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] LSV & ENS de Cachan

61 av. du Prsident Wilson, 94235 Cachan cedex,France

finkel@lsv.ens-cachan.fr

[2] Université Libre de Bruxelles

Blvd du Triomphe, 1050 Bruxelles, Belgium

{jraskin,lvbegin}@ulb.ac.be

Abstract

In this paper, we revisit the forward and backward approaches to the verification of extensions of infinite state Petri Nets. As contributions, we propose an efficient data structure to represent infinite downward closed sets of markings and 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 useful for the verification of finite state abstractions of various concurrent and distributed computer systems. Unfortunately, useful 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 successful techniques for model-checking of finite state systems to infinite state systems and parametric verification. A lot of interesting theoretical results have been obtained, see for example [3, 6, 13, 16]. 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 family of systems. For example, some mutual exclusion protocols have been designed to work for any number of processes that want to share common resources and the verification of such protocols for a specific number of process is not relevant. In this context, several abstraction have proven to be useful, see for example [1, 4, 17]. The work in this paper is directly connected to the context of the so-called *counting abstraction*. When considering the counting abstraction, the model of (infinite) *Petri Nets* and its

*This author was partially supported by a "Crédit aux chercheurs", Belgian National Fund for Scientific Research.

†This author was partially supported by a FRFC grant 2.4530.02.

‡This author was supported by a Walloon Region grant "First Europe".

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 Transfer Nets (note that Reset Nets can be viewed as a subclass of Transfer Nets).

There are two main different approaches for the verification of Petri Nets. The first one is the *forward approach* (that was first defined by Karp and Miller in [20]). This approach starts from the (possibly parametric) set of initial markings and computes an approximation of the closure of the transition relation (often referred 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 we can verify is a subclass of 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 applying iteratively from the set of Bad markings the Pre relation (which is the inverse of the Post relation). If the closure of the Pre relation intersects with the set of initial markings then we know that some Bad markings are reachable. The application of the Pre relation is guaranteed to terminate if the set of Bad markings is upward closed [3, 16]. Unfortunately, in the two cases, a *naive* implementation of the abstract algorithm is not practical. It is not surprising as we know that the *theoretical complexity* of the reachability problem of upward closed sets (also called *coverability problem*) for that class of infinite state models is *very high*, see [22].

So, further research was necessary to obtain more practically useful verification techniques. For the forward approach, we have defined in [15] an heuristic to *minimize* the set of markings to consider when computing the Karp-Miller covering tree. For the backward approach, we have defined in [9] 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 and all the operations on sets involved in the algorithm are directly computed on the underlying compact structure that represents the sets. Practical evaluation of the algorithm has shown that it is much more effective than the naive *enumerative* approach, see [10] for details.

In this paper, as a first contribution, we show how to turn into a symbolic algorithm the enumerative algorithm that we have defined in [15] to compute the minimal coverability set of an infinite state Petri net. For this we use a variant of the data structure that we have defined in [9]. As we have now two symbolic algorithms (one for the forward search, one for the backward search) , 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 its *robustness* in the following sense: it is not only applicable to the basic class of infinite Petri Nets but also to all the extensions that preserve *monotonicity* (see [3]). Transfer Nets (and so broadcast protocols that they generalize) and Reset Nets maintain monotonicity for example. Unfortunately, the forward approach does *not* generalize for those models (see [8, 13]) i.e. in those cases the search is not guaranteed to terminate. Indeed, there are negative results [8] 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. This subclass is of practical interest as it covers all the examples of abstractions of multi-threated JAVA programs that we have analyzed backwardly in [12]. 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 those generated with the backward search.

In [10], 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 toward initial markings. In our previous works, the over-approximation is obtained automatically by computing the structural invariants of the net ([24]). As a third contribution, we propose here to use the symbolic implementation of the minimal coverability set for Petri nets to compute another 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 attack those verification problems that have very high theoretical complexities [23].

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 some notions about upward closed sets, downward closed sets and covering 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 backward algorithm 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 [1].

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* part 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*.

[1]By monotonicity property, we mean that if a transition t can be fired in a marking $\mathbf{m}$, it can also be fired in any markings $\mathbf{m}'$ greater than $\mathbf{m}$.

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 $(P_i \cup \{p_i\}) \cap (P_j \cup \{p_j\}) = \emptyset$.

$\square$

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 place. 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 say that $\mathbf{m}$ is covered by $\mathbf{m}'$ when $\mathbf{m} \preceq \mathbf{m}'$. We denote $\sum_{p \in P} \mathbf{m}(p)$ by $\mathbf{m}(P)$.

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

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 a 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 a source nor 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: $\mathbf{m}' = \mathbf{m}_2 + \mathcal{O}$.

$\square$

Fig. 1 shows an example of MTN modeling the MESI protocol [18]. Dashed arrows represent transfer arcs and plain arrows represent classical Petri Nets arcs. When transition d is fired, one token is removed from the place invalid, then the tokens of the places shared, modified and exclusive are transferred to the place invalid and finally one token is put in exclusive. We now define the semantics of MTNs.

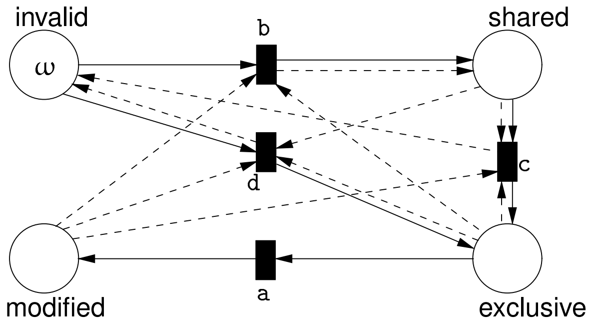

Figure 1: MTN of the MESI protocol.

