Monotonic Extensions of Petri Nets : Forward and Backward Search Revisited

A. Finkel[1], J.-F. Raskin[2], M. Samuelides[1], L. Van Begin[2]

[1] Ecole Normale Supérieure de Cachan, France
[2] Université Libre de Bruxelles, Belgium

Abstract. In this paper, we revisite the forward and backward approaches to the verification of extensions of infinite state Petri Nets. As contributions, we propose an adaptation of the Covering Sharing Trees to compute symbolically the minimal coverability set of Petri Nets, we identify a subclass of Transfer Nets for which the forward approach generalizes and we propose a general strategy to use both the forward and the backward approach for the efficient verification of general Transfer Nets.

1 Introduction

Model-checking techniques have proven usefull for the verification of finite state abstractions of various concurrent and distributed computer systems. Unfortunately, usefull finite state abstractions are often difficult to obtain from the concrete systems to verify. As a consequence, a lot of efforts have been made recently to extend the successfull techniques for model-checking finite state systems to infinite state systems and parametric verification (where we try at once to verify an infinite familly of finite state systems). A lot of interesting theoretical results have been obtained, see for example [ACJT96,BM99,EFM99,FS01]. Nevertheless, a lot of work remains to be done to turn those positive theoretical results into practical verification algorithms.

In parametric verification, we want to verify at once an entire familly of systems. For example, some mutual exclusion protocols have been designed to work for any number of processes that want to share common ressources. Then we would like that our automatic verification methods allow the automatic verification of the protocol for any number of processes. In that context, several abstraction have proven to be useful, see for example [ABB01,BBLS00,GS92]. The work in this paper is directly usefull in the context of the so-called *counting abstraction*. When considering the counting abstraction, the model of (infinite) *Petri Nets* and its extensions, like *transfer Nets* and *Reset Nets*, are particularly important. In this paper, we will discuss efficient techniques to analyze infinite state Petri Nets and Tranfer Nets (and so Reset Nets are that are a subclass of Transfer Nets).

There are basically two different approaches to the verification of *upward closed properties* of Petri Nets. The first is the *forward approach* (that was first defined by Karp and Miller [KM69]). This approach starts from the (possibly parametric) set of initial markings and computes an approximation of the infinite iteration of the transition relation (often refered as the Post relation) over markings defined by the Petri net. That over-approximation is sufficiently precise to completely answer interesting questions about Petri Nets. One of the most important class of properties that this method can answer are the so-called *safety properties*, i.e. can the Petri net ever reach a set of *bad markings*, with the restriction that the set of bad markings is *upward closed*. That result allows us *in theory* to automatically answer any *mutual exclusion* property for example. The *backward approach* represents an alternative to the forward approach for the verification of such properties. The backward approach consists in itering from the set of Bad states the Pre relation (which is the inverse of the Post relation). If the limit of the iteration of the Pre operator intersects with the (possibly parametric) set of initial markings then we know that some Bad markings are reachable. The iteration of the Pre relation is guaranteed to terminated if the set of Bad markings is upward closed [ACJT96,FS01]. Unfortunately, in the two cases, a *naive* implementation of the abstract algorithm is not practical. It is not supprising as we know that the *theoretical complexity* of this verification is *very high* for that class of infinite state models, see [Sch02].

So further research was necessary to obtain more practically usefull verification techniques. For the forward approach, we have defined in [Fin93] an heuristic that try to *minimize* the set of markings that has to be considered when computing the Karp-Miller covering tree. For the backward approach, we have defined in [DR00] a BDD-like structure that allow us to *compactly* represent the infinite sets that are generated during the iteration of the Pre relation. The resulting algorithm for the backward search is *symbolic* in the sense that (minimal) markings are never enumerated during the computation where all the operations involved in the algorithm always directly treat the underlying structure that represents the set. An implementation of the algorithm has shown that the symbolic algorithm is much more effective than the naive enumerative approach [DRV01].

In this paper, as a first contribution, we show how to implement symbolically, using a variant of the data structure that we have defined in [DR00], the enumerative algorithm that we have defined in [Fin93] to compute the minimal coverability set of an infinite state Petri net. As we have now two symbolic algorithms, we are able to make a *fair* comparison between the relative practical merits of the two approaches.

A main advantage of the backward search is that it is *robust* in the following sense: it is not only applicable to the basic class of infinite Petri Nets but also to all the extensions of that model that preserve *monotonicity*. Transfer Nets (and so broadcast protocols that they generalize) and Reset Nets maintain monotonicity for example. Unfortunately, the forward approach does *not* extend for those models. There are negative results [DFS98] that show us that it is not always possible to compute the coverability set for those extensions. Nevertheless, as a second contribution, we show in this paper that we can compute forwardly a weak version of the coverability set for an interesting subclass of Transfer Nets. That weak version allow us to decide the safety properties that can be expressed as upward closed sets of markings. The subclass of transfer that we have identified is of practical interest as it covers all the examples of abstraction of multi-threated JAVA programs that we have analyzed backwardly in [DRV02]. The advantage of the forward approach is that it starts from the set of initial markings and usually generate sets that are more structured than the one generated with the backward search.

In [DRV01], we have shown that the efficiency of the backward search can be improved substantially by using rough approximations of the forward reachable states in order to guide the search towards initial markings. In our previous works, the over-approximation is obtained automatically by computing the structural invariants of the net. As a third contribution, we propose here to use the symbolic implementation of the minimal coverability set to compute an other over-approximation of the forward reachable states of general Transfer Nets that do not fall in the class of models that we have identified. If the over-approximation is *too large* to give an conclusive answer to the safety verification problem, we propose to use this over-approximation to guide the exact backward search. The information collected during this over-approximation is potentially (and often much) richer than the one computed with the invariants. Those heuristics seems necessary to attach those verification problems that have very high theoretical complexities [Sch02].

Structure of the paper . In section 2, we introduce the model of Multi Transfer Nets that contains Petri Nets as a subclass. We also recall there notions about upward closed sets, downward closed sets and convering sets. In section 3, we show how to represent efficiently infinite downward closed sets with a graph based data structure. Section 4 presents a symbolic algorithm to compute the minimal coverability tree of a Petri Net. We have implemented this symbolic algorithm and used our graph based structure to represent the downward closed sets it manipulates. We report in section 5 on the practical behavior of our new symbolic forward algorithm and compare its performances with a symbolic algorithm working backwardly that we have defined and implemented in previous works. In section 6, we identify an interesting subclass of Multi Transfer Nets for which the forward search can be extended, we call this class the Multi Isolated Transfer Nets. In section 7, we suggest the cooperative use of a forward approximation and the backward search for the full class of Multi Transfer Nets.

2 Petri Nets and Multi Transfer Nets

In this section, we define Multi Transfer Nets (MTNs for short), an extension of Petri Nets with fairly general transfers. That extension maintains the monotonicity property of Petri Nets. Petri Nets are defined as a subclass of MTNs.

Definition 1 (Multi Transfer Net). A *Multi Transfer Net* is a pair $\langle \mathcal{P}, \mathcal{B} \rangle$ where: $\mathcal{P} = \{p_1, \ldots, p_n\}$ is a set of places, and $\mathcal{B} = \{M_1, \ldots, M_m\}$ is a set of *multi transfers*. A *multi transfer* M is a tuple $\langle T, \{B_1, \ldots, B_u\} \rangle$ such that

- $T = \langle \mathcal{I}, \mathcal{O} \rangle$ is the Petri Net *transition* of the *multi transfer*: $\mathcal{I}, \mathcal{O} : \mathcal{P} \to \mathbb{N}$;
- each $B_i = \langle P_i, p_i \rangle$ with $P_i \subseteq \mathcal{P}$ (a set of source places) and $p_i \in \mathcal{P}$ (a target place) is a *transfer*.

In order to avoid cyclic transfers, a multi transfer M with set of transfers $\{B_1, \ldots, B_u\}$ must satisfy the following conditions:

1. for any B_i, we require that $p_i \notin P_i$;
2. for any *transfers* B_i and B_j with $B_i \neq B_j$, we require that $(P_i \cup \{p_i\}) \cap (P_j \cup \{p_j\}) = \emptyset$.

