Minimal Coverability Set for Petri Nets: Karp and Miller Algorithm with Pruning

Pierre-Alain Reynier[1], Frédéric Servais[2]

[1] LIF, Université Aix-Marseille & CNRS, France
[2] Department of Computer & Decision Engineering (CoDE), ULB, Belgium

Abstract. This paper presents the Monotone-Pruning algorithm (MP) for computing the minimal coverability set of Petri nets. The original Karp and Miller algorithm (K&M) unfolds the reachability graph of a Petri net and uses acceleration on branches to ensure termination. The MP algorithm improves the K&M algorithm by adding pruning between branches of the K&M tree. This idea was first introduced in the Minimal Coverability Tree algorithm (MCT), however it was recently shown to be incomplete. The MP algorithm can be viewed as the MCT algorithm with a slightly more aggressive pruning strategy which ensures completeness. Experimental results show that this algorithm is a strong improvement over the K&M algorithm as it dramatically reduces the exploration tree.

1 Introduction

Petri nets form an important formalism for the description and analysis of concurrent systems. While the state space of a Petri net may be infinite, many verification problems are decidable. The minimal coverability set (MCS) [2] is a finite representation of a well-chosen over-approximation of the set of reachable markings. As proved in [2], it can be used to decide several important problems. Among them we mention the *coverability* problem to which many safety problems can be reduced (is it possible to reach a marking dominating a given one?); the *boundedness* problem (is the set of reachable markings finite?); the *place boundedness* problem (given a place p, is it possible to bound the number of tokens in p in any reachable marking?); the *semi-liveness* problem (is there a reachable marking in which a given transition is enabled?). Finally, the *regularity problem* asks whether the set of reachable markings is regular.

Karp and Miller (K&M) introduced an algorithm for computing the MCS [8]. This algorithm builds a finite tree representation of the (potentially infinite) unfolding of the reachability graph of the given Petri net. It uses acceleration techniques to collapse branches of the tree and ensure termination. By taking advantage of the fact that Petri nets are strictly monotonic transition systems, the acceleration essentially computes the limit of repeatedly firing a sequence of transitions. The MCS can be extracted from the K&M tree. The K&M Algorithm thus constitutes a key tool for Petri nets, and has been extended to other classes of well-structured transition systems [1].

However, the K&M Algorithm is not efficient and in real-world examples it often does not terminate in reasonable time. One reason is that in many cases it will compute several times a same subtree; two nodes labelled with the same marking will produce the same subtree, K&M will compute both. This observation lead to the MCT algorithm [2]. This algorithm introduces clever optimizations for ensuring that all markings in the tree are incomparable. At each step the new node is added to the tree only if its marking is not smaller than the marking of an existing node. Then, the tree is pruned: each node labelled with a marking that is smaller than the marking of the new node is removed together with all its successors. The idea is that a node that is not added or that is removed from the tree should be covered by the new node or one of its successors. It was recently shown that the MCT algorithm is incomplete [7]. The flaw is intricate and, according to [7], difficult to patch. As an illustration, an attempt to resolve this issue has been done in [9]. However, as proved in [6], the algorithm proposed in [9] may not terminate. In [7], an alternative algorithm, the CoverProc algorithm, is proposed for the computation of the MCS of a Petri net. This algorithm follows a different approach and is not based on the K&M Algorithm.

We propose here the Monotone-Pruning algorithm (MP), an improved K&M algorithm with pruning. This algorithm can be viewed as the MCT Algorithm with a slightly more aggressive pruning strategy which ensures completeness. The MP algorithm constitutes a simple modification of the K&M algorithm, and is thus easily amenable to implementation and to extensions to other classes of systems [1, 3, 4]. Moreover, as K&M Algorithm, and unlike the algorithm proposed in [7], any strategy of exploration of the Petri net is correct: depth first, breadth first, random... It is thus possible to develop heuristics for subclasses of Petri nets. Finally experimental results show that our algorithm is a strong improvement over the K&M Algorithm, it indeed dramatically reduces the exploration tree. In addition, MP Algorithm is also amenable to optimizations based on symbolic computations, as proposed in [5] for MCT.

While the algorithm in itself is simple and includes the elegant ideas of the original MCT Algorithm, the proof of its correctness is long and technical. In fact, the difficult part is its completeness, *i.e.* any reachable marking is covered by an element of the set returned by the algorithm. To overcome this difficulty, we reduce the problem to the correction of the algorithm for a particular class of finite state systems, which we call widened Petri nets (WPN). These are Petri nets whose semantics is widened w.r.t. a given marking m: as soon as the number of tokens in a place p is greater than $m(p)$, this value is replaced by ω. Widened Petri nets generate finite state systems for which the proof of correction of the Monotone-Pruning algorithm is easier as accelerations can be expressed as finite sequences of transitions.

Definitions of Petri nets and widened Petri nets are given in Section 2, together with the notions of minimal coverability set and reachability tree. In Section 3, we recall the K&M Algorithm and present the Monotone-Pruning Algorithm. We compare it with the MCT Algorithm, and prove its termination and correction under the assumption that it is correct on WPN. Finally, in Section 4, we develop the proof of its correction on widened Petri nets. Experimental results are given in Section 5. Omitted proofs can be found in Appendix.

2 Preliminaries

$\mathbb{N}$ denotes the set of natural numbers. To denote that the union of two sets X and Y is disjoint, we write $X \uplus Y$. A quasi order $\leq$ on a set S is a reflexive and transitive relation on S. Given a quasi order $\leq$ on S, a state $s \in S$ and a subset X of S, we write $s \leq X$ iff there exists an element $s' \in X$ such that $s \leq s'$.

Given a finite alphabet Σ, we denote by Σ^* the set of words on Σ, and by ε the empty word. We denote by $\prec$ the (strict) prefix relation on Σ^*: given $u, v \in \Sigma^*$ we whave $u \prec v$ iff there exists $w \in \Sigma^*$ such that $uw = v$ and $w \neq \varepsilon$. We denote by $\preceq$ the relation obtained as $\prec \cup =$.

2.1 Markings, ω-markings and labelled trees

Given a finite set P, a *marking on* P is an element of the set $\mathsf{Mark}(P) = \mathbb{N}^P$. The set $\mathsf{Mark}(P)$ is naturally equipped with a partial order denoted $\leq$.