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 a 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 number of tokens in p does not exceed c ?", i.e. $\exists c \in \mathbb{N} : \forall \mathbf{m} \in \mathsf{Reach}(\mathcal{M}, S_0) : \mathbf{m}(p) \leq c$?. [2] This is called the *place boundedness 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 (possibly infinite) set of markings $\{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$ is the marking $\mathbf{m}$ such that:

$$\begin{cases} \mathbf{m}(p) = \omega & \textbf{if } \forall c \in \mathbb{N} : \exists i \geq 1 : \mathbf{m}_i(p) > c \\ \mathbf{m}(p) = \mathit{max}(\{\mathbf{m}_1(p), \mathbf{m}_2(p), \ldots\}) & \textbf{otherwise}. \end{cases}$$

□

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

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

$$\forall \mathbf{m} : (\mathbf{m} \in S \rightarrow \forall \mathbf{m}' \preceq \mathbf{m} : \mathbf{m}' \in S).$$

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

$$\forall \mathbf{m} : (\mathbf{m} \in S \rightarrow \forall \mathbf{m}' \succeq \mathbf{m} : \mathbf{m}' \in S).$$

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

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

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

$$\mathsf{Lim}(S) = \{\mathbf{m} \in S^{\mathrm{lim}} | \neg \exists \mathbf{m}' \in S^{\mathrm{lim}} : \mathbf{m} \prec \mathbf{m}'\} \cup \{\mathbf{m} \in S | \neg \exists \mathbf{m}' \in S : \mathbf{m} \prec \mathbf{m}'\}$$
$$\text{where } S^{\mathrm{lim}} = \{\mathbf{m} | \mathbf{m} = \mathrm{lub}(\{\ \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n, \ldots\}) \text{ with } \forall i \geq 1 : \mathbf{m}_i \in S \wedge \mathbf{m}_i \prec \mathbf{m}_{i+1}\}.$$

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

Definition 6 (Downward Closure) Let S be a set of ω-markings, the *downward closure* of S, noted $\downarrow S$, is the set of markings $\{\mathbf{m} | \exists \mathbf{m}' \in S \text{ and } \mathbf{m} \preceq \mathbf{m}'\}$ □

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 identified 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 7 (Coverability Set [15, 14]) 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 $\mathbf{m} \in \mathtt{CS}(\mathcal{M}, S_0) \backslash \mathtt{Reach}(\mathcal{M}, S_0)$, there exists an infinite sequence $\mathbf{m}_1 \prec \mathbf{m}_2 \prec \mathbf{m}_3 \ldots$ with for all $i \geq 1 : \mathbf{m}_i \in \mathtt{Reach}(\mathcal{M}, S_0)$ and such that $\mathbf{m} = \mathrm{lub}(\{\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \ldots\})$.

(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 8 (Minimal Coverability Set [15, 14]) The *minimal coverability set* of 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 [15] that the minimal coverability set is unique and can be computed effectively for any Petri Net $\mathcal{M}$ and any finite set of initial ω-markings S_0.
Let us now recall some properties of coverability sets, see [15] for details.

1. any $\mathtt{CS}(\mathcal{M}, S_0)$ 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 $\texttt{CS}(\mathcal{M}, S_0)$ is sufficient to answer any safety verification problem if the set of Bad markings U is upward closed. In fact, we have:

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

3. any $\texttt{CS}(\mathcal{M}, S_0)$ 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 $\mathbf{m} \in \texttt{CS}(\mathcal{M}, S_0)$ such that $\mathbf{m}(p) = \omega$.

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. We start from Sharing Trees (STs) that are data structures introduced in [25] 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. $val : N \to \mathbb{N} \cup \{\top, \bot\}$ is a labeling function for the nodes, and $succ : N \to 2^N$ defines the successors of a node. Furthermore, (1) $val(n) = \top$ iff $n = root$, $succ(root)$=N_1, $val(n) = \bot$ 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 $\langle \top, n_1, \ldots, n_k, \bot \rangle$ such that $n_{i+1} \in succ(n_i)$ for $i = 1, \ldots, k$-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 [25], 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 [11], 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 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 $val : N \to \mathbb{N} \cup \{\top, \bot\} \cup \{\omega\}$. A downward closed covering ST $\mathcal{S}$ has the following semantics :

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

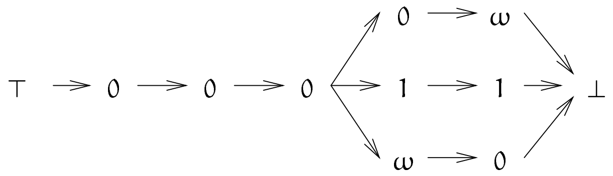

Figure 2: Example of a dcCST that represents an infinite downward closed set.