A Petri Net is a MTN where each multi transfer contains an empty set of transfer (then the multi transfer is just a plain Petri Net transition).

A marking $\mathbf{m} : \mathcal{P} \to \mathbb{N}$ is a function which assigns a value $c \in \mathbb{N}$ to each places. That function can equivalently be seen as a vector of size $|\mathcal{P}|$. We define $\preceq$ on markings such that $\mathbf{m} \preceq \mathbf{m}'$ iff $\mathbf{m}(p) \leq \mathbf{m}'(p), \forall p \in \mathcal{P}$. We note $\mathbf{m}(P) = \sum_{p \in P} \mathbf{m}(p)$.

Definition 2 (Multi Transfer-Enabling). Let M be a multi-transfer with Petri Net *transition* $\langle \mathcal{I}, \mathcal{O} \rangle$. We say that M is *enabled* in $\mathbf{m}$ if $\mathcal{I} \leq \mathbf{m}$.

Definition 3 (Multi Transfer-Firing). Let $M = \langle T, \{B_1, \ldots, B_u\} \rangle$ be a multi transfer enabled in $\mathbf{m}$. *Firing* M in $\mathbf{m}$ leads to the marking $\mathbf{m}'$ (written $\mathbf{m} \rightarrowtail_M \mathbf{m}'$). To define $\mathbf{m}'$, we need to define the intermediary markings $\mathbf{m}_1$ and $\mathbf{m}_2$:

- $\mathbf{m}_1 = \mathbf{m} - \mathcal{I}$. That is, we first remove the tokens needed by the Petri Net part of the transition;
- then we define $\mathbf{m}_2$ as follows:
  - for any place p which is target of a transfer, i.e. $p = p_i$ for some $1 \leq i \leq u$, we have $\mathbf{m}_2(p) = \mathbf{m}_1(p) + \mathbf{m}_1(P_i)$. That is, we transfer all the remaining tokens from the sources to the target;
  - for any place p which is source of a transfer, i.e. $p \in P_i$ for some $1 \leq i \leq u$, we have $\mathbf{m}_2(p) = 0$;
  - for all other places p (which are neither the source or the target of a transfer), we have : $\mathbf{m}_2(p) = \mathbf{m}_1(p)$;
- Finally, $\mathbf{m}'$ is obtained from $\mathbf{m}_2$ by adding the tokens produced by the Petri Net part of the transition, that is: $\mathbf{m}' = \mathbf{m}_2 + \mathcal{O}$.

We now define the semantics of MTNs.

Definition 4 (Operational Semantics). Let $\mathcal{M} = \langle \mathcal{P}, \mathcal{B} \rangle$ be an MTN. A *run* of $\mathcal{M}$ is a sequence of markings $\mathbf{m}_0\mathbf{m}_1 \ldots \mathbf{m}_n$ such that for any i, $0 \leq i < n$, there exists $M \in \mathcal{B}$ such that $\mathbf{m}_i \rightarrowtail_M \mathbf{m}_{i+1}$, $\mathbf{m}_0$ is the *initial* marking of the run and $\mathbf{m}_n$ the *target* marking of the run. A marking $\mathbf{m}'$ is *reachable* from a marking $\mathbf{m}$, written $\mathbf{m} \rightarrowtail^* \mathbf{m}'$, if and only if there exists a run with initial marking $\mathbf{m}$ and target marking $\mathbf{m}'$. The *set of reachable markings* of $\mathcal{M}$ from a set of markings S_0, written $\mathsf{Reach}(\mathcal{M}, S_0)$, is defined as the set $\{\mathbf{m}' \mid \exists \mathbf{m} \in S_0 : \mathbf{m} \rightarrowtail^* \mathbf{m}'\}$.

Given an MTN $\mathcal{M}$, a set of initial markings S_0 and a set of Bad markings U, we are interested by the following two decision problems:

1. "Can the MTN $\mathcal{M}$ reach a Bad marking in U starting from an marking in S_0 ?", i.e. $\mathsf{Reach}(\mathcal{M}, S_0) \cap U =^? \emptyset$. This is called the *safety verification problem.*
2. "Is there a bound $c \in \mathbb{N}$ for the place p such that in any reachable marking of $\mathcal{M}$ starting from S_0, the marking of p does not exceed c ?", i.e. $\forall \mathbf{m} \in \mathsf{Reach}(\mathcal{M}, S_0) : \mathbf{m}(p) \leq c$?. [1] This is called the *place boundedness problem.*

[1] Let us note that the weaker question "Is $c \in \mathbb{N}$ a bound for the place p ?" is a particular case of the first problem.

Before going further, we need some more notations. We define a special value ω. For any $c \in \mathbb{N}$, we have $c < \omega$, and furthermore we have $\omega + c = \omega - c = \omega, \forall c \in \mathbb{N} \cup \{\omega\}$. We extend markings to ω-markings which assigns a value $c \in \mathbb{N} \cup \{\omega\}$ to each places. The $\preceq$ relation on markings is extended to ω-markings in the obvious way.

Definition 5 (Least Upper Bound). The *least upper bound (lub)* of a set of markings $\{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$ is the marking $\mathbf{m}$ such that:

$$\begin{cases} \mathbf{m}(p) = \omega & \textbf{if } \forall c \in \mathbb{N} : \exists i \geq 0 : \mathbf{m}_i(p) > c \\ \mathbf{m}(p) = max(\mathbf{m}_1(p), \mathbf{m}_2(p), \ldots) & \textbf{otherwise}. \end{cases}$$

We say that a set of markings S is *downward closed* iff we have that:

$$\forall \mathbf{m} : (\mathbf{m} \in S \to \forall \mathbf{m}' \leq \mathbf{m} : \mathbf{m}' \in S).$$

Symetrically, we say that a set of markings S is *upward closed* iff we have that:

$$\forall \mathbf{m} : (\mathbf{m} \in S \to \forall \mathbf{m}' \geq \mathbf{m} : \mathbf{m}' \in S).$$

Given a upward closed set S, we note $\mathsf{Min}(S)$ the set of minimal elements defined as:

$$\mathsf{Min}(S) = \{\mathbf{m} \in S \mid \neg\exists \mathbf{m}' \in S : \mathbf{m}' < \mathbf{m}\}.$$

Given a downward closed set S, we note $\mathsf{Lim}(S)$ the set of its limits elements defined as:

$$\begin{aligned} \mathsf{Lim}(S) = \{m \mid m = \mathbf{lub}(\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n, \ldots\}) \textbf{ and } \mathbf{m}_1 < \mathbf{m}_2 < \ldots < \mathbf{m}_n < \ldots \in S\} \\ \cup \{\mathbf{m} \mid \neg\exists \mathbf{m}' \in S : \mathbf{m}' > \mathbf{m}\}. \end{aligned}$$

In $\mathbb{N}^k$, any upward closed set S is identified by its finite set of minimal elements and any downward closed set S is idenfied by its finite set of limit elements (that are vectors potentially with ωs). We are now equipped to define the notion of *coverability set* which is an important tool to answer the decision problems that we have mentioned above.

Definition 6 (Coverability Set [Fin93,Fin90]). A *coverability set* for a MTN $\mathcal{M}$ and a set S_0 of initial markings, noted $\mathtt{CS}(\mathcal{M}, S_0)$, is a set of ω-markings such that

(1) for every $\mathbf{m} \in \mathtt{CS}(\mathcal{M}, S_0)$, there exists an infinite sequence $\mathbf{m}_1 \preceq \mathbf{m}_2 \preceq \mathbf{m}_3 \ldots$ with $\mathbf{m}_i \in \mathtt{Reach}(\mathcal{M}, S_0)$ such that $\mathbf{m} = lub(\{\mathbf{m}_1, \mathbf{m}_2, \ldots\})$.
(2) $\forall \mathbf{m} \in \mathtt{Reach}(\mathcal{M}, S_0)$, there exists $\mathbf{m}' \in \mathtt{CS}(\mathcal{M}, S_0)$ such that $\mathbf{m} \preceq \mathbf{m}'$.

