be used to optimize the application of removal rules. We present two examples of this optimization in the next section.

2.3 Two Introductory Examples

In this section, we present two structures of ordinary Petri nets that correspond to usual color functions, and we show how these color functions can be exploited to improve the efficiency of algorithms for finding structural deadlocks.

The first case where we can benefit from the color functions is the case where several places of the unfolded net can be eliminated at the same time, because their input transitions have the same input places. This is what we call "parallel deduction". We show on an example how it works. Let us consider the colored Petri net in Figure 1. The removal rule associated with the transition of the colored net is:

$$\texttt{[P1, }\varepsilon\texttt{]} \Rightarrow \texttt{[P2, X]}$$

But when considering the unfolded net, we immediately remark that places P2a, P2b and P2c have the same set of input places, namely place P1. Hence, instead of trying the possible assignments of variable X, if we already know that P1 does not belong to a deadlock, we could eliminate all of the instances of P2 at the same time.

To do so, we should rewrite the former removal rule as:

$$\texttt{[P1, }\varepsilon\texttt{]} \Rightarrow \texttt{[P2, C1.all]}$$

where function `C1.all` is defined by : $\texttt{C1.all(x)} = \bigcup_{\texttt{x} \in \texttt{C1}} \{\texttt{x}\}$

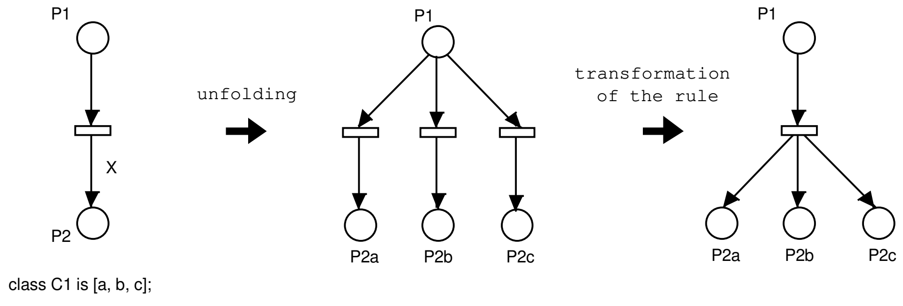

Fig. 1: Parallel deduction

The applicability of this new rule does not depend on any variable, whereas the original rule required the assignment of X. Hence, the number of applications of the rule is divided by the cardinality of the class on which X is defined.

We consider now the colored net in Figure 2, whose graph contains a loop. According to Property 2.2, we know that (P, a) cannot belong to a deadlock, and then according to Definition 2.13, the removal rule associated with T1 is:

$$\texttt{True} \Rightarrow \texttt{[P, a]}$$

If we consider now transition T2, the associated removal rule is:

$$\texttt{[P, X]} \Rightarrow \texttt{[P, X++1]}$$

where `X++1` stands for the successor function, modulo the color class. By trying successively the different assignments of this rule, we deduce that if (P, a) does not belong to a deadlock, then neither (P, b) nor (P, c) belongs to the deadlock. In fact, the successive applications of the rule are equivalent to finding a deadlock in the unfolded net. These successive applications are what we call "iterative deduction".

However, if we try to iterate the application of the rule before any assignment, then we immediately find that this rule can be replaced by the following rule:

`[P, X]  ⇒  [P, C.all]`

for which only one application is necessary in order to obtain the wanted information. The application of this new rule, combined with the application of the rule associated with T1 ensures that no occurrence of P belongs to a deadlock. With this rule, the number of steps of the removal algorithm is divided by the cardinality of the color domain of T2.

In fact, the transformation of the rule is based on the cyclic structure of the graph of the colored net. One lap in the colored cycle provides information, through the color functions, on the color instances of places that must be excluded from the deadlock. This information is immediately reused to try to obtain more results.

Hence, by a combination of the color functions that appear in the cycle, we obtain directly the information given by a repeated application of the original rule.

These two examples show that for particular structures of the unfolded net, represented by specific structures of both the color functions and the graph of the net, the deadlock computation algorithm can be improved.