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 covering ST (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. An example of dcCST is given in Fig. 2. It represents the infinite downward closed set $\{\mathbf{m} \in \mathbb{N}^5 | \mathbf{m} \preceq \langle 0,0,0,0,\omega\rangle \vee \mathbf{m} \preceq \langle 0,0,0,1,1\rangle \vee \mathbf{m} \preceq \langle 0,0,0,\omega,0\rangle\}$. Note that the dcCST encodes efficiently this infinite downward closed set as its limits elements share large prefixes. It is easy to show with straightforward adaptations of proofs from [11] (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 algorithm (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 theoretical results seems not to be a practical 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 of a Petri Net, we generally construct what we call a *coverability tree* (see [20]). This is mainly a tree where the nodes are labeled by the elements of a coverability set and the edges are an approximation of the successor relation between the ω-markings labeling the nodes. Unfortunately, the procedure presented in [20] computes unmanageable trees, even for small Petri Nets. An efficient heuristic is presented in [15]. This algorithm construct the minimal coverability tree, which is a tree where the values of the nodes correspond to the limit elements of the minimal coverability set of the Petri Net. The main idea of the algorithms of [20, 15] is to use an acceleration function f^a when a marking $\mathbf{m}$ is accessible from a markings $\mathbf{m}'$ in the tree with $\mathbf{m}' \prec \mathbf{m}$. The definition of f^a is as follows :
$f^a(\mathbf{m}', \mathbf{m}) = \mathbf{m}''$ such that

$$\begin{cases} \mathbf{m}''(p) = \mathbf{m}'(p) & \textbf{if } \mathbf{m}(p) = \mathbf{m}'(p) \\ \mathbf{m}''(p) = \omega & \textbf{if } \mathbf{m}(p) > \mathbf{m}'(p) \end{cases}$$

More precisely, the algorithm of [15] works as follows. At each step of the construction of the tree, an untreated node annotated with the ω-marking $\mathbf{m}$ is developed by computing its successors (at the beginning, untreated nodes correspond to initial markings). For each successors $\mathbf{m}'$ of $\mathbf{m}$, we have three 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 ω-marking $\mathbf{m}''$ to $\mathbf{m}$ such that $\mathbf{m}'' \prec \mathbf{m}'$, then we take the largest such path and constructs $\mathbf{n} = f^{a}(\mathbf{m}'', \mathbf{m}')$. Finally, we take the farest predecessor $\mathbf{m}'''$ of $\mathbf{m}$ with $\mathbf{m}''' \prec \mathbf{n}$ (possibly different from $\mathbf{m}''$) and replace the subtree rooted by $\mathbf{m}'''$ by $\mathbf{n}$ (see Fig. 3(a)). We finally remove all the subtrees rooted by $\mathbf{m}''$ with $\mathbf{m}'' \prec \mathbf{n}$ (see Fig. 3(b)).

(3) In the other cases, a new node connected to the node of $\mathbf{m}$ and annoted with $\mathbf{m}'$ is added to the tree. As in the previous case, all the subtrees rooted by a marking $\mathbf{m}''$ such that $\mathbf{m}'' \prec \mathbf{m}'$ are removed.

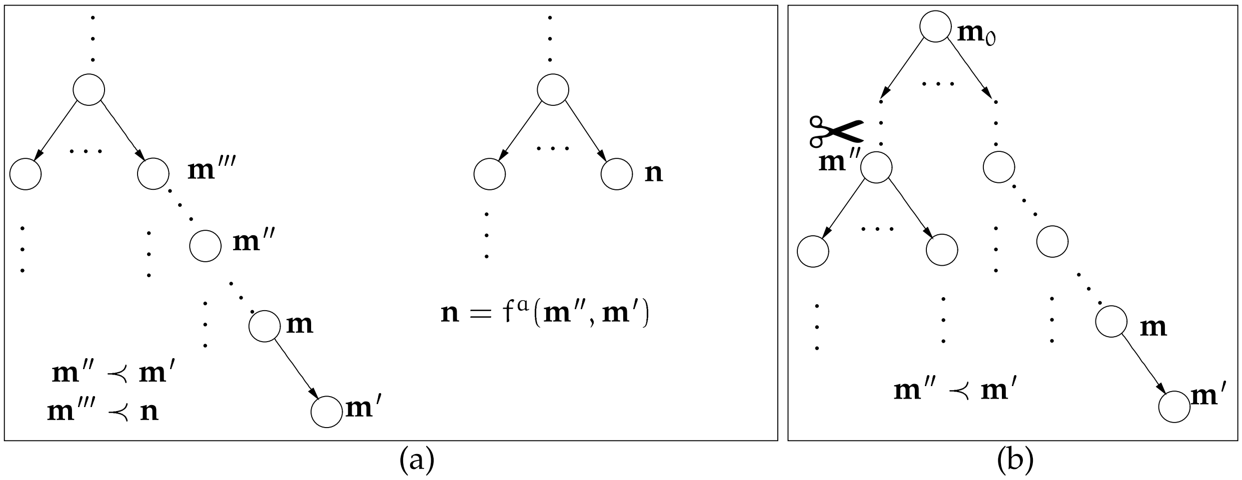

Figure 3: operations on trees

An improvement of this algorithm is to symbolically treat the sets of ω-markings instead of enumerating them. Symbolically means that the computations 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 algorithm of [15]. Fig. 4 shows such an high-level algorithm manipulating sets. In [15], the minimal coverability tree is constructed to manage case (2) and (3). But note that, for those 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 do not maintain the successor relation but only the closure of that relation. It is easy to see that this is sufficient for the case (2) and case (3) of the previous algorithm when we want to answer reachability of an upward closed set of markings or place boundness[3].

All the operations of the enumerative algorithm are replaced by operations on downward closed sets. The symbolic algorithm computing the minimal coverability set of a Petri net is given in figure 4 and works as follows. The algorithm maintains a set F of ω-markings that are untreated (this is the frontier of the search), a set S of ω-markings that contains the nodes already treated, and R^+ is a set of pairs of ω-markings $\langle \mathbf{m}, \mathbf{m}' \rangle$ such that $\mathbf{m}, \mathbf{m}' \in S \cup F$ and $\mathbf{m} \rightarrowtail^* \mathbf{m}'$. All the

[3]Note that the construction of the coverability graph allow us to answer to the *regularity problem* (is the language of the net regular). As the elements of a coverability set are the nodes of a coverability graph, such a graph is constructed by adding edges between the elements of the set by simply applying the Post operation on each of them.