In all the coverability sets, there is an interesting one which is called the minimal coverability set.

Definition 7 (Minimal Coverability Set [Fin93,Fin90]). The *minimal coverability set* for a Petri Net $\mathcal{M}$ for a set S_0 of initial markings is the intersection of all the finite coverability sets of $\mathcal{M}$ for S_0.

We have shown in a previous work [Fin93] that the minimal coverability set is unique and can be computed effectively for any Petri Net $\mathcal{M}$ and any finite set of initial markings S_0.

Given a set of ω-markings, we define its downward closure as follows:

Definition 8 (Downward Closure). Let S be a set of ω-markings, the *downward closure* of S, noted $\downarrow S$, is the set of markings $\{\mathbf{m} | \exists \mathbf{m}' \in S \text{ and } \mathbf{m} \preceq \mathbf{m}'\}$

Let us now recall some properties of any coverability set, see [Fin93] for details.

1. any $\mathtt{CS}$ is an approximation of the reachable markings in the following sense :

$$\mathtt{Reach}(\mathcal{M}, S_0) \subseteq \downarrow \mathtt{CS}(\mathcal{M}, S_0)$$

2. any $\mathtt{CS}$ is sufficient to answer any safety verification problem if the set of Bad markings U is upward closed. In fact, we have:

$$\mathtt{Reach}(\mathcal{M}, S_0) \cap U = \emptyset \textbf{ iff} \downarrow \mathtt{CS}(\mathcal{M}, S_0) \cap U = \emptyset$$

3. any $\texttt{CS}$ is sufficient to give an answer to the place boundedness problem. A place p is bounded in $\texttt{Reach}(\mathcal{M}, S_0)$ **iff** there does not exist a ω-marking $\mathbf{m} \in \texttt{CS}$ such that $\mathbf{m}(p) = \omega$.

In [KM69], Karp and Miller give an algorithm to compute a coverability set for any Petri Net with a downward closed set of initial markings S_0. In [Fin93], we have defined a heuristic that computes a minimal coverability set. Basically, the algorithm constructs incrementally a coverability tree (as in [KM69]) but where all the nodes are labelled with limit elements of a downward closed underlying set. Our algorithm in [Fin93] is enumerative. In the next section, we adapt a data structure to compactly represent downward closed set of markings. The data structure will be used later in section 4 to obtain a symbolic version of our algorithm to compute the minimal coverability set of a Petri Net.

3 Downward Closed Covering Sharing Trees

The motivation of this section is to define a way to compactly represent (possibly infinite) downward closed sets of markings. For that we start from Sharing Trees (STs) that are data structures introduced in [ZL94] to efficiently store tuples of integers. A sharing tree $\mathcal{S}$ is a rooted acyclic graph with nodes partitioned in *k-layers* such that: all nodes of layer i have successors in the layer $i+1$; a node cannot have two successors with the same label; finally, two nodes with the same label in the same layer do not have the same set of successors. Formally, $\mathcal{S}$ is a tuple $(N, V, root, end, val, succ)$, where $N = \{root\} \cup N_1 \cup \ldots \cup N_k \cup \{end\}$ is the finite set of *nodes* (N_i is the set of nodes of *layer* i and, by convention, $N_0 = \{root\}$ and $N_{k+1} = \{end\}$), $V = \{x_1, x_2, \ldots, x_k\}$ is a set of variables. Intuitively, N_i is associated to x_i. $val : N \rightsquigarrow \mathbb{Z} \cup \{\top, \bot\}$ is a labeling function for the nodes, and $succ : N \rightsquigarrow 2^N$ defines the successors of a node. Furthermore, (1) $val(n) = \top$ iff $n = root$, $val(n) = \bot$ iff $n = end$, $succ(end)=\emptyset$; (2) for $i : 0, \ldots, k$, $\forall n \in N_i$, $succ(n) \subseteq N_{i+1}$ and $succ(n) \neq \emptyset$; (3) $\forall n \in N$, $\forall\, n_1, n_2 \in succ(n)$, if $n_1 \neq n_2$ then $val(n_1) \neq val(n_2)$. (4) $\forall i, 0 \leq i \leq k$, $\forall n_1, n_2 \in N_i$, $n_1 \neq n_2$, if $val(n_1) = val(n_2)$ then $succ(n_1) \neq succ(n_2)$. A path of a k-sharing tree is a sequence of nodes $\langle n_1, \ldots, n_m \rangle$ such that $n_{i+1} \in succ(n_i)$ for $i = 1, \ldots, m$-1. Paths represent *tuples of size* k of natural numbers. We use $elem(\mathcal{S})$ to denote the *flat denotation* of a k-sharing tree $\mathcal{S}$:

$$elem(\mathcal{S}) = \{ \ \langle val(n_1), \ldots, val(n_k) \rangle \mid \langle \top, n_1, \ldots, n_k, \bot \rangle \text{ is a path of } \mathcal{S} \ \}.$$

Conditions (3) and (4) ensure the maximal sharing of prefixes and suffixes among the tuples of the flat denotation of a sharing tree. The *size* of a sharing tree is the number of its *nodes* and *edges*. The number of tuples in $elem(\mathcal{S})$ can be exponentially larger than the size of $\mathcal{S}$. As shown in [ZL94], given a set of tuples $\mathcal{A}$ of size k, there exists a unique sharing tree such that $elem(\mathcal{S}_\mathcal{A}) = \mathcal{A}$ (modulo isomorphisms of graphs). Given two finite sets of tuples of integers represented as two STs $\mathcal{S}_1$ and $\mathcal{S}_2$, there are polynomial time algorithms in the size of the two STs to compute $\mathcal{S}_3$ such that $\mathcal{S}_3 = \mathcal{S}_1 \cap \mathcal{S}_2$, $\mathcal{S}_3 = \mathcal{S}_1 \cup \mathcal{S}_2$ and $\mathcal{S}_3 = \mathcal{S}_1 \setminus \mathcal{S}_2$ and a polynomial algorithm to decide if $\mathcal{S}_1 \subseteq \mathcal{S}_2$. But STs can only represent finite sets. In [DRV02b], we have proposed to use an extension of that data-structure to represent infinite upward-closed sets of markings. We adapt here this idea in order to represent (potentially infinite) downward closed sets of markings. We know that a (potentially infinite) downward closed set of markings (or more generally set of natural number vectors) is identified by its finite set of limit elements. This set is a finite set of ω-markings. We will use ST to represent this finite set of ω-markings. We define downward closed covering ST as ST with $val : N \rightarrow \mathbb{N} \cup \{\top, \bot\} \cup \{\omega\}$. A downward closed covering ST $\mathcal{S}$ has the following semantics :

$$[\![\mathcal{S}]\!] = \{\mathbf{m} \in \mathbb{N}^k \,|\, \exists \mathbf{m}' \in elem(\mathcal{S}) : \mathbf{m} \preceq \mathbf{m}'\}$$

So, $[\![\mathcal{S}]\!]$ is the downward closure of $elem(\mathcal{S})$, i.e. $[\![\mathcal{S}]\!] = \downarrow elem(\mathcal{S})$. We will say that a downward closed CST (dcCST) $\mathcal{S}$ is irredundant if $\neg \exists \mathbf{m}_1, \mathbf{m}_2 \in elem(\mathcal{S})$ with $\mathbf{m}_1 \prec \mathbf{m}_2$. In our experience, it is often better to keep $\mathcal{S}$ irredundant. It is easy to show with straigtforward adaptations of proofs from [DRV02] (where we define CST to represent infinite upward-closed sets) that (i) there is no polynomial time algorithm (unless P = NP) to decide $[\![\mathcal{S}_1]\!] \subseteq [\![\mathcal{S}_2]\!]$ where $\mathcal{S}_1, \mathcal{S}_2$ are two dcCSTs, (ii) there is no polynomial time (unless P = NP) to compute from $\mathcal{S}_1$ an irredundant dcCST $\mathcal{S}_2$ such that $[\![\mathcal{S}_1]\!] = [\![\mathcal{S}_2]\!]$. We will show that those negative theoritical results seems not to be a pratical obstacle to the use of dcCST in a symbolic algorithm computing the minimal coverability set. The algorithm using dcCSTs is given in the next section. Practical evaluation of the algorithm is given in section 5.