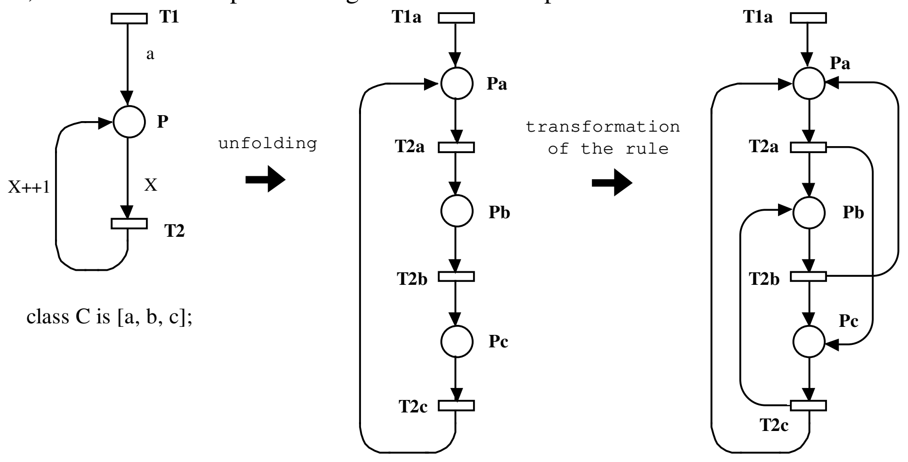

Fig. 2: Iterative deduction

What we have presented on the examples is in fact a very general approach, namely transforming the removal rules of the unfolded net by applying meta-rules on them. In order to obtain automatic transformations, we develop and formalize the approach in the next section.

3 Removal Rules and Meta-Rules

The aim of the meta-rules is to rewrite removal rules in order to improve their application. At every step of the computation of a deadlock, one application of a rule must provide as much information as possible. The computation of a deadlock relies on the initial creation of one rule per transition of the net. A meta-rule is a transformation that applies to a rule R, or to a set of rules, and that produces a new rule R', or a set of new rules. A meta-rule can modify not only the rule, but also its color domain, i.e., the possible assignments of the variables that appear in the rule. The application of a meta-rule transforms a rule in such a way that one application of the new rule corresponds to a series of applications of the original rule in the unfolded net. The efficiency of the approach is thus linked to the number of ordinary rules that have been replaced by this transformation.

As removal rules correspond to transitions, we can identify a set of rules with the colored net from which they have been written. The modification of a rule that results from the application of a meta-rule can be seen as a transformation of the associated Petri net. Based on this identification, we extend to rules notions that exist for colored nets. The skeleton of a rule will be the set of places that appear in the rule, disregarding the color instances that are associated with these places. We will also use p• to denote the set of rules for which p appears in the hypothesis.

3.1 Basic Meta-Rules

In an ordinary Petri net, the algorithm for computing a structural deadlock consists in removing the places that cannot belong to the deadlock. In a colored Petri net, we only remove color instances of the places. However, if all the instances of a place have been removed, then we can remove the place. Removing the place from the net is equivalent to remove it from the rules. Actually, in both cases this suppression accounts for the fact that nothing more will be proved for this place, and also that any hypothesis requiring that a subset of color instances of this place does not belong to the deadlock will be true. Hence, our first meta-rule consists in removing the places for which everything has been proved.

Definition 3.1 (MR1) *Let p_0 be a place such that no element of $[p_0, C(p_0)]$ belongs to a deadlock. Then meta-rule MR1 consists in replacing rule R by rule R' such that C(R') = C(R) and:*

$$R : \bigwedge_{p \in P_1} [p, f_p] \Rightarrow \bigwedge_{q \in P_2} [q, f_q]$$

$$R' : \bigwedge_{p \in P_1 \setminus \{p_0\}} [p, f_p] \Rightarrow \bigwedge_{q \in P_2 \setminus \{p_0\}} [q, f_q]$$

Once MR1 has been applied, some rules may no longer have a conclusion. These rules can be removed because they will give no further information.

Definition 3.2 (MR2) *Let R be a rule whose conclusion is empty. Then meta-rule MR2 consists in removing R.*

The two former meta-rules do not depend on the color functions that appear around the transition corresponding to rule R. Hence, they can be applied on the skeleton of the colored net as well. We present now two meta-rules that improve the removal process by using the structure of the colored net through the color functions.

3.2 Parallel Deduction

The following meta-rule corresponds to the parallel deduction process. As we have shown on the example of Section 2.3, we can define a meta-rule exploiting the case where the hypothesis of a rule is partially independent of the conclusion. The meaning of this meta-rule can be explained as follows. We recognize on the color functions around a transition that, in the unfolded net, some transitions have the same input places. The semantics of the meta-rule is then to substitute all these transitions by a unique transition, whose input places are those common input places and whose output places are the union of the output places of the transitions. And the substitution is performed at the colored net level.