1: **function** MinimalCoverabilitySet ($\langle \mathcal{P}, \mathcal{B} \rangle ; S_0$) **return** S

2: $F \longleftarrow S_0$

3: $S \longleftarrow S_0$

4: $R^+ \longleftarrow \emptyset$

5: **while** $F \neq \emptyset$ **do**

6: $Succ \longleftarrow \{\mathbf{m}' | \exists \mathbf{m} \in F, M \in \mathcal{B} : \mathbf{m} \rightarrowtail_M \mathbf{m}'\}$

7: $Succ \longleftarrow max(Succ) \setminus \{\mathbf{m} | \exists \mathbf{m}' \in S : \mathbf{m} \preceq \mathbf{m}'\}$

8: $R^+ \longleftarrow R^+ \cup \{\langle \mathbf{m}, \mathbf{m}' \rangle | \mathbf{m} \in F \wedge \mathbf{m}' \in Succ \wedge \exists M \in \mathcal{B} : \mathbf{m} \rightarrowtail_M \mathbf{m}'\}$

9: $R^+ \longleftarrow R^+ \cup \{\langle \mathbf{m}, \mathbf{m}' \rangle | \mathbf{m}' \in Succ \wedge \exists \mathbf{m}'' \in F, M \in \mathcal{B} : \mathbf{m}'' \rightarrowtail_M \mathbf{m}' \wedge \langle \mathbf{m}, \mathbf{m}'' \rangle \in R^+\}$

10: $Succ', R^{+\prime} \longleftarrow$ Acc$(S, Succ, R^+)$

11: $S \longleftarrow S \cup Succ'$

12: $S \longleftarrow max(S)$

13: $F \longleftarrow max(Succ')$

14: $R^{+\prime} \longleftarrow R^+ \cap (S \times S)$

15: **endwhile**

Figure 4: Symbolic Algorithm that computes the minimal coverability set for a Petri net $\langle \mathcal{P}, \mathcal{B} \rangle$ and a set of initial ω-marking S_0.

sets of ω-markings or pairs of ω-markings are represented using dcCSTs and all the operations of the algorithm are symbolic in the sense that they all works directly on the structure of the dcCSTs representing the sets. All the algorithms on dcCSTs are easy adaptations of algorithms on CSTs that we have defined in [11].

At each iteration of the loop (line 5), the following operations are performed. In line (6), the set of the new reachable ω-markings ($Succ$) is computed and only the maximal elements of this set that are not covered by an ω-marking computed in previous iterations are kept (line 7). Lines 8 and 9 update R^+ for those markings. In line 10, we compute the accelerations (see description of the function Acc). We then add the successor ω-markings of the frontier that are obtained after acceleration to the set S of ω-markings computed so far (line 11). In line 12, we suppress from S all the ω-markings that are not maximal. As the new frontier we only consider the maximal elements computed during the current iteration (possibly accelerated) (line 13). After the minimization of S, the relation R^+ is updated.

The acceleration function is given in Fig. 5 and works as follows. It takes as arguments the set of new reachable ω-markings ($Succ$), the set of reachable ω-markings computed in the previous iterations (S) and the accessibility relations on all those ω-markings (R^+). First, the set of pairs that have to be accelerated is computed (line 2). The first component of those pairs are called source of the acceleration, the second one is called the target of the acceleration and the result of the acceleration is called the accelerated ω-marking. The arcs between the source of the acceleration and the accelerated ω-marking are computed (line 3), the arcs between a predecessors of a source of an acceleration and the corresponding accelerated ω-marking are computed (line 4) and finally, the arcs between the successors of a source of an acceleration that are predecessors of the target of the acceleration are linked to the accelerated ω-marking (line 5). R^+ is adjusted by adding all those arcs and the set of successors is adjusted by adding all the accelerated elements (lines 7-8).

1: **function** Acc (S;$Succ$;R^+) **return** $Succ'$,$R^{+\prime}$
2: $\quad G \longleftarrow \{\langle \mathbf{m}, \mathbf{m}' \rangle \in R^+ : \mathbf{m} \prec \mathbf{m}'\}$
3: $\quad H_1 \longleftarrow \{\langle \mathbf{m}, \mathbf{m}' \rangle | \exists \mathbf{m}'' : \langle \mathbf{m}, \mathbf{m}'' \rangle \in G \land \mathbf{m}' = f^a(\mathbf{m}', \mathbf{m}'')\}$
4: $\quad H_2 \longleftarrow \{\langle \mathbf{m}, \mathbf{m}' \rangle | \exists \langle \mathbf{m}''', \mathbf{m}'' \rangle \in G : \mathbf{m}' = f^a(\mathbf{m}''', \mathbf{m}'') \land \langle \mathbf{m}, \mathbf{m}''' \rangle \in R^+\}$
5: $\quad H_3 \longleftarrow \{\langle \mathbf{m}, \mathbf{m}' \rangle | \exists \langle \mathbf{m}''', \mathbf{m}'' \rangle \in G : \mathbf{m}' = f^a(\mathbf{m}''', \mathbf{m}'') \land \langle \mathbf{m}''', \mathbf{m} \rangle \in R^+ \land \langle \mathbf{m}, \mathbf{m}'' \rangle \in R^+\}$
6: $\quad H \longleftarrow H_1 \cup H_2 \cup H_3$
7: $\quad R^{+\prime} \longleftarrow R^+ \cup H$
8: $\quad Succ' \longleftarrow Succ \cup \{\mathbf{m} | \exists \mathbf{m}' : \langle \mathbf{m}', \mathbf{m} \rangle \in H\}$

Figure 5: Function that accelerates all the accelerable cycles.