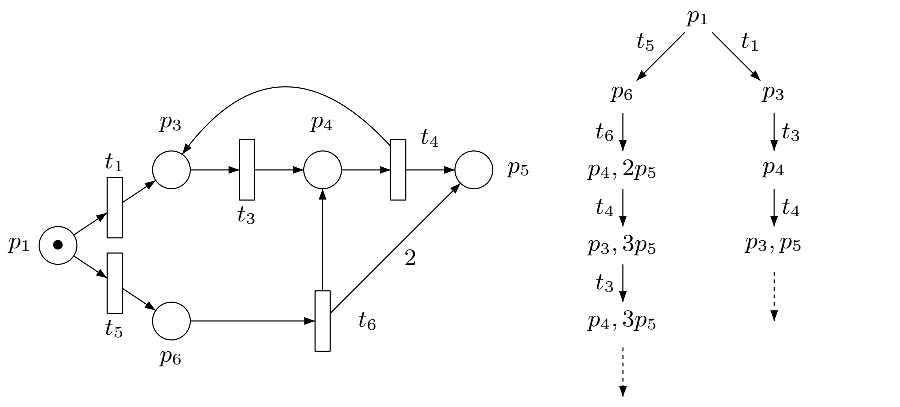
Given a marking $m \in \mathsf{Mark}(P)$, we represent it by giving only the positive components. For instance, $(1, 0, 0, 2)$ on $P = (p_1, p_2, p_3, p_4)$ is represented by the multiset $\{p_1, 2p_4\}$. An *ω-marking on* P is an element of the set $\mathsf{Mark}^\omega(P) = (\mathbb{N} \cup \{\omega\})^P$. The order $\leq$ on $\mathsf{Mark}(P)$ is naturally extended to this set by letting $n < \omega$ for any $n \in \mathbb{N}$, and $\omega \leq \omega$. Addition and substraction on $\mathsf{Mark}^\omega(P)$ is obtained using the rules $\omega + n = \omega - n = \omega$ for any $n \in \mathbb{N}$. The ω-marking $(\omega, 0, 0, 2)$ on $P = (p_1, p_2, p_3, p_4)$ is represented by the multiset $\{\omega p_1, 2p_4\}$.

Given two sets Σ_1 and Σ_2, a labelled tree is a tuple $\mathcal{T} = (N, n_0, E, \Lambda)$ where N is the set of nodes, $n_0 \in N$ is the root, $E \subseteq N \times \Sigma_2 \times N$ is the set of edges labelled with elements of Σ_2, and $\Lambda : N \to \Sigma_1$ labels nodes with elements of Σ_1. We extend the mapping Λ to sets of nodes: for $S \subseteq N$, $\Lambda(S) = \{\Lambda(n) \mid n \in S\}$. Given a node $n \in N$, we denote by $\mathsf{Ancestor}_\mathcal{T}(n)$ the set of ancestors of n in $\mathcal{T}$ (n included). If n is not the root of $\mathcal{T}$, we denote by $\mathsf{parent}_\mathcal{T}(n)$ its first ancestor in $\mathcal{T}$. Finally, given two nodes x and y such that $x \in \mathsf{Ancestor}_\mathcal{T}(y)$, we denote by $\mathsf{path}_\mathcal{T}(x, y) \in E^*$ the sequence of edges leading from x to y in $\mathcal{T}$. We also denote by $\mathsf{pathlabel}_\mathcal{T}(x, y) \in \Sigma_2^*$ the label of this path.

2.2 Petri nets

Definition 1 (Petri nets (PN)). *A Petri net $\mathcal{N}$ is a tuple (P, T, I, O, m_0) where P is a finite set of* places*, T is a finite set of* transitions *with $P \cap T = \emptyset$, $I : T \to \mathsf{Mark}(P)$ is the backward incidence mapping, representing the* input *tokens, $O : T \to \mathsf{Mark}(P)$ is the forward incidence mapping, representing* output *tokens, and $m_0 \in \mathsf{Mark}(P)$ is the initial marking.*

The semantics of a PN is usually defined on markings, but can easily be extended to ω-markings. We define the semantics of $\mathcal{N} = (P, T, I, O, m_0)$ by its associated labeled transition system $(\mathsf{Mark}^\omega(P), m_0, \Rightarrow)$ where $\Rightarrow \subseteq \mathsf{Mark}^\omega(P) \times \mathsf{Mark}^\omega(P)$ is the transition relation defined by $m \Rightarrow m'$ iff $\exists t \in T$ s.t. $m \geq I(t) \wedge m' = m - I(t) + O(t)$. For convenience we will write, for $t \in T$, $m \xRightarrow{t} m'$ if $m \geq I(t)$ and $m' = m - I(t) + O(t)$. In addition, we also write $m' = \mathsf{Post}(m, t)$, this defines the operator Post which computes the successor of an ω-marking by a transition. Given an ω-marking m and a transition t, we write $m \overset{t}{\Rightarrow} \cdot$ iff there exists $m' \in \mathsf{Mark}^\omega(P)$ such that $m \overset{t}{\Rightarrow} m'$. The relation $\Rightarrow^*$ represents the reflexive and transitive closure of $\Rightarrow$. We say that a marking m is *reachable in* $\mathcal{N}$ iff $m_0 \Rightarrow^* m$. We say that a Petri net is bounded if the set of reachable markings is finite.



(a) A Petri net $\mathcal{N}$. (b) Prefix of its reachability tree.

Fig. 1. A Petri net with its reachability tree.

Example 1. We consider the Petri net $\mathcal{N}$ depicted on Figure 1(a), which is a simplification of the counter-example proposed in [7], but is sufficient to present our definitions. The initial marking is $\{p_1\}$, depicted by the token in the place p_1. This net is not bounded as place p_5 can contain arbitrarily many tokens. The execution of the Monotone-Pruning algorithm on the original counter-example of [7] can be found in Appendix D. ⌟

2.3 Minimal Coverability Set of Petri Nets

We recall the definition of minimal coverability set introduced in [2].