In order to give a formal expression of this meta-rule, we introduce two functions:

Definition 3.3 *Let C1 and C2 be two sets.*

- *The projection π of $C1 \times C2$ on C1 is defined by:* $\pi(<x, y>) = x$
- *Σ is a function on C1 that defines the sum over the elements of C2:*

$$\Sigma(x) = \bigcup_{y \in C2} \{<x, y>\}$$

Consider now a rule R with a domain $C(R) = C1 \times C2$, where C1 and C2 can in turn be Cartesian products of color sets. For a partial independence of the hypothesis and the conclusion, we want a condition on the color functions in the hypothesis, such that the hypothesis is verified independently of the assignment of the variable in C2.

A necessary and sufficient condition is that the functions of the hypothesis can be written as $g_p = f_p \circ \pi$, where f_p is a function defined on C1. After the application of the meta-rule, the composition of function f_p with a projection is no longer necessary, as the domain of the rule has been reduced to C1.

In the conclusion of rule R, a function f_q that applies to $C1 \times C2$ is associated to each place q. But the aim of the meta-rule is to obtain a new rule, whose domain is only C1 and whose conclusion is the union of the conclusions obtained for all the assignments of the variable in C2. Hence, once an assignment has been done for the variable in C1, we first compute all the couples that associate this assignment to an element of C2. This is done by applying function Σ. Then we apply f_q to all these couples.

Definition 3.4 (MR3) *Let R be a rule whose domain can be written as $C(R) = C1 \times C2$, and whose expression is:*

$$R : \bigwedge_{p \in P1} [p, f_p \circ \pi] \Rightarrow \bigwedge_{q \in P2} [q, f_q]$$

Then the meta-rule MR3 produces a new rule R' whose domain is C(R') = C1, and whose expression is :

$$R' : \bigwedge_{p \in P1} [p, f_p] \Rightarrow \bigwedge_{q \in P2} [q, f_q \circ \Sigma]$$

However, the problem is to detect in which cases this meta-rule can be applied. The easiest to detect and also the most frequent case is when the expression of a rule contains in the conclusion a variable that does not appear in the hypothesis. The application of the meta-rule then consists in replacing this variable by the constant representing all the elements of the domain of the variable.

We now present the last meta-rule that corresponds to the iterative deduction process.

3.3 Iterative Deduction

The following meta-rule corresponds to the iterative deduction process. As we have shown on the example of Section 2.3, we can define a meta-rule exploiting the case where there exists a circuit such that each of its transition has only one input place in the graph of the Petri net. We call external places w.r.t. a circuit places that are connected to a transition of the circuit but do not belong to the circuit. We are thus interested in circuits without external input places.

Moreover, if the functions valuating the input arcs of the transitions of the circuit are all identity functions, then all the transitions of the circuit only have one input place in the unfolded net. Hence, if a place of the unfolded circuit does not belong to the deadlock, then no descendant of this place obtained by a path meeting only transitions of the circuit can belong to the deadlock.

The meaning of the meta-rule can be explained as follows. We substitute to the output places of a transition the set of descendant places of this transition. This set of places can be obtained by computing the transitive closure of the subnet under consideration. An algorithm for this computation can be found in [18]. It uses the transitive closure of a color function that we recall now.

Definition 3.5 *Let E and F be two sets, f be a function P(E) → P(F). The transitive closure f^* of f is defined by:*

$$c' \in f^*(c) \iff \exists n \geq 0 \text{ such that } c' \in f^n(c)$$

where $f^n = f \circ \ldots \circ f$ (n times).

For the sake of clarity, we first present a simplified version of the meta-rule. In this version, we only consider a circuit without external output places.

Let $p_0, \ldots, p_{n-1}$ be the places belonging to the circuit, and let f_i be the color function valuating the input arc of place p_{i+1}. Starting from an instance c of p_i, we can reach the descendant instances c' of p_j that are such that:

$$c' \in f_{j-1} \circ \ldots \circ f_i(c)$$

Hence, as the graph is cyclic, from an instance c of p_i, we may reach all the instances c' of p_i that are such that

$$c' \in f_{i-1} \circ \ldots \circ f_i(c)$$

But these instances may in turn reach instances c'' such that

$$c'' \in f^{i-1} \circ \ldots \circ f^{i}(c'), \text{ i.e., } c'' \in f^{i-1} \circ \ldots \circ f^{i} \circ f^{i-1} \circ \ldots \circ f^{i}(c)$$

By repeating the process, from an instance c of p_i, we can reach the descendant instances c' of p_i that are such that

$$c' \in (f^{i-1} \circ \ldots \circ f^{i})^{*}(c)$$

As know how to compute the colors that can be reached from these colors of p_i for any place p_j belonging to the circuit , we can now give the general expression of the meta-rule for a circuit without external output places.