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 can potentially have an exponential gain both in memory usage and execution time using dcCST to represent sets of ω-markings and the closure of the transition relation.

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 infinite states Petri Nets and their monotonic extensions.

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

Figure 6: Conceptual differences between forward and backward search.

Sets. Starting from an upward closed set, the application of Pre preserves the upward closure of the set. So, the backward search manipulates upward closed sets. As we have seen, a coverability set is the set of limit elements of a downward closed set of markings that over-approximates the set of reachable markings. So, forward approach manipulates 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 upward closed safety properties.

Techniques to guarantee termination. By applying the Pre operation, it is guaranteed to reach a fixpoint after a finite number of iterations. Forward approach needs an acceleration function 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 always 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* can be solved with the forward approach.

Dependence of the search. When computing the coverability tree, we start the construction of the tree from the set of initial markings and the tree does not depend on the property that we want verify. On the other hand, with the backward approach, the computation depends on the property to verify. Note that if $\mathsf{Pre}^*(\mathtt{U})$ does not intersect with the set of initial markings then no set computed during the fixpoint computation contains any reachable marking.

5.2 Practical Comparison

We have applied our symbolic forward algorithm on a set of parametrized Petri Nets [4]. The results of the experiments are shown in Fig. 7 and compared with the results obtained using our symbolic backward algorithm defined in [10]. We have run the backward algorithm with and without the invariant heuristic of [10]. The invariant heuristic computes *structural invariants* of the Petri net and uses them to prune the backward search: every minimal element defining an upward closed set that does not intersect with the set of solutions of the structural invariants is suppressed. This heuristic is safe as the set of solutions of the structural invariants over-approximates the reachable markings of the Petri net. Our set of examples is composed by some concurrent and production systems as the multipoll ([21]), the mesh2x2 ([2]) and its extension to 3x2 case, the flexible manufacturing system (FMS) of [7], the central server model (CSM) of [2] and the PNCSA protocol analysed in [5, 15].

As we can see from the figures, the forward search is always faster than the backward search. This is due to the fact that the behavior of the Petri net is much more regular when executed from its initial markings than when executed backwardly from a possible non reachable set of

[4] see the web page `http://www.ulb.ac.be/di/ssd/lvbegin/CST/index.html` for a detailed description of the examples.

markings. As we can see for the $PNCSA_1$ and $PNCSA_2$ examples where we verify two different properties, efficiency of the backward search can depend a lot on the property that we want to verify. The invariant heuristic is very useful to obtain reasonable execution times in the backward search, this confirms the observation that getting some information about (an over-approximation of) the reachable markings is important.

Case Study	P	T	NN_{Post}	NE_{Post}	I_{Post}	EX_{Post}	M_{Post}	NN_{Pre}	NE_{Pre}	I_{Pre}	EX^1_{Pre}	M_{Pre}	EX^2_{Pre}
$PNCSA_1$	31	36	100	80	56	1s	2336kb	176	89	17	16.42s	3720kb	2.5s
$PNCSA_2$	31	36	100	80	56	1s	2336kb	368	132	38	>4h	48620kb	43.79s
CSM	14	13	35	16	10	0.02s	1472kb	109	152	11	0.1s	2116kb	0.07s
FMS	22	20	36	24	16	0.08s	1712kb	881	1695	46	6.13s	5680kb	5.98s
mesh2x2	32	32	78	256	8	0.13s	1784kb	344	427	15	0.8s	2600kb	0.8s
mesh3x2	52	54	106	6400	10	0.48s	2176kb	987	2224	21	6.5s	4992kb	6.5s
multipool	18	21	104	220	11	0.14s	1676kb	196	5641	18	2.1s	2516kb	-

Figure 7: Benchmarks on an AMD Athlon 900Mhz 500Mbytes : P=No. of places; T = No. of transitions; NN_{Post} (NN_{Pre}) = nodes of the sharing tree describing the minimal coverability set (the backward fixpoint); NE_{Post} (NE_{Pre}) = cardinality the minimal coverability set (the backward fixpoint); I_{Post} (I_{Pre}) =No. of iterations of the forward (backward) algorithm to reach the fixpoint; EX_{Post}(EX^1_{Pre}) = Execution time to reach the forward (backward) fixpoint ; EX^2_{Pre} = Execution time to reach the backward fixpoint using structural invariants;

6 A Weaker Notion of the Minimal Coverability Set

As we have seen in the last section, there are conceptual advantages to use the backward approach and practical evidences that plead for the forward approach. Unfortunately, we know that it is not always possible to compute a coverability set for natural extensions of Petri Nets [8]. 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 examples of abstraction of JAVA programs that we have analyzed with the backward approach in [12].

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. We then define an algorithm to compute this weak coverability set and illustrate its behavior on a simple example.

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 is the target of the transfer B_i.

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

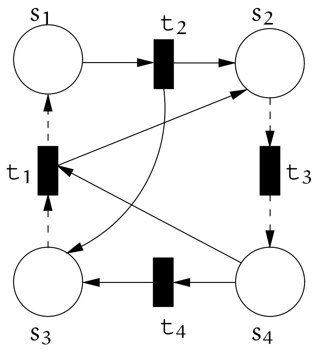

Figure 8: An example of Multi Isolated Transfer Net.