Definition 2. *A coverability set of a Petri net* $\mathcal{N} = (P, T, I, O, m_0)$ *is a finite subset* C *of* $\mathsf{Mark}^\omega(P)$ *such that the two following conditions hold:*

1) *for every reachable marking* m *of* $\mathcal{N}$*, there exists* $m' \in C$ *such that* $m \leq m'$*,*

2) *for every $m' \in C$, either m' is reachable in $\mathcal{N}$ or there exists an infinite strictly increasing sequence of reachable markings $(m_n)_{n\in\mathbb{N}}$ converging to m'.*

A coverability set is minimal iff no proper subset is a coverability set. We denote by $\mathrm{MCS}(\mathcal{N})$ *the minimal coverability set of $\mathcal{N}$.*

Note that two elements of a minimal coverability set are incomparable. Computing a minimal coverability set from a coverability set is easy. Note also that if the PN is bounded, then a set is a coverability set iff it contains all maximal reachable markings.

Example 2 (Example 1 continued). The MCS of the Petri net $\mathcal{N}$ is composed of the following ω-markings: $\{p_1\}$, $\{p_6\}$, $\{p_3, \omega p_5\}$, and $\{p_4, \omega p_5\}$.

2.4 Reachability tree of Petri nets

We recall the notion of reachability tree for a PN. This definition corresponds to the execution of the PN as a labelled tree. We require it to be coherent with the semantics of PN (soundness), to be complete w.r.t. the firable transitions, and to contain no repetitions. Naturally, this reachability tree may be infinite as soon as the PN has an infinite execution.

Definition 3 (Reachability tree of a PN). *The reachability tree of a* PN $\mathcal{N} = (P, T, I, O, m_0)$ *is (up to isomorphism) a labelled tree* $\mathcal{R} = (N, n_0, E, \Lambda)$, *with* $E \subseteq N \times T \times N$ *and* $\Lambda : N \to \mathsf{Mark}(P)$, *s. t.:*

Root: $\Lambda(n_0) = m_0$,
Sound: $\forall (n, t, n') \in E, \Lambda(n) \overset{t}{\Rightarrow} \Lambda(n')$,
Complete: $\forall n \in E, \forall t \in T, \left(\exists m \mid \Lambda(n) \overset{t}{\Rightarrow} m\right) \Rightarrow (\exists n' \in N \mid (n, t, n') \in E)$
Unicity: $\forall n \in N, \forall t \in T, (n, t, n') \in E \wedge (n, t, n'') \in E \Rightarrow n' = n''$

Using notations introduced for labelled trees, the following property holds:

Lemma 1. $\forall x, y \mid x \in \mathsf{Ancestor}_{\mathcal{R}}(y), \Lambda(y) = \mathsf{Post}(\Lambda(x), \mathsf{pathlabel}_{\mathcal{R}}(x, y))$.

Example 3 (Example 1 continued). A prefix of the reachability tree of $\mathcal{N}$ is depicted on Figure 1(b). Each node is represented by its label (a marking). ⌟

2.5 Widened Petri nets

We present an operation which, given a (potentially unbounded) Petri net, turns it into a finite state system. Let P be a finite set, and $\varphi \in \mathsf{Mark}(P)$ be a marking. We consider the *finite* set of ω-markings whose finite components (*i.e.* values different from ω) are less or equal than φ. Formally, we define $\mathsf{Mark}^{\omega}_{\varphi}(P) = \{m \in \mathsf{Mark}^{\omega}(P) \mid \forall p \in P, m(p) \leq \varphi(p) \vee m(p) = \omega\}$. The widening operator Widen_{φ} maps an ω-marking into an element of $\mathsf{Mark}^{\omega}_{\varphi}(P)$: $\forall m \in \mathsf{Mark}^{\omega}(P)$,

$$\forall p \in P, \mathsf{Widen}_{\varphi}(m)(p) = \begin{cases} m(p) & \text{if } m(p) \leq \varphi(p) \\ \omega & \text{otherwise.} \end{cases}$$

Note that this operator trivially satisfies $m \leq \mathsf{Widen}_{\varphi}(m)$.

Definition 4 (Widened Petri net). *A widened Petri net (*WPN *for short) is a pair* $(\mathcal{N}, \varphi)$ *composed of a* PN $\mathcal{N} = (P, T, I, O, m_0)$ *and of a marking* $\varphi \in \mathsf{Mark}(P)$ *such that* $m_0 \leq \varphi$.

The semantics of $(\mathcal{N}, \varphi)$ is given by its associated labelled transition system $(\mathsf{Mark}^{\omega}_{\varphi}(P), m_0, \Rightarrow_{\varphi})$ where for $m, m' \in \mathsf{Mark}^{\omega}_{\varphi}(P)$, and $t \in T$, we have $m \overset{t}{\Rightarrow}_{\varphi} m'$ iff $m' = \mathsf{Widen}_{\varphi}(\mathsf{Post}(m,t))$. We carry over from PN to WPN the notions of reachable marking, reachability tree... We define the operator Post_{φ} by $\mathsf{Post}_{\varphi}(m,t) = \mathsf{Widen}_{\varphi}(\mathsf{Post}(m,t))$. Subscript φ may be omitted when it is clear from the context. Finally, the minimal coverability set of a widened Petri net $(\mathcal{N}, \varphi)$ is simply the set of its maximal reachable states as its reachability set is finite. It is denoted $\mathrm{MCS}(\mathcal{N}, \varphi)$.

We state the following result, whose proof easily follows by induction.

Proposition 1. *Let* $(\mathcal{N}, \varphi)$ *be a* WPN, *and* m *be a reachable marking of* $\mathcal{N}$. *Then there exists an* ω*-marking* m' *reachable in* $(\mathcal{N}, \varphi)$ *such that* $m \leq m'$.

Example 4 (Example 1 continued).