4 Symbolic Computation of the Minimal Covering Set for PN

To compute a coverability set for a Petri Net, we generally construct what we call a *coverability graph (tree)* (see [KM69]). This is mainly a graph (tree) where the nodes are labelled by the maximal or limit elements of a coverability set and the edges are an approximation of the successor relation between the ω-markings labelling the nodes. Unfortunately, the procedure presented in [KM69] computes unmanagable graphs, even for small Petri Nets. An efficient heuristic is presented in [Fin93]. It consists to construct the minimal coverability tree, which is a tree where the value of the nodes correspond to the minimal coverability set of the Petri Net. The idea is as follows. At each step of the construction of the tree, an untreated node annotated with the ω-marking $\mathbf{m}$ is developped by computing its successors (at the beginning, untreated nodes corresponds to initial markings). For each successors $\mathbf{m}'$ of $\mathbf{m}$, we have four cases :

(1) there is an already computed ω-marking $\mathbf{m}''$ such that $\mathbf{m}' \preceq \mathbf{m}''$, then $\mathbf{m}'$ is forgotten.
(2) there is at least one path in the tree from a marking $\mathbf{m}''$ to $\mathbf{m}$ such that $\mathbf{m}'' \preceq \mathbf{m}'$, then we take the largest such path, remove the subtree starting from $\mathbf{m}''$ (without removing $\mathbf{m}''$) and replace $\mathbf{m}''$ by $\mathbf{n}$ where $\mathbf{n}(p) = \mathbf{m}''(p)$ if $\mathbf{m}''(p) = \mathbf{m}'(p)$, ω otherwise (see Fig. 1).
(3) there is an already computed marking $\mathbf{m}''$ which is not a predecessor of $\mathbf{m}$ such that $\mathbf{m}'' \preceq \mathbf{m}'$ then the subtree rooted by $\mathbf{m}''$ is removed, a new node connected to the node of $\mathbf{m}$ and annotated with $\mathbf{m}'$ is added to the tree (see Fig. 2). Note that we check for this case even if we first fall in the previous one.
(4) in the other cases, a new node connected to the node of $\mathbf{m}$ and annotated with $\mathbf{m}'$ is added to the tree.

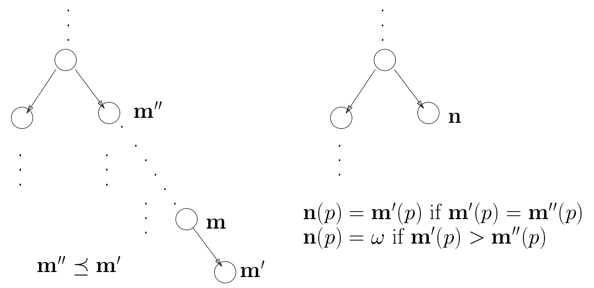

Fig. 1. ω detection and subtree removal.

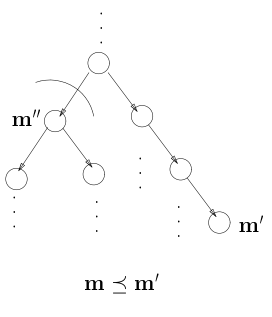

Fig. 2. Removing of subtrees.

An improvement of this algorithm is to symbolically treat the sets of untreated markings instead of enumerating them. Symbolically means that computation on sets are done by making some global computations on the data structures used to represent them. To make this possible, we first define a set-based version of the

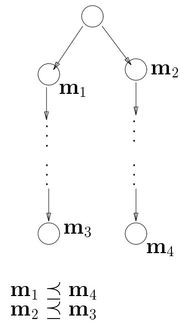

Fig. 3. Problematic case when subtrees are removed.

algorithm of [Fin93]. Fig. 11 in appendix shows such an high-level algorithm manipulating sets. In [Fin93], the minimal coverability tree is only constructed to manage case (2) and (3). But note that for these cases, we only need the reachability relation between ω-markings rather than the direct successor relation of the tree. For this reason, we do not compute a coverability tree in the symbolic algorithm: we will not maintain the successor relation as in the tree but only maintain the closure of that relation. It is easy to see that this is sufficient for the case (2) and case (3) of the algorithm.

The three cases of the enumerative algorithm are replaced by operations on downward closed sets of markings. The algorithm removing the non-maximal markings from the frontier (case (1) in the description of the algorithm) is presented in Fig. 12 in appendix, replacing value by ω (case 2) is shown in Fig. 13 in appendix and the algorithm pruning subtrees (case 3) is presented in Fig. 15 in appendix. In this last case, the set-based version of the algorithm introduces some complications. As we compute all the successor nodes of the untreated ω-markings at once, the situation of Fig. 3 can occur. The node $\mathbf{m}_4$ can be used to supress the subtree at $\mathbf{m}_1$, while $\mathbf{m}_3$ can be used to suppress the subtree rooted at $\mathbf{m}_2$. Note that this situation can not occur in the enumerative version of the algorithm as either $\mathbf{m}_3$ is enumerated first and the subtree rooted at $\mathbf{m}_2$ is suppressed, and so $\mathbf{m}_4$ is never generated. If $\mathbf{m}_4$ is enumerated first, the situation is symetric. In the set-based case, we have to detect this kind of cycle and keep one of the subtree (see algorithm of Fig. 14) in appendix.

As previously explained, the size of dcCST can be logarithmic in the size of the set of limit elements it represents. In this way, we could have an exponential gain using dcCST to represent sets of ω-markings and the over-approximation of the successor relation and by symbollically computing operations on sets directly on the graph structure of the dcCST.

5 Comparison With Backward Approach

5.1 Conceptual Comparison

In this section, we recall some facts about the forward and backward approach for the verification of extensions of unbounded Petri Nets.

Forward	**Backward**
Downward closed Sets	Upward closed Sets
Over-approximation of successors	Exact set of predecessors
Acceleration	No Acceleration
Not Robust	Robust
Safety,Place boundedness	Safety
Depends on initial markings	Depends on bad markings

Fig. 4. Conceptual differences between forward and backward search.

Sets. Starting from an upward closed set, applying Pre preserve the upward closure of the set. So, the backward search manipulates upward closed sets. As we have seen, a coverability set can be viewed as representing a downward closed set of markings. In this point of view, forward approach manipulates a downward closed sets.

Approximation of the computation. The backward approach computes the exact set of predecessors of an upward closed set of bad markings and the forward approach computes an approximation of the reachable markings that is still precise enough to verify safety properties expressed as upward closed sets.

Techniques to garantee termination. By applying the Pre operation, it is garanteed to reach a fixpoint after a finite number of iterations. Forward approach needs an acceleration function (putting new ω in the case 3 of the algorithm presented in section 4) to reach a fixpoint when the net is unbounded.

Robustness. Backward approach is robust for extensions of Petri Nets that maintain monotonicity of the model, on the other hand forward approach cannot be extended for those natural extensions of Petri Nets.

Properties. Only covering properties can be decided with the backward algorithm but in addition *place boundness* is known to be decidable with the forward approach.

Dependance of the search. When computing the coverability tree, we start from the initial markings and compute the coverability set that can be used to verify any safety property. On the other hand, with the backward approach, the computation depends on the property to verify. If $\mathsf{Pre}^*(U)$ does not intersect with the set of initial markings then no set computed during the the fixpoint iterations contain a reachable marking. A priori, this last remark pleads for the forward approach.

5.2 Practical Comparison

We applied our symbolic algorithm on some parametrized Petri Nets [2]. Results are shown in Fig. 5. We also verify these models with our symbolic backward algorithm defined in [DRV01]. This algorithm uses structural invariants ([STC98]) to prune the backward search space. But in these examples, this heuristics doesn't work well (for example, – means in the table that no invariant can be used), except for the PNCSA example. In all the cases, the forward fixpoint is always smaller than the backward fixpoint (pruned with invariants). This has a direct implication on execution times which are in most of the cases smaller in the forward case. Further, if we do not use the structural invariants heuristic to prune the backward search then we see that execution time for the backward search are not satisfactory. Unfortunately, usefull structural invariants does not always exist.