(i) a place cannot be a source of two different transfers;

(ii) a place cannot be the target of one transfer and source in another one;

(iii) a place source of a transfer cannot be source of the Petri Net part of a multi transfer.

and formally as follows:

(i) $\forall p \in \mathcal{P} : \neg\exists M_i, M_j \in \mathcal{B}$ with $\langle P_i, p_i\rangle \in \mathsf{Transfer}(M_i)$, $\langle P_j, p_j\rangle \in \mathsf{Transfer}(M_j)$, $(\langle P_i, p_i\rangle \neq \langle P_j, p_j\rangle)$, $p \in P_i$ and $p \in P_j$.

(ii) $\neg\exists p \in \mathcal{P}$ such that $\exists M_i, M_j \in \mathcal{B}$ with $\langle P_i, p_i\rangle \in \mathsf{Transfer}(M_i), \langle P_j, p_j\rangle \in \mathsf{Transfer}(M_j)$ and $p \in P_i$ 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. 8. We now define a weak notion of coverability set that is parametrized by a upward closed set U.

Definition 10 (Weak Coverability Set) Given a MTN $\mathcal{M}$, a set of initial ω-markings S_0 and an upward closed set U defined by an unique minimal marking $\mathbf{m}_U$ [5], a weak coverability set of $\mathcal{M}$ according to S_0 and $\mathbf{m}_U$ (noted $\mathtt{WCS}(\mathcal{M}, S_0, \mathbf{m}_U)$) is a set of ω-markings such that the two following conditions holds:

(1) $\forall \mathbf{m} \in \mathtt{WCS}(\mathcal{M}, S_0, \mathbf{m}_U) \setminus \mathtt{Reach}(\mathcal{M}, S_0)$, there exists an infinite sequence $\mathbf{m}_1 \preceq \mathbf{m}_2 \preceq \ldots \preceq \mathbf{m}_n \preceq \ldots$ with $\mathbf{m}_i \in \mathtt{Reach}(\mathcal{M}, S_0)$ for any $i \geq 1$ and such that:

[5] 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.

- if $\mathbf{m}(p) = \omega$ then either $\mathbf{m}_i(p) > \mathbf{m}_{i-1}(p)$, for all $i > 1$, or $\mathbf{m}_i(p) \geq \mathbf{m}_U(p)$, for all $i \geq 1$.
- if $\mathbf{m}(p) \neq \omega$, we have for any $i \geq 0 : \mathbf{m}_i(p) = \mathbf{m}_{i+1}(p)$ and $\mathbf{m}(p) = \mathbf{m}_0(p)$.

(2) $\forall \mathbf{m} \in \mathtt{Reach}(\mathcal{M}, S_0)$, there exists $\mathbf{m}' \in \mathtt{WCS}(\mathcal{M}, S_0, \mathbf{m}_U)$ with $\mathbf{m} \preceq \mathbf{m}'$.

□

A weak coverability set over-approximates the set of reachable states and all the coverability sets. Nevertheless, it is still sufficiently precise to verify upward closed safety properties. On the contrary, it cannot always be used to decide place boundedness. This is formally expressed by the following proposition:

Proposition 1 *Let $\mathcal{M}$ be a MTN, an initial marking S_0 and an upward-closed set U with the minimal marking $\mathbf{m}_U$, the following holds :*

$$\mathtt{Reach}(\mathcal{M}, S_0) \subseteq \downarrow \mathtt{CS}(\mathcal{M}, S_0) \subseteq \downarrow \mathtt{WCS}(\mathcal{M}, S_0, \mathbf{m}_U)$$
$$\downarrow \mathtt{WCS}(\mathcal{M}, S_0, \mathbf{m}_U) \cap U \neq \emptyset \textbf{ iff } \mathtt{Reach}(\mathcal{M}, S_0) \cap U \neq \emptyset$$

□

6.2 Algorithm to Compute a `WCS`

In this section, we give an algorithm that computes a weak coverability set for a Multi Isolated Transfer Net $\mathcal{M}$, a set of initial markings and an upward closed set U, defined by $\mathbf{m}_U$. The algorithm is presented as an extension of the enumerative algorithm given in section 4. At each step of the construction of the tree, an untreated node annoted with the ω-marking $\mathbf{m}$ is developed 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 three cases as in the algorithm of section 4 for computing the minimal coverability set of a Petri Net. Cases (1), (3) are identical to the enumerative algorithm of section 4, we only detail case (2) that define the following acceleration (that we call `AccIsolated`) adapted to our subclass of Transfer Nets.

(2) if there is a sequence of transitions σ in the tree from a marking $\mathbf{m}''$ to $\mathbf{m}$ such that $\mathbf{m}'' \prec \mathbf{m}'$ (so, we have $\mathbf{m}'' \rightarrowtail_\sigma \mathbf{m} \rightarrowtail_M \mathbf{m}'$), then we construct $\mathbf{n}$ as follows :
we have $\mathbf{n}(p) = \omega$ **if** $\exists$ a path from $\mathbf{m}''$ to $\mathbf{m}$ corresponding to the sequence of transition σ with $\mathbf{m}'' \prec \mathbf{m}'$ and

(1) p is not the source or the target of a transition in $\sigma \cdot M$ and the marking strictly increases from $\mathbf{m}''$ to $\mathbf{m}$. More formally:

$$\forall \langle P, p_t \rangle \in \mathsf{Transfer}(\sigma \cdot M) : p \notin (\{p_t\} \cup P) \text{ and } \mathbf{m}''(p) < \mathbf{m}'(p)$$