Consider the mapping φ associating 1 to places p_1, p_3, p_4 and p_6, and 2 to place p_5, and the widened Petri net $(\mathcal{N}, \varphi)$. Then for instance from marking $\{p_5, p_6\}$, the firing of t_6 results in the marking $\{p_4, \omega p_5\}$, instead of the marking $\{p_4, 3p_5\}$ in the standard semantics. Similarly, consider the prefix of the reachability tree of $\mathcal{N}$ depicted on Figure 1(b). For $(\mathcal{N}, \varphi)$, the reachability tree is obtained by substituting the ω-marking $\{p_3, \omega p_5\}$ (resp. $\{p_4, \omega p_5\}$) to the marking $\{p_3, 3p_5\}$ (resp. $\{p_4, 3p_5\}$), as we have $\varphi(p_5) = 2$. One can compute the MCS of this WPN. Due to the choice of φ, it coincides with the MCS of $\mathcal{N}$. ⌟

3 Monotone-Pruning Algorithm

3.1 Karp and Miller Algorithm.

The K&M Algorithm [8] is a well known solution to compute a coverability set of a PN. It is represented as Algorithm 1 (with a slight modification as in [8], the algorithm computes simultaneously all the successors of a marking). K&M algorithm uses an external acceleration function $\mathsf{Acc} : 2^{\mathsf{Mark}^{\omega}(P)} \times \mathsf{Mark}^{\omega}(P) \to \mathsf{Mark}^{\omega}(P)$ which is defined as follows:

$$\forall p \in P, \mathsf{Acc}(M, m)(p) = \begin{cases} \omega & \text{if } \exists m' \in M \mid m' < m \wedge m'(p) < m(p) < \omega \\ m(p) & \text{otherwise.} \end{cases}$$

K&M Algorithm builds a tree in which nodes are labelled by ω-markings and edges by transitions of the Petri net. Roughly, it consists in exploring the reachability tree of the PN, and in applying the acceleration function Acc on branches of this tree. Note that the acceleration may compute ω-markings that are not reachable. The correction of this procedure relies on the strict monotonicity of PN and on the fact that the order $\leq$ on ω-markings is well-founded.

Theorem 1 ([8]). *Let* $\mathcal{N}$ *be a* PN. K&M *algorithm terminates and computes a coverability set of* $\mathcal{N}$.

Algorithm 1 The K&M Algorithm

```
Require: A Petri net $\mathcal{N} = (P, T, I, O, m_0)$.
Ensure: A labelled tree $\mathcal{C} = (X, x_0, B, \Lambda)$ such that $X$ is a coverability set of $\mathcal{N}$.
 1: Let $x_0$ be a new node such that $\Lambda(x_0) = m_0$.
 2: $X := \{x_0\}$; $\mathsf{Wait} := \{(x_0, t) \mid \Lambda(x_0) \overset{t}{\Rightarrow} \cdot\}$; $B := \emptyset$;
 3: while $\mathsf{Wait} \neq \emptyset$ do
 4:   Pop $(n', t)$ from $\mathsf{Wait}$. $m := \mathsf{Post}(\Lambda(n'), t)$;
 5:   if $\nexists y \in \mathsf{Ancestor}_{\mathcal{C}}(n') \mid \Lambda(y) = m$ then
 6:     Let $n$ be a new node s.t. $\Lambda(n) = \mathsf{Acc}(\Lambda(\mathsf{Ancestor}_{\mathcal{C}}(n')), m)$;
 7:     $X+ = \{n\}$; $B+ = \{(n', t, n)\}$; $\mathsf{Wait}+ = \{(n, u) \mid \Lambda(n) \overset{u}{\Rightarrow} \cdot\}$;
 8:   end if
 9: end while
10: Return $\mathcal{C} = (X, x_0, B, \Lambda)$.
```

3.2 Definition of the algorithm

We present in this section our algorithm which we call Monotone-Pruning Algorithm as it includes a kind of horizontal pruning. We denote this algorithm by MP. It involves the acceleration function Acc used in the Karp and Miller algorithm. However, it is applied in a slightly different manner.

Algorithm 2 Monotone Pruning Algorithm for Petri Nets.

```
Require: A Petri net $\mathcal{N} = (P, T, I, O, m_0)$.
Ensure: A labelled tree $\mathcal{C} = (X, x_0, B, \Lambda)$ and a partition $X = \mathsf{Act} \uplus \mathsf{Inact}$
    such that $\Lambda(\mathsf{Act}) = \mathrm{MCS}(\mathcal{N})$.
 1: Let $x_0$ be a new node such that $\Lambda(x_0) = m_0$;
 2: $X := \{x_0\}$; $\mathsf{Act} := X$; $\mathsf{Wait} := \{(x_0, t) \mid \Lambda(x_0) \overset{t}{\Rightarrow} \cdot\}$; $B := \emptyset$;
 3: while $\mathsf{Wait} \neq \emptyset$ do
 4:   Pop $(n', t)$ from $\mathsf{Wait}$.
 5:   if $n' \in \mathsf{Act}$ then
 6:     $m := \mathsf{Post}(\Lambda(n'), t)$;
 7:     Let $n$ be a new node such that $\Lambda(n) = \mathsf{Acc}(\Lambda(\mathsf{Ancestor}_{\mathcal{C}}(n') \cap \mathsf{Act}), m)$;
 8:     $X+ = \{n\}$; $B+ = \{(n', t, n)\}$;
 9:     if $\Lambda(n) \not\leq \Lambda(\mathsf{Act})$ then
10:       $\mathsf{Act}- = \{x \mid \exists y \in \mathsf{Ancestor}_{\mathcal{C}}(x). \Lambda(y) \leq \Lambda(n) \wedge (y \in \mathsf{Act} \vee y \notin \mathsf{Ancestor}_{\mathcal{C}}(n))\}$;
11:       $\mathsf{Act}+ = \{n\}$; $\mathsf{Wait}+ = \{(n, u) \mid n \overset{u}{\Rightarrow} \cdot\}$;
12:     end if
13:   end if
14: end while
15: Return $\mathcal{C} = (X, x_0, B, \Lambda)$ and $(\mathsf{Act}, \mathsf{Inact})$.
```

As Karp and Miller Algorithm, MP Algorithm builds a tree $\mathcal{C}$ in which nodes are labelled by ω-markings and edges by transitions of the Petri net. Therefore it proceeds in an exploration of the reachability tree of the Petri net, and uses acceleration along branches to reach the “limit” markings. In addition, it can prune branches that are covered by nodes on other branches. Therefore, nodes of the tree are partitionned in two subsets: active nodes, and inactive ones. Intuitively, active nodes will form the minimal coverability set of the Petri net, while inactive ones are kept to ensure completeness of the algorithm.

Given a pair (n', t) popped from Wait, the introduction in $\mathcal{C}$ of the new node obtained from (n', t) proceeds in the following steps:

1. node n' should be active (test of Line 5) ;
2. the "regular" successor marking is computed: $m = \mathsf{Post}(\Lambda(n'), t)$ (Line 6) ;
3. this marking is accelerated w.r.t. the *active ancestors* of node n', and a new node n is created with this marking: $\Lambda(n) = \mathsf{Acc}(\Lambda(\mathsf{Ancestor}_{\mathcal{C}}(n') \cap \mathsf{Act}), m)$ (Lines 7 and 8) ;
4. the new node n is declared as active if, and only if, it is not covered by an existing active node (test of Line 9 and Line 11) ;
5. update of Act: some nodes are "deactivated" (Line 10).