6 A Weak Extension of the MCS

As we have seen in the last section, there are conceptual avantages to use the forward approach and practical evidences that plead for the forward approach. Unfortunately, we know that computing a coverability set for extensions of Petri Nets is not always possible [DFS98]. In this section, we identify a subclass of MTNs for which we can compute a weaker notion of coverability set. We call this class Multi Isolated Transfer Nets as the restriction is that any two transfers do not share places. This class is of practical importance as it covers all the example of JAVA programs that we have analyzed with the backward approach in [DRV02b].

This section is organized as follows. We first formally define the subclass of Multi Isolated Transfer Nets. Then we define the weaker notion of coverability set that we can construct for this class of systems and which is sufficient to verify any safety property expressed as an upward closed set of markings. We then define an algorithm to compute this weak coverability set and illustrate its behavior on a simple example.

[2] see the web page `http://www.ulb.ac.be/di/ssd/lvbegin/CST/index.html`

Case Study	P	T	$\mathbf{I}_{\mathsf{Post}}$	$\mathbf{NN}_{\mathsf{Post}}$	$\mathbf{NE}_{\mathsf{Post}}$	$\mathbf{time}_{\mathsf{Post}}$	$\mathbf{I}_{\mathsf{Pre}}$	$\mathbf{NN}_{\mathsf{Pre}}$	$\mathbf{NE}_{\mathsf{Pre}}$	$\mathbf{time}^1_{\mathsf{Pre}}$	$\mathbf{time}^2_{\mathsf{Pre}}$
PNCSA	31	36	56	100	80	1.30s	17	176	89	16.42s	2.5s
PNCSA	31	36	-	-	-	-	32	2694	1661	>3h	43.79s
Kanban	16	16	7	18	1	0.03s	59	720	432637	1353s	-
CSM	14	13	11	35	16	0.04s	11	109	152	0.1s	0.07s
FMS	22	20	16	36	24	0.12s	46	881	1695	6.13s	5.98s
mesh2x2	32	32	8	78	256	0.21s	15	344	427	0.8s	0.8s
mesh3x2	52	54	11	91	720	0.7s	21	987	2224	6.5s	6.5s
multipool	18	21	12	104	220	0.2s	18	196	5641	2.1s	-

Fig. 5. Benchmarks on an AMD Athlon 900Mhz 500Mbytes: **P**=n. places, **T**=n. transitions; $\mathbf{I}_{\mathsf{Post}}$=n. iterations before reaching the forward fixpoint; $\mathbf{NN}_{\mathsf{Post}}$=nodes of the sharing tree describing the minimal coverability set; $\mathbf{NE}_{\mathsf{Post}}$=cardinality the minimal coverability set; $\mathbf{time}_{\mathsf{Post}}$=ex.time to compute the minimal coverability set; $\mathbf{I}_{\mathsf{Pre}}$=n. iterations before reaching the forward fixpoint without invariants; $\mathbf{NN}_{\mathsf{Pre}}$= nodes of the sharing tree describing the backward fixpoint without invariants; $\mathbf{NE}_{\mathsf{Pre}}$=cardinality of the backward fixpoint without invariants; $\mathbf{time}^1_{\mathsf{Pre}}$=ex. time to compute the backward fixpoint without invariants; $\mathbf{time}^2_{\mathsf{Pre}}$=ex. time to compute the backward fixpoitn with invariants.

6.1 Multi Isolated Transfer Nets

Given a multi transfer $M = \langle T, \{B_1, B_2, \ldots, B_u\}\rangle$, we use the notation $\mathsf{Transfer}(M)$ to designate its set of transfers $\{B_1, B_2, \ldots, B_u\}$. Remember that each B_i is a pair $\langle P_i, p_i\rangle$, where P_i is the set of sources and p_i the target of the transfer B_i.

Definition 9. A *Multi Isolated Transfer Net* $\mathcal{M} = \langle \mathcal{P}, \mathcal{B}\rangle$ is an MTN satisfying the following additional conditions, expressed informally as follows :

(i) a place can not be a source of two different transfers;
(ii) A place can not be the target of one transfer and source in another transfer;
(iii) a place source of a transfer can not be source of the Petri Net part of a transition.

and formally as follows :

(i) $\forall p \in \mathcal{P} : \neg\exists M_i, M_j \in \mathcal{B}$ with $\langle P_i, p_i\rangle \in \mathsf{Transfer}(M_i)$, $\langle P_j, p_j\rangle \in \mathsf{Transfer}(M_j)$, $(\langle P_i, p_i\rangle \neq \langle P_j, p_j\rangle)$, and such that $p \in P_i$ and $p \in P_j$.
(ii) $\neg\exists p \in \mathcal{P}$ such that $\exists M_i, M_j \in \mathcal{B}$ with $\langle P_i, p_i\rangle \in \mathsf{Transfer}(M_i), \langle P_j, p_j\rangle \in \mathsf{Transfer}(M_j)$, $p \in P_i$ and $p = p_j$.
(iii) $\forall M_i \in \mathcal{B}, \forall\langle P_i, p_i\rangle \in \mathsf{Transfer}(M_i), \forall p \in P_i : \neg\exists M_j \in \mathcal{B}$ with $M_j = \langle\langle \mathcal{I}, \mathcal{O}\rangle, B\rangle$ and $\mathcal{I}(p) > 0$.

As an illustration of Multi Isolated Transfer Net, consider Fig. 6. Dashed arrows represent transfer arcs and plain arrows represent classical Petri Nets arcs.

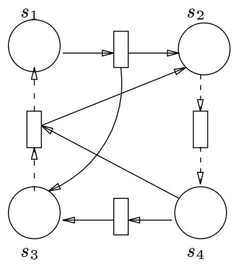

Fig. 6. An example of Multi Isolated Transfer Net.

We now define a weak notion of coverability set that is parameterized by a upward closed set U.

Definition 10 (Weak Coverability Set). Given a MTN $\mathcal{M}$, an initial marking $\mathbf{m}_0$ and an upward closed set U defined by an unique minimal marking $\mathbf{m}_U$ [3], a weak coverability set of $\mathcal{M}$ according to $\mathbf{m}_0$ and $\mathbf{m}_U$ (noted $\mathtt{WCS}(\mathcal{M}, \mathbf{m}_0, \mathbf{m}_U)$) is a set of ω-markings such that the two conditions holds

- $\forall \mathbf{m} \in \mathtt{Reach}(\mathcal{M}, \mathbf{m}_0)$, there exists $\mathbf{m}' \in \mathtt{WCS}(\mathcal{M}, \mathbf{m}_0, \mathbf{m}_U)$ with $\mathbf{m} \preceq \mathbf{m}'$.
- $\forall \mathbf{m} \in \mathtt{WCS}(\mathcal{M}, \mathbf{m}_0, \mathbf{m}_U) \setminus \mathtt{Reach}(\mathcal{M}, \mathbf{m}_0)$, there exists an infinite sequence $\mathbf{m}_1 \prec \mathbf{m}_2 \prec \ldots \prec \mathbf{m}_n \prec \ldots$ with $\mathbf{m}_i \in \mathtt{Reach}(\mathcal{M}, \mathbf{m}_0)$ such that
  - if $\mathbf{m}(p) = \omega$ then either $\mathbf{m}_i(p) > \mathbf{m}_{i-1}(p)$, for all $i > 0$, or $\mathbf{m}_i(p) \geq \mathbf{m}_U(p)$, for all $i \geq 0$.
  - if $\mathbf{m}(p) \neq \omega$, we have $\mathbf{m}(p) = \mathbf{m}_i(p)$, for all $i \geq 0$.

If we assume that the empty set of markings is defined by the minimal ω-marking $\mathbf{m}_\emptyset = \langle \omega, \ldots, \omega \rangle$, we have that a weak coverability set of $\mathcal{M}$ according to the initial marking $\mathbf{m}_0$ and $\mathbf{m}_\emptyset$ is a coverability set of $\mathcal{M}$ for $\mathbf{m}_0$.