Definition 3.6 (simplified MR4)

Let $p_0, \ldots, p_{n-1}$ be n places belonging to a cycle of the colored net. Let t_i be the output transition of p_i in the cycle, and R_i be the rule associated with t_i. If R_i has the following expression:

$$R_i : \quad [p_i,\ id] \quad \Rightarrow \quad [p_{i+1},\ f_i]$$

the application of the meta-rule transforms R_i in R'_i such that:

$$R'^{i} : \quad [p^{i},\ id] \quad \Rightarrow \quad \bigwedge_{j=0}^{n-1} [p^{j},\ f^{j-1} \circ \ldots \circ f^{i} \circ (f^{i-1} \circ \ldots \circ f^{i})^{*}]$$

where the operations on the indices are performed modulo n.

The introduction of external output places does not change the expression of the color functions, but these places must now appear in all the rules. The meta-rule becomes:

Definition 3.7 (MR4)

Let $p_0, \ldots, p_{n-1}$ be n places belonging to a cycle of the colored net. Let t_i be the output transition of p_i in the cycle, and P_i be the set of output places of t_i that do not belong to the cycle. Let R_i be the rule associated to t_i. If R_i has the following expression:

$$R_i : \quad [p_i,\ id] \quad \Rightarrow \quad [p_{i+1},\ f_i] \quad \bigwedge_{q \in P_i} \quad [q,\ g_q]$$

the application of the meta-rule transforms R_i in R'_i such that:

$$R'^{i} : \quad [p^{i},\ id] \quad \Rightarrow \quad \bigwedge_{j=0}^{n-1} \left\{ [p^{j},\ f^{j-1} \circ \ldots \circ f^{i} \circ (f^{i-1} \circ \ldots \circ f^{i})^{*}] \quad \bigwedge_{q \in P_j} [q,\ g^{q} \circ f^{j-1} \circ \ldots \circ f^{i} \circ (f^{i-1} \circ \ldots \circ f^{i})^{*}] \right\}$$

where the operations on the indices are performed modulo n.

We have considered only identity functions on input arcs. The result can be easily extended to the case where the functions on input arcs are bijective, by using the same kind of transformation as for reductions in colored nets [19].

4 An Algorithm to Compute Structural Deadlocks

4.1 Presentation and Definition of the Algorithm

This algorithm differs from Algorithm 2 on the two following points:

- The occurrence of places in rules is used only at the "skeleton" level.
- The structures of both the color functions and the skeleton are exploited by meta-rules MR3 and MR4.

More precisely, the algorithm works as follows. The first step (instructions 1-7) initializes the color instances of places that are excluded from the deadlock, the rules to be examined - those with a hypothesis containing a place for which at least one color is excluded from the deadlock - and tries to apply the meta-rules once.

The main loop (instructions 8-20) then applies one rule at a time with all the possible assignments, updating the excluded color instances of places and the rules to be examined. The main optimization (instructions 15-17) is the detection of the application of the meta-rules; due to the test in (15), the execution of instruction 16 always applies at least one meta-rule (see below). In this loop, all the control instructions work at the skeleton level and colors only appear in the application of an ordinary rule. The end of the algorithm (instructions 21-22) is identical to that of Algorithm 2. The algorithm is developed below.

Notation Given E, some subset of PC, we need to know the subset of colors of a place included in E. So we introduce the following notation: $E.p = \{c \mid (p, c) \in E\}$

Abstract Algorithm 3

```
(1)  Compute the elementary circuits of the skeleton
(2)  Removed := $P_{exc}$
(3)  To_Examine := $\emptyset$
(4)  For all place p such that Removed.p $\neq \emptyset$ do
(5)       Insert p• in To_Examine
(6)  done
(7)  Apply the Meta-rules on Removed
(8)  While To_Examine $\neq \emptyset$ Do
(9)       Extract R from To_Examine
(10)      A := {c | R is applicable for c on Removed}
(11)      For all c in A do
(12)           Apply R for c on Removed ;
(13)           For all p such that Removed.p has increased do
(14)                Insert p• in To_Examine;
(15)                If Removed.p = C(p) then
(16)                     Apply the Meta-rules
(17)                endif
(18)           done
(19)      done
(20) done
(21) Deadlock := PC \ Removed
(22) If $P_{inc} \subset$ Deadlock and Deadlock $\neq \emptyset$ then
          return (success, Deadlock)
     else return (failure).
```