We detail the update of the set Act. Intuitively, one wants to deactivate nodes (and their descendants) that are covered by the new node n. This would lead to deactivate a node x iff it owns an ancestor y dominated by n, *i.e.* such that $\Lambda(y) \leq \Lambda(n)$. This condition has to be refined to obtain a correct algorithm (see Remark 1). In MP Algorithm (see Line 10), node x is deactivated iff its ancestor y is either active ($y \in \mathsf{Act}$), or is not itself an ancestor of n ($y \notin \mathsf{Ancestor}_{\mathcal{C}}(n)$). In this case, we say that x *is deactivated by* n. This subtle condition constitutes the main difference between MP and MCT Algorithms (see Remark 1).

Consider the introduction of a new node n obtained from $(n', t) \in \mathsf{Wait}$, and a node y such that $\Lambda(y) \leq \Lambda(n)$, y can be used to deactivate nodes in two ways:

– if $y \notin \mathsf{Ancestor}_{\mathcal{C}}(n)$, then no matter whether y is active or not, all its descendants are deactivated (represented in gray on Figure 2(a)),
– if $y \in \mathsf{Ancestor}_{\mathcal{C}}(n)$, then y must be active ($y \in \mathsf{Act}$), and in that case all its descendants are deactivated, except node n itself as it is added to Act at Line 11 (see Figure 2(b)).

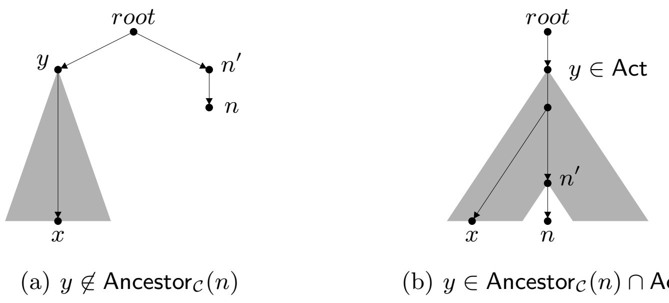

root
y
n'
n
x
root
y ∈ Act
n'
x
n
(a) $y \notin \mathsf{Ancestor}_{\mathcal{C}}(n)$
(b) $y \in \mathsf{Ancestor}_{\mathcal{C}}(n) \cap \mathsf{Act}$

Fig. 2. Deactivations of MP Algorithm.

The main result of the paper is that MP Algorithm terminates and is correct:

Theorem 2. *Let $\mathcal{N}$ be a* PN. MP *algorithm terminates and computes a minimal coverability set of $\mathcal{N}$.*

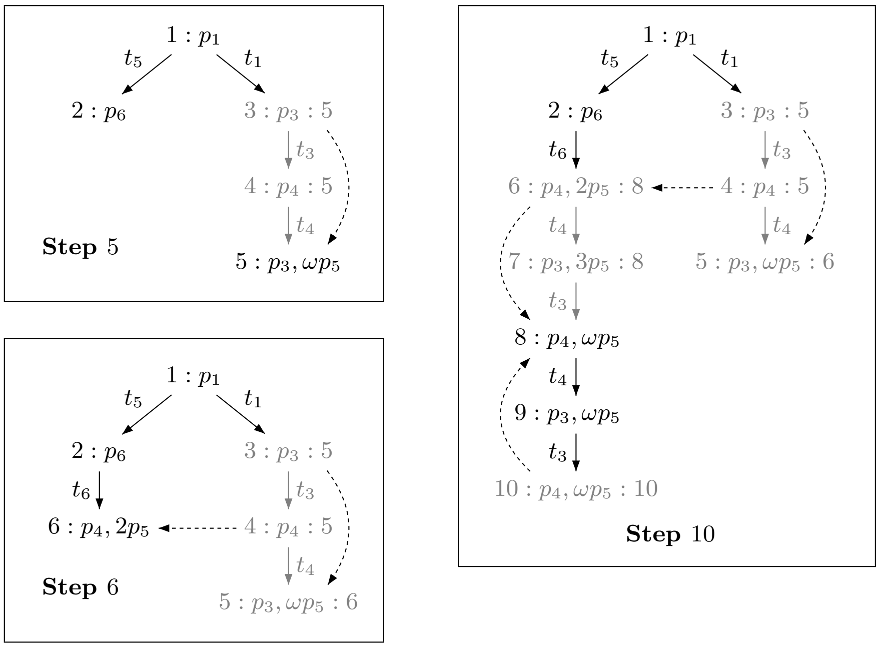


Fig. 3. Snapshots of the execution of MP Algorithm on PN $\mathcal{N}$.

Example 5 (Example 1 continued). We consider the execution of MP Algorithm on the PN $\mathcal{N}$. Three intermediary steps (5, 6 and 10) are represented on Figure 3. The numbers written on the left (before the separator ":") of nodes indicate the order in which nodes are created. Nodes that are deactivated are represented in light gray, and dashed arrows indicate how nodes are deactivated. In addition, the number of the node in charge ot the deactivation is represented at the right (after the separator ":"). In the following explanations, node n_i denotes the node that has been created at step i:

- At step 5, the new node n_5 ($\{p_3, p_5\}$) covers node n_3 ($\{p_3\}$), which is thus deactivated, together with its descendants, except node n_5 that is just added.
- At step 6, the new node n_6 ($\{p_4, 2p_5\}$) covers node n_4 ($\{p_4\}$). This node was already deactivated but as it lies on another branch, it can be used to discard its descendants. As a consequence node n_5 is deactivated.
- At step 10, the new node n_{10} is covered by node n_8, which is still active. Thus n_{10} is immediately declared as inactive. ⌟

After step 10, MP terminates and the active nodes give the MCS of $\mathcal{N}$.