Proposition 1. *Let $\mathcal{M}$ be a MTN, an initial marking $\mathbf{m}_0$ and an upward-closed set U with the minimal marking $\mathbf{m}_U$, the following holds :*

$$\mathtt{Reach}(\mathcal{M}, \mathbf{m}_0) \subseteq \downarrow \mathtt{CS}(\mathcal{M}, \mathbf{m}_0) \subseteq \downarrow \mathtt{WCS}(\mathcal{M}, \mathbf{m}_0, \mathbf{m}_U)$$
$$\downarrow \mathtt{WCS}(\mathcal{M}, \mathbf{m}_0, \mathbf{m}_U) \cap U \neq \emptyset \textbf{ iff } \mathtt{Reach}(\mathcal{M}, \mathbf{m}_0) \cap U \neq \emptyset$$

The weak coverability set over-approximates the set of reachable states and all the coverability sets. Nevertheless, it is still sufficiently precise to verify safety properties expressed by an upward closed set U. On the contrary, it can not always be used to decide place boundedness.

6.2 Algorithm to Compute a WCS

In this section, we give an algorithm that compute a weak coverability set for an Multi Isolated Transfer Net $\mathcal{M}$, an initial marking $\mathbf{m}_0$ and an upward closed set U, defined by $\mathbf{m}_U$. The algorithm is as follows. At each step of the construction of the tree, an untreated node annoted with the ω-marking $\mathbf{m}$ is developped by computing its successors (at the beginning, untreated nodes corresponds to initial markings). For each successors $\mathbf{m}'$ of $\mathbf{m}$ by firing M, we have four cases as in the algorithm of section 4 for computing the minimal coverability set of a Petri Net. Cases (1), (3) and (4) are identical, we only detail case (2) that define the following acceleration (that we call `AccIsolated`) adapted to our subclass of Transfer Nets.

(2) there is at least one path in the tree from a marking $\mathbf{m}''$ to $\mathbf{m}$ such that $\mathbf{m}'' \preceq \mathbf{m}'$ ($\mathbf{m}'$ is obtained from $\mathbf{m}$ by firing M), then we take the largest such path corresponding to a sequence σ of transitions, remove the subtree rooted at $\mathbf{m}''$ and replace $\mathbf{m}''$ by $\mathbf{n}$ defined as follows :
- for any place p which is not source or target of a transfer in the sequence of transitions $\sigma \cdot M$, we have :

$$\begin{cases} \mathbf{n}(p) = \mathbf{m}'(p) \textbf{ if } \mathbf{m}'(p) = \mathbf{m}''(p), \\ \mathbf{n}(p) = \omega \qquad \textbf{if } \mathbf{m}'(p) > \mathbf{m}''(p). \end{cases}$$

- for the other places, we define $\mathbf{n}$ as follows. By definition, for each place p that is source or target of a transfer in $\sigma \cdot M$, we know that there exists $\langle P_i, p_i \rangle \in \mathsf{Transfer}(\sigma \cdot M)$ with either $p \in P_i$ or $p = p_i$. Then we take each $\langle P, t \rangle \in \mathsf{Transfer}(\sigma \cdot M)$ and define $\mathbf{n}$ as, for t:

$$\begin{cases} \mathbf{n}(t) = \mathbf{m}'(t) \textbf{ if either } \mathbf{m}'(t) > \mathbf{m}''(t) \textbf{ or } \mathbf{m}'(t) = \mathbf{m}''(t) \wedge \exists s \in P : \mathbf{m}'(s) > \mathbf{m}''(s) \\ \mathbf{n}(t) = \mathbf{m}'(t) \textbf{ otherwise}. \end{cases}$$

  for any source $s \in P$:

$$\begin{cases} \mathbf{n}(s) = \omega \qquad \textbf{if } \mathbf{m}'(s) > \mathbf{m}''(s) \textbf{ and } \mathbf{m}'(s) \geq \mathbf{m}_U(s) \\ \mathbf{n}(s) = \mathbf{m}'(s) \textbf{ otherwise}. \end{cases}$$

The following theorem state the correction of the algorithm defined with the acceleration above :

[3] This restriction is to simplify the presentation, the extension to a finite set of minimal markings (and so to any upward closed set) is not difficult.

Theorem 1. *If $\mathcal{M}$ is a Multi Isolated Transfer Net, $\mathbf{m}_0$ an initial marking, $\mathbf{m}_U$ a marking defining a non-empty upward closed set U, then the algorithm defined in section 4 with the acceleration* `AccIsolated`*, computes a weak coverability set.*

Due to space limitation, we omit the formal proof of this statement and give the main idea that underlies the proof of correctness. The only particularity of the acceleration defined above is for places involved in a transfer. Let us take the case of a target place t. In the definition of the accelaration, we put ω in two cases (1) when t has strictly increased, (2) if one of its sources has strictly increased. Case (1) is classically justified by monotonicity of the model, i.e. repeating $\sigma \cdot M$ will add at least the same number of tokens in t each time (remember that t is not the source of any tranfer). Case (2) is justified as follows: t is not the source of any transfer, so the sequence $\sigma \cdot M$ takes out of t a constant number of tokens each time $\sigma \cdot M$ is fired. Furthermore, as a source of t striclty increased between $\mathbf{m}''$ and $\mathbf{m}'$ and this place is not the source of another transfer, when repeating $\sigma \cdot M$, from $\mathbf{m}'$, we know that the target will increase strictly and then we simply apply the justification of case (1). For the sources, the justification is as follows. When a source increases strictly between $\mathbf{m}''$ and $\mathbf{m}'$, we are not sure that it will increase beyond any bound, but we know that it will not decrease when repeating $\sigma \cdot M$. So if its value is greater or equal to $\mathbf{m}_U(s)$, we know that it will stay so. By putting ω in s, we do not take a too rough approximation for the following two reasons. First, putting ω in s is safe w.r.t. the intersection with U, that is $\mathbf{n}$ has an intersection with U iff $\mathbf{m}'$ and all the marking reachable from $\mathbf{m}'$ by iterating $\sigma \cdot M$ has an intersection with U. Second, it is easy to see that if we put ω in a source s, then there is also a ω in the target t, and this ω will never leave the place t as we know that t is not the source of a transfer. As a consequence, the ω in s is guaranteed not to “propage” unsafely in the net.

We are planning to extend our symbolic implementation of the algorithm of section 4 and test it in the near future.

6.3 An Examplative Run of the Algorithm

Fig. 7 shows another simple example of Multi Isolated Transfer Net. Fig. 8 shows the computations of algorithms presented in previous the subsection and section 4. Fig. 8(a) presents the computation of the algorithm of section 4 after two iterations. At this step, it detects that the generated successor for the transition `b` is greater than the parametrized initial marking. Thus, case (3) is applied : ω is added to the initial ω-marking in s_3, the initial marking is then replaced and the procedure continue from this root. Finally, the algorithm ends after computing the tree shown in Fig. 8. According to the upward closed set $s_2 \geq 1 \wedge s_3 \geq 1$, the algorithm of previous subsection computes the same tree. Thus, the computed weak coverability set coincides with the over-approximation computed by the algorithm of [Fin93] and the two algorithms allow us to conclude that there is a mutual exclusion between place s_2 and s_3. But if the safety property to be verified is $s_3 \geq 2$, only the algorithm computing the weak coverability set returns the right answer. At the second step, the algorithm still detects that the successor for transition `b` is greater than the initial marking but doesn't put ω for the place s_3 because the number of tokens is not enough to intersect with the upward closed set $s_3 \geq 2$. Fig. 8(c) shows the final tree computed by this algorithm.

7 The General Case: Combinaison of Forward and Backward Search

As we already recalled the forward approach can not be extended to the full class of MTNs [DFS98]. But as shown in Fig. 5, backward approach seems to be *explosion prone* and seems to be usefull only when some information can be used to *guide* the search. Structural invariants have been used to prune the backward search space in [DRV01,DRV02b]. But as shown in the table of Fig. 5, the invariants not always give interesting information and not always speed up the search.