(2) p is the target of a transfer in $\sigma \cdot M$ and either the marking of p increases strictly from $\mathbf{m}''$ to $\mathbf{m}'$ or there is a source of that transfer that increases strictly from $\mathbf{m}''$ to $\mathbf{m}'$. More formally:

$$\exists \langle P, p_t \rangle \in \mathsf{Transfer}(\sigma \cdot M) \text{ with } (p = p_t) \text{ and}$$
$$(\mathbf{m}''(p) < \mathbf{m}'(p) \vee \exists p' \in P : \mathbf{m}''(p') < \mathbf{m}'(p'))$$

(3) p is a source of a transfer in $\sigma \cdot M$ and the marking of p increases strictly from $\mathbf{m}''$ to $\mathbf{m}'$, and furthermore the marking of p is greater or equal to $\mathbf{m}_U(p)$. More formally:

$$\exists \langle P, p_t \rangle \in \mathsf{Transfer}(\sigma \cdot M) \text{ with } p \in P \text{ and } \mathbf{m}''(p) < \mathbf{m}'(p) \text{ and } \mathbf{m}(p) \geq \mathbf{m}_U(p)$$

otherwise, $\mathbf{n}(p) = \mathbf{m}'(p)$.
Finally, we take the farest predecessor $\mathbf{m}'''$ of $\mathbf{m}$ with $\mathbf{m}''' \prec \mathbf{n}$ (possibly different from $\mathbf{m}''$) and replace the subtree rooted by $\mathbf{m}'''$ by $\mathbf{n}$.

Note that as we consider Multi Isolated Transfer Net, those three cases are mutually exclusive. The following theorem states the correction of the algorithm defined with the acceleration above :

Theorem 1 *If $\mathcal{M}$ is a Multi Isolated Transfer Net, S_0 a finite set of initial ω-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`*, terminates and 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 transfer). 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 strictly 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 of the transfer, 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

We have seen in the previous subsection that our acceleration for Multi Isolated Transfer Nets does not always put a ω in a source that is increasing. Note that this is the only difference with the algorithm for Petri Net. Applying the usual algorithm for Petri net to a MTN may result in an over-approximation. We illustrate this phenomenon on an example.

Fig. 9(a) shows a simple example of Multi Isolated Transfer Net. We consider $\langle 1,0,0,\omega\rangle$ as initial ω-marking. Fig. 9(b-d) shows the computations of algorithms presented in the previous sections. Fig. 9(b) presents the tree of the enumerative algorithm for Petri Net after two iterations. At this step, it detects that the successor for the transition b is greater than the initial ω-marking $\langle 1,0,0,\omega\rangle$. Thus, case (2) is applied (so, here we forget that s_3 is a source of a transfer in a) : ω is added to the initial ω-marking in s_3, the initial marking is then replaced and the procedure continue from this new unique node. Finally, the algorithm ends after computing the tree shown in Fig. 9(c).

According to the upward closed set $s_2 \geq 1 \wedge s_3 \geq 1$, the algorithm specialized for Multi Isolated Transfer net computes the same tree. Thus, the computed weak coverability set coincides with the over-approximation computed by the algorithm for Petri net 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 "no reachable markings satisfies $s_3 \geq 2$", only our algorithm computing the weak coverability set returns the right answer. At the second step, as shown by Fig. 9(b) the algorithm also detects that the successor for transition b is greater than the initial marking but doesn't put ω for the place s_3 because s_3 is the source of a transfer and the number of tokens is not enough to intersect with the upward closed set $s_3 \geq 2$. Fig. 9(d) shows the final tree computed by the new algorithm.

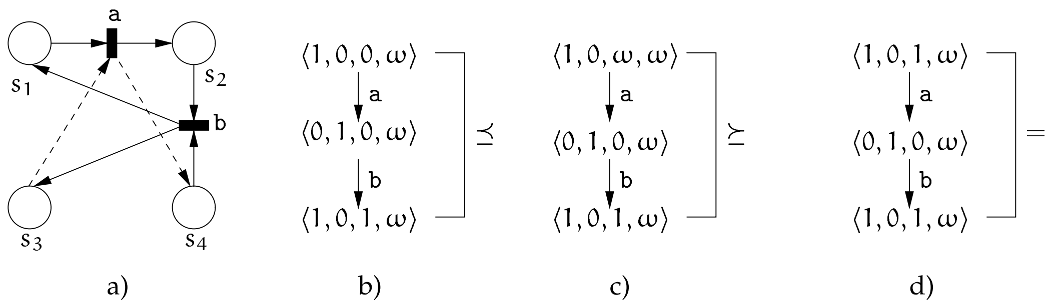

Figure 9: An example of Isolated Multi-transfer Net.

7 The General Case: Combination of Forward and Backward Search

As we already recalled the forward approach cannot be extended to the full class of MTNs [8]. But as shown in Fig. 7, backward approach seems to be more *explosion prone* and seems to be useful only when some information about potential reachable markings can be used to *guide* the search. Structural invariants have been used to prune the backward search space in [10, 12]. Unfortunately, results of Fig. 7 show that this technique does not always speed up the search.

Fig. 10 shows results on some MTNs corresponding to abstractions of Java programs analysed with the symbolic backward algorithm of [12]. Without information to prune the search space, the backward algorithm can take a lot of time, about one hour for the P/C_{bug} example and for the corrected version of the model. In those particular examples, the over-approximation computed by the structural invariants is very effective, compare the columns in the table of Fig. 10 giving the time for the computation of Pre^* without and with the invariant heuristic. But this is not always the case.