Remark 1 (Comparison with the MCT *Algorithm.).* One can verify that, except the fact that the MCT algorithm develops all successors of a node simultaneously (as K&M does), the MCT Algorithm can be obtained from the MP Algorithm by a subtle modification. The only difference comes from the deactivation of nodes. In MP, inactivate nodes can be used to deactivate nodes. In MCT, only active nodes are used to deactivate nodes. More precisely, MCT Algorithm is obtained by replacing Line 10 by the following Line :

$10'$: $\quad \mathsf{Act}- = \{x \mid \exists y \in \mathsf{Ancestor}_{\mathcal{C}}(x). \Lambda(y) \leq \Lambda(n) \wedge y \in \mathsf{Act}\};$

Thus, condition $(y \in \mathsf{Act} \vee y \notin \mathsf{Ancestor}_{\mathcal{C}}(n))$ in MP is replaced by the stronger condition $y \in \mathsf{Act}$ to obtain MCT. In particular, this shows that more nodes are pruned in MP Algorithm.

Note also that if one considers the trivial condition `true`, *i.e.* one considers all active and inactive nodes to discard nodes, then one looses the termination of the algorithm. Consider Example 5. With this condition, node n_9 covers node n_7 and thus deactivates node n_8 (this does not happen in MP as n_7 is an inactive ancestor of n_9). But then, node n_{10} covers n_8 and deactivates n_{10}, and so on.

*Remark 2 (*MP *Algorithm for widened Petri nets.).* In the sequel, we will consider the application of MP Algorithm on widened Petri nets. Let $(\mathcal{N}, \varphi)$ be a WPN. The only difference is that the operator Post (resp. $\Rightarrow$) must be replaced by the operator Post_φ (resp. $\Rightarrow_\varphi$). For WPN, MP Algorithm enjoys an additional property. One can prove by induction that all markings computed by MP are reachable in $(\mathcal{N}, \varphi)$. Indeed, the acceleration is consistent with the semantics of $(\mathcal{N}, \varphi)$, *i.e.* all markings computed by Acc belong to $\mathsf{Mark}_\varphi^\omega(P)$ (where P denotes the set of places of $\mathcal{N}$), provided the arguments of Acc do.

Example 6 (Example 4 continued). Consider the WPN $(\mathcal{N}, \varphi)$ introduced in Example 4. Running MP on this WPN also yields the trees depicted on Figure 3.

3.3 Termination of MP Algorithm

Theorem 3. MP *Algorithm terminates.*

Proof. We proceed by contradiction, and assume that the algorithm does not terminate. Let $\mathcal{C} = (X, x_0, B, \Lambda)$ and $X = \mathsf{Act} \uplus \mathsf{Inact}$ be the labelled tree and the partitions computed by MP. As $\mathcal{C}$ is of finite branching (bounded by $|T|$), there exists by König's lemma an infinite branch in this tree. We fix such an infinite branch, and write it $b = x_0 \xrightarrow{t_0} x_1 \xrightarrow{t_1} x_2 \ldots$, with $(x_i, t_i, x_{i+1}) \in B, \forall i$.

Let $n \in X \setminus \{x_i \mid i \geq 0\}$. We claim that n cannot deactivate any of the x_i's. By contradiction, if for some i, x_i is deactivated by n, then all the descendants of x_i are also deactivated, except n if it is a descendant of x_i. As n does not belong to b, this implies that for any $j \geq i$, x_j is deactivated. This is impossible because branch b is infinite and the algorithm only computes successors of active nodes (test of Line 5).

By definition of the acceleration function, two cases may occur: either one of the active ancestors is strictly dominated, and then a new ω will appear in the resulting marking ($\exists p \mid \Lambda(n)(p) = \omega > m(p)$), or no active ancestors is strictly dominated, and then the acceleration has no effect on marking m ($\Lambda(n) = m$). We say that in the first case, there is an "effective acceleration".

By definition of the semantics of a Petri net on ω-markings, once a marking has value ω on a place p, so will all its successors. Thus, as there are finitely many places, a finite number of effective accelerations can occur on branch b. We consider now the largest suffix of the branch b containing no effective accelerations: let i be the smallest positive integer such that for any $j \geq i$, we have $\Lambda(x_{j+1}) = \mathsf{Post}(\Lambda(x_j), t_j)$.

We will prove that the set $S = \{x_j \mid j \geq i\}$ is an infinite set of active nodes with pairwise incomparable markings, which is impossible as the set $\mathsf{Mark}^\omega(P)$ equipped with partial order $\leq$ is a well-founded quasi-order, yielding the contradiction.

We proceed by induction and prove that for any $j \geq i$ the set $S_j = \{x_k \mid j \geq k \geq i\}$ contains active nodes with pairwise incomparable markings. Recall that we have shown above that nodes not on b cannot deactivate nodes of b. Consider set S_i. When node x_i is created, it must be declared as active, otherwise none of its successors are built, that is impossible as b is infinite. Let $j \geq i$, assume that property holds for S_j, and consider the introduction of node x_{j+1}. We prove that x_{j+1} is active, that it deactivates no node of S_j, and that the markings of S_{j+1} are pairwise incomparable. First, as for node x_i, the new node x_{j+1} must be declared as active when it is created. Thus it is covered by no active node (Line 9). By induction, elements of S_j are active, and thus we have $\Lambda(x_{j+1}) \not\leq \Lambda(x)$ for any $x \in S_j$. Second, as we have $\Lambda(x_{j+1}) = \mathsf{Post}(\Lambda(x_j), t_j)$, we do not use an effective acceleration, and thus x_{j+1} strictly dominates none of its active ancestors. In particular, this implies that it does not deactivate any of its ancestors, and thus any element of S_j. Moreover, by induction, elements of S_j are active nodes, thus $\Lambda(x_{j+1}) \not> \Lambda(x)$ for any $x \in S_j$: elements of S_{j+1} are pairwise incomparable. □

3.4 Correction of MP Algorithm

We reduce the correction of MP Algorithm for Petri nets to the correction of this algorithm for widened Petri nets, which are finite state systems. This latter result is technical, and proved in the next section (see Theorem 5).

We use this theorem to prove:

Theorem 4. MP *Algorithm for Petri nets is correct.*

Proof. Let $\mathcal{N} = (P, T, I, O, m_0)$ be a PN, $\mathcal{C} = (X, x_0, B, \Lambda)$ be the labelled tree and $X = \mathsf{Act} \uplus \mathsf{Inact}$ be the partition built by MP Algorithm on $\mathcal{N}$. As MP Algorithm terminates, all these objects are finite. We will prove that $\Lambda(\mathsf{Act})$ is the minimal coverability set of $\mathcal{N}$.