Explanations and details of implementation

(1) In order to apply meta-rule MR4, we compute all the elementary circuits of the skeleton. We choose to compute the circuits before eliminating the places in P_{exc} because the result of this computation can be used for different problems on the same net, such as the research of deadlocks and traps, or the research of deadlocks with different conditions, namely different sets P_{exc}.

The computation of the circuits can be done in a time proportional to the product of the size of the skeleton and the number of circuits [18]. Anyway, this time is independent of the size of the color domains, which is the relevant criterion of complexity for colored nets. To test efficiently if a circuit fulfils the condition of MR4, we associate to each circuit the number of external input places and we link each place to the circuits of which it is an external input place. Each time we apply MR1, we update these numbers and see if new applications of MR4 are possible.

(3),(5),(8),(9),(14) `To_Examine` is a set-type variable which provides a termination test to the algorithm. Again the updating of this variable is related to the skeleton of the set of rules. Each time a color of a place is added to Removed, the rules with this place appearing in the hypothesis are selected for examination. With a link between a place and each rule where the place appears in the hypothesis, the updating of `To_Examine` is quick.
(10),(12) These steps are the most time-consuming ones and they depend on the color domains. Many heuristics, used in simulation techniques, (and already applicable to Algorithm 2) optimize this step but the complexity remains color domain dependent. *But the aim of Algorithm 3 is to reduce the color domain of the rules and to transform the conclusions of the rules in such a way that the cost of testing is minimal and the information brought by the application of the rule is maximal.*

(13),(15) The evolution of Removed.p can easily be memorised with two counters, one for the current cardinality and one for the preceding one. Hence these two tests are a comparison of integers. If the test of instruction (15) is successful, we already know that we can apply the deletion meta-rule MR1 on place p, and possibly any of the three other meta-rules:

- MR2 becomes applicable if p was the last element of the conclusion of a rule
- MR3 becomes applicable if p appeared in the hypothesis of a rule and determined part of the assignment. This can be detected by a very simple syntactic analysis of the color functions, such as for instance the vanishing of a variable in the hypothesis of a rule. The next section will give an illustration of this.
- MR4 becomes applicable if p was the last external input place of a circuit.

One can see that the overhead time added by the test of the meta-rules is minimal and once again independent of the color domain.

4.2 Application: The Dining Philosophers

The net in Figure 3 models the dining philosophers, in the case where they do not take both forks at the same time. Initially all the philosophers are `Thinking` and all `Forks` are free. When a philosopher `X` wants to eat, he takes his right fork (`X++1`) and `Waits` for his left fork (`X`). If his left fork is free, then he `Eats`. When he stops eating, he releases the two forks and starts thinking again. We will use C to denote the set of philosophers (and of forks).

We are looking for possibly deficient, i.e., insufficiently marked deadlocks. For the philosophers net, such a deadlock may appear with waiting philosophers.

So we look for deadlocks that do not contain place Wait, and we start our algorithm with P_{exc} = [Wait, C(Wait)] and P_{inc} = ∅. The rules associated with transitions T1, T2 and T3 respectively are:

```
R1   [Think, X] ∧ [Forks, X++1] ⇒ [Wait, X]
R2   [Wait, X] ∧ [Forks, X] ⇒ [Eat, X]
R3   [Eat, X] ⇒ [Forks, X + X++1] ∧ [Think, X]
```

We first (instruction 1) compute the elementary circuits and we find three ones:

(Think, Wait, Eat), (Forks, Wait, Eat), (Forks, Eat)

Then we initialize (instruction 2) Removed with [Wait, C] and (instructions 3-6) we update To_Examine with rule R2. We now apply the meta-rules (instruction 7): MR1 is applicable on Wait, and the rules become:

```
R1   [Think, X] ∧ [Forks, X++1] ⇒ True
R2   [Forks, X] ⇒ [Eat, X]
R3   [Eat, X] ⇒ [Forks, X + X++1] ∧ [Think, X]
```

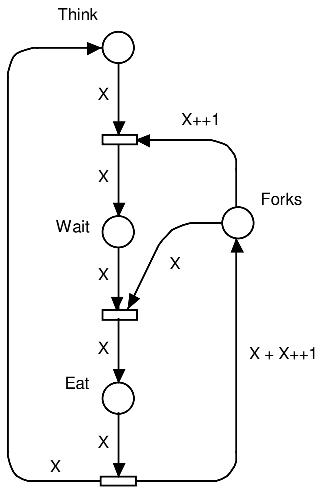

Fig. 3: The dining philosophers with forks taken separately.