On the other hand, the forward symbolic algorithm of section 4 has always very short execution times on those examples. Unfortunately, that algorithm only computes an over-approximation of the coverability set for that class of system. For the $P/C^1_{correct}$ and P/C^2 examples, the algorithm of section 4 established that the bad markings were not reachable. In the other cases, that is for P/C^1_{bug} and the I/D systems, the forward over-approximation cannot be conclusive as bad markings are reachable. Note that bad markings are really reachable but we can not decide it with the over-approximation given by this algorithm. So in those cases, the backward algorithm has to be applied [6].

Case Study	**P**	**T**	$\mathbf{NN}_{\mathsf{Post}}$	$\mathbf{NE}_{\mathsf{Post}}$	$\mathbf{I}_{\mathsf{Post}}$	$\mathbf{EX}_{\mathsf{Post}}$	$\mathbf{M}_{\mathsf{Post}}$	$\mathbf{NN}_{\mathsf{Pre}}$	$\mathbf{NE}_{\mathsf{Pre}}$	$\mathbf{I}_{\mathsf{Pre}}$	$\mathbf{EX}^1_{\mathsf{Pre}}$	$\mathbf{N}_{\mathsf{Pre}}$	$\mathbf{EX}^2_{\mathsf{Pre}}$
I/D	32	28	218	40	73	3.16s	2728kb	800	3073	30	20.11s	6008kb	3.3s
P/C^1_{bug}	44	37	688	93	78	37.4s	4772kb	8418	12234	29	1h12m	33884kb	9.21s
$P/C^1_{correct}$	44	37	306	25	18	0.63s	2248kb	12479	8396	29	55m50s	31884kb	0.02s
P/C^2	20	16	107	14	33	0.23s	1732kb	566	810	19	3.1s	3012kb	0.04s

Figure 10: verification of MTNson an AMD Athlon 900Mhz 500Mbytes

Let us now show that the forward approximation computed by the algorithm of section 4 can give more information than the structural invariants. The parametrized MTN of the MESI protocol (see [18]) has only one structural invariant which says that the number of tokens in all the places is constant in all the forward reachable markings. As the place invalid can contain any number of tokens (it contains ω tokens in the initial ω-markings), this invariant does not give any information to prune the backward search (see [10] for more 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 $\langle\mathsf{invalid}, \mathsf{shared}, \mathsf{modified}, \mathsf{exclusive}\rangle$). This over-approximation allows us to verify the mutual exclusion property between shared and exclusive. Other interesting properties cannot be verified with this over-approximation. As an example, consider the mutual exclusion property that asks that at most one token can be in exclusive. The backward approach can be applied efficiently to answer this question by eliminating during the 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.

First, if we have to verify that a Multi Transfer Net that does not fall in the class of Multi Isolated Transfer Net cannot reach an upward closed set of Bad markings U, we first compute an over-approximation of the minimal coverability set with the symbolic procedure defined in section 4. Let us note $\mathcal{O}$ this set of ω-markings. Then, we check if $\downarrow \mathcal{O} \cap U \neq \emptyset$. If this is the case, then we are done, we know that the MTN cannot reach U. Otherwise, we can use $\mathcal{O}$ in conjunction with the backward search as follows. Instead of computing $\mu X \cdot U \cup \mathsf{Pre}(X)$ we compute

[6] 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 and we do not have currently an efficient symbolic implementation of that algorithm

$\mu X \cdot ((U \sqcap \downarrow \mathcal{O}) \cup (\mathsf{Pre}(X) \sqcap \downarrow \mathcal{O}))$ where $\sqcap$ is defined as :

$$\sqcap : \text{ upward closed set } \times \text{ downward closed set } \rightarrow \text{upward closed set}$$

$$S \sqcap T = \{\mathbf{m} | \exists \mathbf{m}' \in S : \mathbf{m}' \preceq \mathbf{m} \text{ and } \mathbf{m}' \in T\}$$

So, we remove from S any marking whose upward closure does not intersect with T. This is mainly the same idea that we use in the invariant heuristic defined in [10]. But the information computed using the forward search is always at least as precise as the one computed using the structural invariants and often much more precise. We will experiment with this heuristic in the near future.

References

[1] 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.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Series in Parallel Computing. John Wiley & Sons, 1995.

[3] 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.

[4] 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.

[5] B. Bérard and L. Fribourg. Reachability analysis of (timed) Petri nets using real arithmeti In *Proc. CONCUR'99*, LNCS 1664, pages 178-193, 1999.

[6] 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.

[7] G. Ciardo, A.S. Miner. Storage Alternatives for Large Structured State Space. In *Proc. Modelling Techniques and Tools for Computer Performance Evaluation*, LNCS 1245, pages 44-57, 1997.

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

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

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

[11] G. Delzanno, J-F. Raskin and L. Van Begin. Covering Sharing Trees: Efficient Data Structures for the Automated Verification of Parameterized Systems. *http://www.ulb.ac.be/di/ssd/jfr/CST.ps*, 2002.

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

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

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

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

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

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

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

[19] 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.

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

[21] P. Marenzoni, S. Caselli, G. Conte. Analysis of Large GSPN Models : A Distributed Solution Tool. In *Proc. Int. Work. on Petri Nets and Performance*, 1997.

[22] C. Rackoff. The Covering and Boundness Problem for Vector Addition Systems. *Theoretical Computer Science* 6, 223, 1978.

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

[24] 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.

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