First note that elements of $\Lambda(\mathsf{Act})$ are pairwise incomparable: this is a simple consequence of Lines 9, 10 and 11. Thus, we only have to prove that it is a coverability set.

The soundness of the construction, *i.e.* the fact that elements of $\Lambda(\mathsf{Act})$ satisfy point 2 of Definition 2, follows from the correction of the acceleration function.

To prove the completeness, *i.e.* point 1 of Definition 2, we use the correction of MP Algorithm on widened Petri nets. We can consider, for each place $p \in P$, the largest value appearing in a marking during the computation. This defines a marking $\varphi \in \mathsf{Mark}(P)$.

We consider now the widened Petri net $(\mathcal{N}, \varphi)$ and the execution of MP Algorithm on it (see Remark 2). We claim that there exists an execution of this algorithm which builds the same labelled tree $\mathcal{C}$ and the same partition. This execution is obtained by picking the same elements in the list Wait. This property can be proven by induction on the length of the execution of the algorithm. Indeed, by definition of marking φ, operators Post and Post_φ are equivalent on the markings computed by the algorithm. Thus, both algorithms perform exactly the same accelerations and compute the same ω-markings.

By correction of MP Algorithm on WPN (see Theorem 5), we obtain $\Lambda(\mathsf{Act}) = \mathrm{MCS}(\mathcal{N}, \varphi)$. By Proposition 1, any marking reachable in $\mathcal{N}$ is covered by a reachable marking of $(\mathcal{N}, \varphi)$, and thus by $\mathrm{MCS}(\mathcal{N}, \varphi) = \Lambda(\mathsf{Act})$. □

4 MP Algorithm for WPN

We devote this section to the proof that MP Algorithm is correct on WPN.

4.1 Outline

As for Petri nets, the main difficulty is to prove the completeness of the set returned by MP. Therefore, we introduce in

Subsection 4.2 a notion of *exploration of a* WPN which corresponds to a tree built on the reachability tree of the WPN, with some additional properties. This structure allows to explicit the effect of accelerations. We prove in Subsection 4.3 that MP indeed builds an exploration. In fact, other algorithms like K&M or MCT also do build explorations. Finally, we prove that the exploration built by MP is complete in Subsection 4.4: we show that any reachable marking is covered by an active node. Therefore, we introduce a notion of covering path which intuitively explicits the sequence of transitions that remain to be fired from a given active node to reach the desired marking. We prove that such covering paths are not cyclic, that is promesses are always fulfilled.

4.2 Exploration of a WPN

To build a coverability set the different algorithms we consider (K&M, MCT and MP) proceed in a similar way. Roughly, the algorithm starts with the root of the reachability tree and picks a firable transition t. Then it picks a descendant that may either be the direct child by t (no acceleration) or a descendant obtained after skipping a few nodes (acceleration), this descendant must be strictly greater than the direct child (by t). Then if this node is not covered by the previously selected (and active) nodes, a pruning may occur (not in the K&M algorithm):

some active nodes are deactivated, intuitively because the subtree rooted at the new node should cover those nodes. The process continues with active nodes.

This process can be viewed as an exploration of the reachability tree $\mathcal{R} = (N, n_0, E, \Lambda)$ in the following sense. We define below an exploration as a tuple $\mathcal{E} = (X, B, \alpha, \beta)$, where X is the subset of N explored by the algorithm, B is an edge relation on X, such that $(x, t, x') \in B$ if x' is the node built by the algorithm when processing the transition t firable from x. The function α gives the order in which nodes of X are explored by the algorithm. The function β gives the position at which a node is deactivated, *i.e.* $\beta(n) = i$ if n is deactivated (pruned) when the i-th node appears.

Definition 5 (Exploration). *Given a* WPN $(\mathcal{N}, \varphi)$ *and its reachability tree* $\mathcal{R} = (N, n_0, E, \Lambda)$*, an exploration of* $\mathcal{S}$ *is a tuple* $\mathcal{E} = (X, B, \alpha, \beta)$ *such that*

- X *is a finite subset of* N*,*
- $B \subseteq X \times T \times X$*,*
- $n_0 \in X$*,*
- $(X, n_0, B, \Lambda_{|X})$ *is a labelled tree,*
- α *is a bijection from* X *to* $\{1, \dots, |X|\}$*, and*
- β *is a mapping from* X *to* $\{1, \dots, |X|\} \cup \{+\infty\}$*.*

For any $1 \le i \le |X|$*, we define the sets* $X_i = \{x \in X \mid \alpha(x) \le i\}$, $\mathsf{Inact}_i = \{x \in X \mid \beta(x) \le i\}$*, and* $\mathsf{Act}_i = X_i \setminus \mathsf{Inact}_i$*. We let* $\mathsf{Act} = \mathsf{Act}_{|X|}$ *and* $\mathsf{Inact} = \mathsf{Inact}_{|X|}$*.*

In addition, we require the following conditions:

(i) $\alpha \le \beta$*,*
(ii) $\forall x, y \in X,\ x \in \mathsf{Ancestor}_{\mathcal{R}}(y) \Rightarrow \alpha(x) \le \alpha(y)$*,*
(iii) $\forall (x, t, y) \in B, \alpha(y) \le \beta(x)$*,*
(iv) T**-completeness:** $\forall x \in \mathsf{Act}, \forall t \in T$ *s.t.* $\Lambda(x) \overset{t}{\Rightarrow}_{\varphi} \cdot, \exists y \in X \mid (x, t, y) \in B$*,*
(v) $\forall (x, t, y) \in B$*, there exists* $z \in N$ *such that:*
  (a) $(x, t, z) \in E$*, and* $z \in \mathsf{Ancestor}_{\mathcal{R}}(y)$*,*
  (b) $\mathsf{Post}_{\varphi}(\Lambda(x), t) = \Lambda(z) \le \Lambda(y)$*.*

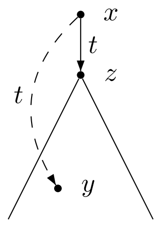


Fig. 4. Condition (v) of Definition 5.