MR2 is applicable on R1 and the rules become:

```
R2   [Forks, X] ⇒ [Eat, X]
R3   [Eat, X] ⇒ [Forks, X + X++1] ∧ [Think, X]
```

MR4 is applicable on the circuit (Forks, Eat). As X^* is `X` and $(X\ +\ X{+}{+}1)^*$ is `C.All`, the rules become:

```
R2    [Forks,X]⇒[Forks,C.all]∧[Eat,C.all] ∧ [Think,C.all]
R3    [Eat,X] ⇒ [Eat,C.all]∧[Forks,C.all] ∧ [Think,C.all]
```

The net in Figure 4 graphically describes the two rules. At instruction 7, To_examine contains R2, so we extract it and try to apply it.

However, as Removed.forks is empty, no assignment is possible and A is empty (instruction 10). Thus we exit from the main loop and return success with the following deadlock:

```
[Eat, C.all] ∧ [Forks, C.all] ∧ [Think, C.all]
```

This deadlock contains all the forks. Is there a deadlock that contains a strict subset of forks ? To answer this question, we initialize P_{exc} = [Wait, C(Wait)] ∪ [Forks, c] and $P_{inc} = \varnothing$ where c is an arbitrary color. All the steps until instruction 10 are identical to the former ones.

We find now that rule R2 is applicable for c and its application updates Removed to PC. Then instructions 15-16 delete all places and rules. The algorithm finishes with an empty deadlock, hence returning failure.

It must be noted that the most efficient implementation of Algorithm 2 would have required 2.n (n is the number of philosophers) applications of rules, whereas Algorithm 3 requires only one application !

For a last illustration of Algorithm 3, we test if our first deadlock contains traps.

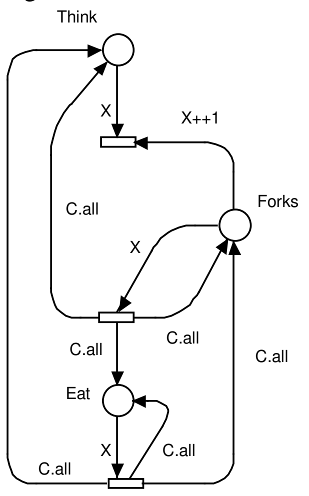

Fig. 4: Transformed net after one application of MR1, MR2 and MR4

So we start our algorithm with the reversed net and with P_{exc} = [Wait, C(Wait)] and $P_{inc} = \emptyset$. The model is presented in Figure 5. The rules become:

```
R1   [Wait, X] ⇒ [Think, X] ∧ [Forks, X++1]
R2   [Eat, X] ⇒ [Wait, X] ∧ [Forks, X]
R3   [Forks, X + X++1] ∧ [Think, X] ⇒ [Eat, X]
```

The circuits are the reverse of the preceding circuits. To_Examine is initialized with R1. Let us look at the application of the meta-rules: MR1 deletes Wait and the rules become:

```
R1   True ⇒ [Think, X] ∧ [Forks, X++1]
R2   [Eat, X] ⇒ [Forks, X]
R3   [Forks, X + X++1] ∧ [Think, X] ⇒ [Eat, X]
```

Now MR3 is applicable on rule R1, the decomposition of the domain is a particular case C = {ε} x C and is detected as variable X appears in the conclusion and does not appear in the hypothesis. Thus the rules become:

```
R1   True ⇒ [Think, C.all] ∧ [Forks, C.all]
R2   [Eat, X] ⇒  [Forks, X]
R3   [Forks, X + X++1] ∧ [Think, X] ⇒ [Eat, X]
```

The loop is executed once with an application of R1; Removed is updated with [Think, C] and [Forks, C], so we test the application of the meta-rules. MR1 deletes places Think and Forks; MR2 deletes rules R1 and R2; MR3 transforms rule R3 in:

```
R3   True ⇒ [Eat, C.all]
```

The second execution of the loop applies rule R3 which updates Removed to PC. The application of meta-rules is detected again and place Eat and rule R3 are deleted. Then the algorithm exits from the loop and returns failure since Deadlock is empty. Hence, the deadlock excluding place Wait we had obtained does not contain a trap.