In the case of extensions of Petri nets, approximation of the set of reachable markings computed with a forward approach can be used to guide the exact backward search when structural invariants do not work well. For example, algorithm of section 4 applied on MTNs can give interesting information that can afterwards be used by the exact backward search. Fig. 9 shows results on some MTNs corresponding to abstraction of Java programs analysed with a symbolic backward algorithm (see [DRV02b]). Without information to

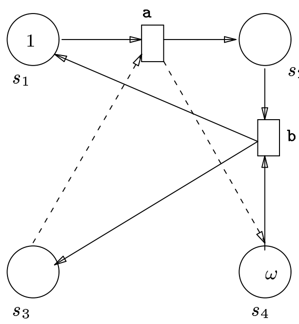

Fig. 7. An example of Isolated Multi-transfer Net.

$$\begin{array}{c} \langle 1,0,0,\omega\rangle \\ \downarrow \mathtt{a} \\ \langle 0,1,0,\omega\rangle \\ \downarrow \mathtt{b} \\ \langle 1,0,1,\omega\rangle \\ \text{a)} \end{array} \preceq \qquad \begin{array}{c} \langle 1,0,\omega,\omega\rangle \\ \downarrow \mathtt{a} \\ \langle 0,1,0,\omega\rangle \\ \downarrow \mathtt{b} \\ \langle 1,0,1,\omega\rangle \\ \text{b)} \end{array} \succeq \qquad \begin{array}{c} \langle 1,0,1,\omega\rangle \\ \downarrow \mathtt{a} \\ \langle 0,1,0,\omega\rangle \\ \downarrow \mathtt{b} \\ \langle 1,0,1,\omega\rangle \\ \text{c)} \end{array} =$$

Fig. 8. construction of coverability tree

prune the search space, backward algorithm can take a lot of time, more than half an hour for the P/C$_{bug}$ example and one hour for the corrected version of the model. Forward approach using the symbolic algorithm of section 4 ends in the worst case in 88s. In those particular examples, the over-approximation computed by the structural invariants is very effective, see the column giving the time of the computation of Pre^* in the table of Fig. 9. But as we will see later in this section, the information contained in the forward over-approximation can be more interesting then the information given by the invariants and so more effective to prune the backward search.

Case Study	**P**	**T**	$\mathbf{I}_{\mathsf{Post}}$	$\mathbf{NN}_{\mathsf{Post}}$	$\mathbf{NE}_{\mathsf{Post}}$	$\mathbf{time}_{\mathsf{Post}}$	$\mathbf{I}_{\mathsf{Pre}}$	$\mathbf{NN}_{\mathsf{Pre}}$	$\mathbf{NE}_{\mathsf{Pre}}$	$\mathbf{time}^1_{\mathsf{Pre}}$	$\mathbf{time}^2_{\mathsf{Pre}}$
I/D	32	28	74	218	40	3.38s	10	1542	1823	9.77s	0.82s
P/C$_{bug}$	44	37	137	688	93	88.17s	14	20421	15543	37min55s	2.86s
P/C$_{correct}$	44	37	27	306	25	1.1s	29	12479	8396	55min50s	0.02s

Fig. 9. Verification of MTNs.

In the P/C$_{correct}$ example, algorithm finds no bug but in the two other cases, bugs exists but no conclusion can directly be drawn from the forward search of the algorithm of section 4 because of the over-approximation. In these cases, backward algorithm can be applied to find an *error trace* by using the information previously computed by the forward over-approximation. These three examples fall in the Multi Isolated Transfer Net[4] subclass but more complex models, for which it can not exist an exact forward search computing a minimal coverability set, can be analysed with this methodology. For example, to analyse the parametrized MTN of the MESI protocol (see [Han93]) in the backward approach, the only structural invariant `invalid` + `modified` + `exclusive` + `shared` of the net that says that the number of tokens in those four places stays contant is not usefull for the backward search. This is because the initial marking contains ω in the place `invalid`, see [DRV01] for details. But by applying the symbolic algorithm presented in section 4, we obtain that the reachable markings are covered by the ω-markings $\{\langle\omega,\omega,\omega,0\rangle, \langle\omega,0,\omega,\omega\rangle\}$ (where markings are encoded as

[4] We could have applied the algorithm of section 6 to compute a `WCS` that is sufficient to verify the safety properties but our remarks in this section are more general and apply to the full class of MTNs.

⟨invalid, shared, modified, exclusive⟩). This over-approximation allows us to verify the mutual exclusion property between shared and exclusive. Other properties cannot be verified with this over-approximation. As an axample, consider the mutual exclusion property that asks if at most one token can be in exclusive. Backward approach can be applied efficiently to answer this question by eleminating during the backward search all the upward closed sets that do not intersect with the over-approximation computed forwardly. This over-approximation contains more information than the structural invariants and so is potentially more effective than the structural invariant heuristic. So instead of opposing the forward and backward approaches, we propose to use them together to obtain an efficient way to analyze general Transfer Nets.

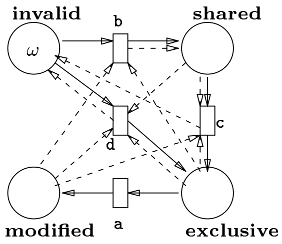

Fig. 10. MTN of the MESI protocol.

References

[ABB01] P. Abdulla, L. Boasson, A. Bouajjani. Effective Lossy Queue Languages. In Proc. 28th Intern. Coll. on Automata, Languages and Programming (ICALP'01), LNCS, Crete (Greece), July 2001.

[ACJT96] P. A. Abdulla, K. Cerāns, B. Jonsson and Y.-K. Tsay. General Decidability Theorems for Infinite-State Systems. In *Proc. LICS'96*, pages 313–321, 1996.

[BBLS00] K. Baukus, S. Bensalem, Y. Lakhnech, and K. Stahl. Abstracting WS1S Systems to Verify Prameterized Networks. In the proceedings of the 6th International Conference, TACAS 2000, LNCS 1785, Springer-Verlag, 2000.

[BM99] A. Bouajjani and R. Mayr. Model Checking Lossy Vector Addition Systems. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of LNCS, pages 323–333. Springer, 1999.

[DFS98] C. Dufourd, A. Finkel, Ph. Schnoebelen. Reset Nets Between Decidability and Undecidability. In *Proc. ICALP'98*, LNCS 1443, pages 103-115,Springer, 1998.

[DR00] G. Delzanno, and J. F. Raskin. Symbolic Representation of Upward-closed Sets. In *Proc. TACAS 2000*, LNCS 1785, pages 426-440, 2000.

[DRV01] G. Delzanno, J.-F. Raskin, and L. Van Begin. Attacking Symbolic State Explosion. In *Proc. CAV'01*, LNCS 2102, pages 298-310, 2001.

[DRV02] G. Delzanno, J-F. Raskin and L. Van Begin. Covering Sharing Trees: Efficient Data Structures for the Automated Verification of Parameterized Systems. *submitted for publication to sttt*, 2002.

[DRV02b] G. Delzanno, J-F. Raskin and L. Van Begin. Towards the Automated Verification of Multithreaded Java Programs. *International Conference on Tools and Algorithms for Construction and Analysis of Systems*, Grenoble, Frane, 2002.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the Verification of Broadcast Protocols. In *Proc. LICS'99*, pages 352–359, 1999.

[Fin90] A. Finkel. Reduction and covering of infinite reachability trees. Information and Computation, 89(2):144-179, December 1990.

[Fin93] A. Finkel. The minimal coverability graph for Petri nets. In In *Advances in Petri Nets '93*, LNCS 674, pages 210-243. Springer, 1993.

[FS01] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63-92, 2001.

[GS92] S. M. German, A. P. Sistla. Reasoning about Systems with Many Processes. *JACM* 39(3): 675–735 (1992)

[Han93] J. Handy. The Cache Memory Book. Academic Press, 1993.

[Joh90] D. S. Johnson. A Catalog of Complexity Classes. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A, Algorithm and Complexity*, Elsevier, 1990.

[KM69] R. M. Karp and R. E. Miller. Parallel Program Schemata. *Journal of Computer and System Sciences*, 3, pages 147-195, 1969.