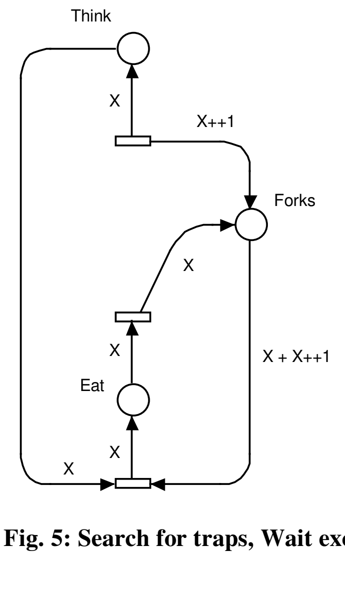

Fig. 5: Search for traps, Wait excluded

It must be noted that the most efficient implementation of Algorithm 2 would have required 2.n (n being the number of philosophers) applications of rules, whereas Algorithm 3 requires only two applications !

4.3 Comparison Between Algorithms 2 and 3

The efficiency of Algorithm 3 compared with Algorithm 2 depends on the number of applications of meta-rules MR3 and MR4 along its execution. So it is difficult to give any theoretical measure of the complexity. In the worst case, the overhead time added to Algorithm 3 is negligible compared with the assignment of the rules, and thus the complexity is in the same order.

However, to estimate the average complexity, we can observe that MR3 becomes applicable as soon as a place conditioning the assignment of a transition disappears, and this happens frequently when the algorithm is applied to colored nets modelling real systems. Moreover MR4 is based on the existence of circuits, which are numerous in the skeleton of a colored net, especially when liveness and boundedness are required. The additional constraints are usually not fulfilled initially. Nevertheless the deletion of places increases the possibility of satisfying the structural condition (no external input places) and the transformation of the color domains by meta-rule MR3 yields the occurrence of the functional condition (existence of identities).

One application of MR3 followed by the application of the transformed rule corresponds to n applications of the initial rule where n is the size of the vanishing color domain. One application of MR4 followed by the application of the transformed rule corresponds to at least one application of all the rules of the circuit. In fact as soon as the output functions of the circuit are different from identity, the reduction factor is proportional to the product of a color domain by the length of the circuit.

We point out that in many cases the computation is parameterized: it remains valid for a family of models where only the size of the color classes changes. Thus the deadlock characterization we perform on the model of the philosophers is independent of the number of philosophers. Such a characterization would have been impossible with Algorithm 2. We now plan to develop a parameterized version of Algorithm 3 for well-formed nets [20].

Last but not least, the results are easier to interpret. Their expression is only a function of the high-level description, and thus uses the same notations that have been given by the designer to describe the model. Unlike our algorithm, Algorithm 2 provides an extensive representation of deadlocks.

All the results can be easily transposed for the detection of traps, as we have shown in the example of the philosophers.

5 Conclusions

In this paper we have presented an efficient algorithm for finding deadlocks and traps in colored nets. The algorithm exploits both the structure of the net and the structure of the color functions. We have shown on an example how the meta-rules speed up the computation of colored deadlocks. The efficiency of the algorithm strongly depends on the number of times meta-rules can be applied. The first experiments have shown that in most cases, the conditions of application are fulfilled. We are now working on the integration of the algorithm in the CASE AMI [21] in order to obtain statistical results on the efficiency of the algorithm.

A forthcoming work is the specialization of this algorithm for syntactically well defined nets, with a complexity almost independent of the size of the color domains. After characterizing classes of colored nets for which some structural property is a necessary or sufficient liveness condition, this algorithm will allow us to decide liveness for such nets.

References

[1] W.Reisig: *Place-Transition Systems*. In Petri Nets: Central models and their properties, W.Brauer, W.Reisig and G.Rozenberg eds., LNCS n° 254, Springer- Verlag, 1987, pp 117-141.

[2] E.W. Mayr: *An Algorithm for the General Petri Net Reachability Problem*. In SIAM. Journal of Computing n° 13, 1984.