[Sch02] Ph. Schnoebelen. Verifying Lossy Channel Systems Has Nonprimitive Recursive Complexity. LSV Technical Report, Ecole Normale Superieure de Cachan, France, 2002.

[STC98] M. Silva, E. Teruel, and J. M. Colom. Linear Algebraic and Linear Programming Techniques for Analysis of Place/Transition Net Systems. In W. Reisig and G. Rozenberg, editors. Lectures on Petri Nets I: Basic Models. *Advances in Petri Nets*, LNCS 1491, pages 308-309. Springer, 1998.

[ZL94] D. Zampuniéris, and B. Le Charlier. Efficient Handling of Large Sets of Tuples with Sharing Trees. In *Proc. DCC'95*, 1995.

Appendix

```
Function MinimalcoveringSet(init, PN) : set of markings
    R+ = {⟨−, init⟩}
    AlreadyComputed = {init}
    Succ = {init}
    while Succ ≠ ∅ do
        OldSucc = Succ
        //Succ = Post(OldSucc)
        Succ = {m′ | ∃m ∈ OldSucc, t ∈ T : m →_t m′}
        //SuccR+ = PostR+(R+, OldSucc)
        SuccR+ = {⟨m, m′⟩ | ∃⟨m, m″⟩ ∈ R+, t ∈ T : m″ ∈ OldSucc ∧ m″ →_t m′}
        //case 1
        MinimizeSucc(Succ, SuccR+, AlreadyComputed)
        R+ = R+ ∪ SuccR+
        //case 2
        Putω(Succ, SuccR+, AlreadyComputed, R+)
        //computation of subtree to remove avoiding the problem presented in Fig.3
        Roots = ComputeTreeToRemove(R+, AlreadyComputed, Succ)
        //case 3
        RemoveSubTree(R+, Roots, AlreadyComputed, Succ)
        AlreadyComputed = AlreadyComputed ∪ Succ
    endW
    return AlreadyComputed
```

Fig. 11. Algorthm computing symbolically the minimal coverability set.

```
Procedure MinimizeSucc(Succ, SuccR+, AlreadyComputed)
    Succ = Succ \ {m | ∃m′ ∈ AlreadyComputed : m ⪯ m′}
    NewSucc = Succ \ {m | ∃m′ ∈ Succ ∧ m ≺ m′}
    Succ = NewSucc
    SuccR+ = SuccR+ \ {⟨m, m′⟩ | m′ ∉ Succ}
```

Fig. 12. Removing of the tuples from the frontier that are not maximal.

Procedure $Put\omega(Succ, SuccR^+, AlreadyComputed, R^+)$
//computation of the predecessors wich permit to put new ω
$Rel_{\prec} = \{\langle \mathbf{m}, \mathbf{m}' \rangle | \mathbf{m} \in ReachedElem \wedge \mathbf{m}' \in Succ \wedge \mathbf{m} \prec \mathbf{m}'\}$
$Pred_{\prec} = Rel_{\prec} \cap R^+$
//Keep the farest predecessors
$Subsumed = \{\mathbf{m} | \exists \langle \mathbf{m}, \mathbf{m}' \rangle \in Pred_{\prec}\}$
$Roots = \{\mathbf{m} \in Subsumed | \nexists \langle \mathbf{m}', \mathbf{m} \rangle \in R^+ : \mathbf{m}' \in Subsumed\}$
$NewPred_{\prec} = Pred_{\prec} \setminus \{\langle \mathbf{m}, \mathbf{m}' \rangle | \mathbf{m} \notin Roots\}$
$Pred_{\prec} = NewPred_{\prec}$
//put ω on the right places
$M_\omega = \{\langle \mathbf{m}, \mathbf{m}' \rangle | \exists \langle \mathbf{m}, \mathbf{m}'' \rangle \in Pred_{\prec} : \mathbf{m}'(p) = \mathbf{m}''(p) \; if \; \mathbf{m}(p) = \mathbf{m}''(p), \mathbf{m}'(p) = \omega \; otherwise\}$
//keep only the maximal elements
$NewM_\omega = M_\omega \setminus \{\mathbf{m} | \exists \mathbf{m}' \in M_\omega : \mathbf{m} \prec \mathbf{m}'\}$
$M_\omega = NewM_\omega$
//Compute the elements to add in R^+
$NewSubTree = \{\langle \mathbf{m}, \mathbf{m}' \rangle | \exists \mathbf{m}'' : \langle \mathbf{m}, \mathbf{m}'' \rangle \in R^+ \wedge \langle \mathbf{m}'', \mathbf{m}' \rangle \in M_\omega\}$
//Removing of the elements of R^+ corresponding to the prunig
//of subtrees rooted by non − maximal predecessors
$RemoveSubTree(R^+, Roots, AlreadyComputed, Succ)$
//add new elements with ω in R^+ and Succ
$R^+ = R^+ \cup NewSubTree$
$Succ = Succ \cup \{\mathbf{m} | \langle \mathbf{m}', \mathbf{m} \rangle \in M_\omega\}$

Fig. 13. Putting the ω on the markings of the frontier.

Function $ComputeTreeToRemove(R^+, AlreadyComputed, Succ) \; : \; set \; of \; markings$
//computation of the tuples $\langle \mathbf{m}, \mathbf{m}' \rangle$ where $\mathbf{m}, \mathbf{m}'$ are non − maximal markings
//and a successor of $\mathbf{m}'$ is greater than $\mathbf{m}$
$Rel_{\prec} = \{\langle \mathbf{m}, \mathbf{m}' \rangle | \mathbf{m} \in alreadyComputed \wedge \mathbf{m}' \in Succ \wedge \mathbf{m} \prec \mathbf{m}'\}$
$Rel_S = \{\langle \mathbf{m}, \mathbf{m}' \rangle | \exists \langle \mathbf{m}, \mathbf{s} \rangle, \langle \mathbf{m}', \mathbf{s}' \rangle \in Rel_{\prec} : \langle \mathbf{m}', \mathbf{s} \rangle \in R^+\}$
$Rel_S^* = transitive \; closure \; of \; Rel_S$
//storing of the non − maximal markings wich are not in the case described
//by Fig.3
$RemovableTree = \{\mathbf{m} | \nexists \langle \mathbf{m}, \mathbf{m} \rangle \in Rel_S^*\}$
$ProblematicNodes = \{\mathbf{m} | \exists \langle \mathbf{m}, \mathbf{m} \rangle \in Rel_S^*\}$
//we define the relation $<$ on markings as follow :
//$\mathbf{m} < \mathbf{m}'$ iff $\mathbf{m}(p) > \mathbf{m}'(p)$ implies $\exists p' < p : \mathbf{m}(p') < \mathbf{m}'(p')$
//in the case described in Fig.3, we keep only the subtree rooted by
//the greatest marking according to the total order $<$
$Rel_< = \{\langle \mathbf{m}, \mathbf{m}' \rangle | \langle \mathbf{m}, \mathbf{m}' \rangle \in Rel_S^* \wedge \mathbf{m} < \mathbf{m}'\}$
$RemovableTree = RemovableTree \cup \{\mathbf{m} \in ProblematicNodes | \nexists \langle \mathbf{m}, \mathbf{m}' \rangle \in Rel_<\}$
return $RemovableTree$

Fig. 14. Computation of the irrelevant subtree avoiding problem presented in Fig. 3

Procedure $RemoveSubTree(R^+, Roots, Alreadycomputed, Succ)$
//computation of the markings of the subtree to remove
$Remove = Roots \cup \{\mathbf{m} | \exists \langle \mathbf{m}', \mathbf{m} \rangle \in R^+ \; with \; \mathbf{m}' \in Roots\}$
//removing of these markings
$AlreadyComputed = AlreadyComputed \setminus Remove$
$Succ = Succ \setminus Remove$
$R^+ = R^+ \setminus \{\langle \mathbf{m}, \mathbf{m}' \rangle | \mathbf{m} \in Remove\}$
$R^+ = R^+ \setminus \{\langle \mathbf{m}, \mathbf{m}' \rangle | \mathbf{m}' \in Remove\}$

Fig. 15. Removing of subtrees.