[3] E. Best: *Structure Theory of Petri Nets: the Free Choice Hiatus*. In Petri Nets: Central models and their properties, W.Brauer, W.Reisig and G.Rozenberg eds., LNCS n° 254, Springer- Verlag, 1986, pp 168-205.

[4] J. Martinez, M. Silva: *A Simple and Fast Algorithm to Obtain all Invariants of a Generalized Petri Net*. In Informatik Fachberichte n° 52, C.Girault and W.Reisig eds., Springer-Verlag, 1982, pp 301-310.

[5] K.Lautenbach: *Linear Algebraic Techniques for Place / Transition Nets*. In Petri Nets: Central models and their properties, W.Brauer, W.Reisig and G.Rozenberg eds., LNCS n° 254, Springer- Verlag, 1986, pp 142-167.

[6] F. Commoner: *Deadlocks in Petri nets*. In Applied Data Res. Inc., Wakefield, MA, 1972.

[7] J. Esparza, M. Silva: *A Polynomial-Time Algorithm to Decide Liveness of Bounded Free-Choice Nets*. Hildesheimer Informatikberichte 12/90, Institut für Informatik, Univ. Hildesheim.

[8] K.Lautenbach: *Linear Algebraic Calculation of Deadlocks and Traps*. In Concurrency and Nets, K.Voss, H.Genrich and G.Rozenberg eds., Springer Verlag, 1987, pp 315-336.

[9] K.Barkaoui, M.Minoux: *A Polynomial-Time Graph Algorithm to Decide Liveness of Some Basic Classes of Bounded Petri Nets*. In Application and Theory of Petri Nets 92, Proc. of the 13th Conference, K. Jensen ed., LNCS n° 616, Springer-Verlag, Sheffield, UK, 1992, pp 62-74.

[10] H.J. Genrich: *Predicate / Transition Nets*. In High-level Petri Nets. Theory and Application, K. Jensen and G. Rozenberg eds., Springer-Verlag, 1991, pp 3-43.

[11] K. Jensen: *Coloured Petri Nets: A High Level Language for System Design and Analysis*. In High-level Petri Nets. Theory and Application, K. Jensen and G. Rozenberg eds., Springer-Verlag, 1991, pp 44-119.

[12] J. Ezpeleta, J.M. Couvreur: *A New Technique for Finding a Generating Family of Siphons, Traps and ST-Components. Application to Colored Petri Nets*. In proc. of the 12th International Conference on Application and Theory of Petri Nets, Gjern, Denmark, June 1991, pp 145-164.

[13] J.M. Couvreur, S. Haddad, J.F. Peyre: *Computation of Generative Families of Positive Flows in Coloured Nets*. In proc. of the 12th International Conference on Application and Theory of Petri Nets, Gjern, Denmark, June 1991.

[14] J.F. Peyre: *Résolution Paramétrée de Systèmes Linéaires. Application au Calcul d'Invariants et à la Génération de Code Parallèle*. Thèse de l'Université Paris 6, March 1993 (in French).

[15] M. Minoux, K. Barkaoui: *Deadlocks and Traps in Petri Nets as Horn-Satisfiability Solutions and some Related Polynomially Solvable Problems*. Discrete Applied Mathematics n° 29, 1990.

[16] J. Vautherin: *Parallel Systems Specifications with Coloured Petri Nets and Algebraic Specifications*. In Advances in Petri Nets 1987, Springer-Verlag, 1987, pp 293-308.

[17] G. Findlow: *Obtaining Deadlock-Preserving Skeletons for Coloured Nets,* in Application and Theory of Petri Nets 92, Proc. of the 13th Conference, K. Jensen ed., LNCS n° 616, Springer-Verlag, Sheffield, UK, 1992, pp 173-192.

[18] D.B. Johnson: *Finding all Elementary Circuits of a Directed Graph*, SIAM J.Computer, vol.4, n° 1, 1975.

[19] S. Haddad: *A Reduction Theory for Coloured Nets*. In High-level Petri Nets. Theory and Application, K. Jensen and G. Rozenberg eds., Springer-Verlag, 1991, pp 399-425.

[20] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad: *Stochastic Well-Formed Nets and Symmetric Modeling Applications*, to appear in IEEE Transactions on Computers.

[21] J.M. Bernard, J.L. Mounier, N. Beldiceanu, S. Haddad: *AMI an Extensible Petri Net Interactive Workshop*, Proc. of the 9th European Workshop on Application and Theory of Petri Nets, Venice, Italy, June 1988.