The first condition states that nodes cannot be deactivated strictly before being selected. The second condition states that nodes are selected downward: one can not select a node that has a descendant already selected. Condition (iii) states that the algorithm explores subtrees of active nodes only. Condition (iv) enforces that all firable transitions of active nodes are explored. The last condition requires that the selected descendant is either the direct child by the selected transition t or a descendant of this child whose marking is greater than the marking of the child (acceleration). In the sequel, we denote by $\mathsf{Ancestor}_{\mathcal{E}}(\cdot)$ the ancestor relation considered in the labelled tree $(X, n_0, B, \Lambda_{|X})$. By definition, we have the following simple property: $\forall x \in X, \mathsf{Ancestor}_{\mathcal{E}}(x) = \mathsf{Ancestor}_{\mathcal{R}}(x) \cap X$.

It is easy to verify that sets Act_i and Inact_i form a partition of X_i ($X_i = \mathsf{Act}_i \biguplus \mathsf{Inact}_i$) and that sets $(\mathsf{Inact}_i)_i$ are increasing ($\mathsf{Inact}_i \subseteq \mathsf{Inact}_{i+1}, \forall i < |X|$).

Remark 3. A trivial case of exploration is obtained when relation B coincides with the restriction of relation E to the set X. This case in fact corresponds to the exploration obtained by an algorithm that would perform no acceleration.

Remark 4. It can be proven that K&M and MCT applied on WPN do build explorations. Consider K&M Algorithm. As it deactivates no node, it yields $\beta(n) = +\infty$ for any node n. However, it uses some accelerations and therefore some nodes are skipped but it respects condition (v).

4.3 MP-exploration of a WPN

Let $(\mathcal{N}, \varphi)$ be a WPN with $\mathcal{N} = (P, T, I, O, m_0)$, and $\mathcal{C} = (X, x_0, B, \Lambda)$, $X = \mathsf{Act} \uplus \mathsf{Inact}$ be the labelled tree and the partition returned by the MP Algorithm. We define here the two mappings α and β that allow to show that the labelled tree $\mathcal{C}$ can be interpreted as an exploration in the sense of Definition 5.

Mapping α. It is simply defined as the order in which elements of X are built by the MP Algorithm.

Mapping β. Initially, the set Act contains the node x_0. Any new node n can enter only once in set Act, immediately when it is added in X (Line 11) (and can thus be removed from Act at most once). We define mapping β as follows:

- if a node x never enters set Act, then we let $\beta(x) = \alpha(x)$.
- if a node x enters set Act and is never removed, then we let $\beta(x) = +\infty$.
- finally, in the last case, let x be a node which enters set Act and is removed from it during the execution of the algorithm. Nodes can only be removed from set Act at Line 10. Then let n be the node added to X at Line 8 during this execution of the **while** loop, we define $\beta(x) = \alpha(n)$.

Remark 5. Using these definitions of mappings α and β, one can verify that intermediary values of sets X and Act computed by the algorithm coincide with sets X_i and Act_i defined in Definition 5.

Example 7 (Example 6 continued). On Figure 3, numbers indicated on the left and on the right of nodes correspond to values of mappings α and β. When no number is written on the right, this means that the node is active, and then the value of β is $+\infty$.

Embedding of $\mathcal{C} = (X, x_0, B, \Lambda)$ in the reachability tree. In the labelled tree $\mathcal{C}$ built by the algorithm, the label of the new node n obtained from the pair (n', t) is computed by function Acc. To prove that $\mathcal{C}$ can be embedded in the reachability tree of $(\mathcal{N}, \varphi)$, we define a mapping called the concretization function which expresses the marking resulting from the acceleration as a marking reachable in $(\mathcal{N}, \varphi)$ from marking $\Lambda(n')$. Intuitively, an acceleration represents the repetition of some sequences of transitions until the upper bound is reached. As the system is finite (we consider widened Petri nets), we can exhibit a particular sequence of transitions which allows to reach this upper bound.

Definition 6 (Concretization function). *The concretization function is a mapping γ from B^* to T^*. Given a sequence of adjacent edges $b_1 \dots b_k \in B$, we define $\gamma(b_1 \dots b_k) = \gamma(b_1) \dots \gamma(b_k)$. We let $M = \max\{\varphi(p) \mid p \in P\} + 1$.*

Let $b = (n', t, n) \in B$. The definition of γ proceeds by induction on $\alpha(n')$: we assume γ is defined on all edges $(x, u, y) \in B$ such that $\alpha(x) < \alpha(n')$.

Let $m = \mathsf{Post}_\varphi(\Lambda(n'), t)$, then there are two cases, either :

1. *$\Lambda(n) = m$ (t is not accelerated), then we define $\gamma(b) = t$, or*
2. *$\Lambda(n) > m$. Let $X' = \{x_1, \dots, x_k\}$ (x_i's are ordered w.r.t. α) defined by: $X' = \{x \in \mathsf{Ancestor}_\mathcal{C}(n') \cap \mathsf{Act}_{\alpha(n)-1} \mid \Lambda(x) \le m \wedge \exists p. \Lambda(x)(p) < m(p) < \omega\}$. For each $j \in \{1, \dots, k\}$, let $w_j = \mathsf{path}_\mathcal{C}(x_i, n') \in B^*$. Then we define: $\gamma(b) = t(\gamma(w_1)t)^M \dots (\gamma(w_k)t)^M$.*

We prove in Appendix the following Lemma which states the expected property of the concretization function:

Lemma 2. *Let $x, y \in X$ such that $x \in \mathsf{Ancestor}_\mathcal{C}(y)$, and let $w = \mathsf{path}_\mathcal{C}(x, y)$. Then we have $\mathsf{Post}_\varphi(\Lambda(x), \gamma(w)) = \Lambda(y)$.*

This result allows to prove by induction that the labelled tree built by MP is, up to an isomorphism, included in the reachability tree of the WPN (see details in Appendix B.2), and is thus an exploration:

Proposition 2 (MP-exploration). *The execution of* MP *Algorithm on a* WPN $(\mathcal{N}, \varphi)$ *defines an exploration $\mathcal{E}$ of $(\mathcal{N}, \varphi)$. We call this exploration an* MP-exploration of $(\mathcal{N}, \varphi)$.

4.4 Main proof

Theorem 5. MP *Algorithm for* WPN *terminates and computes the* MCS.

Termination of MP for WPN can be proved as in Theorem 3. As a consequence of Lemma 2, MP algorithm only computes markings that are reachable in the WPN, therefore the algorithm is sound. We devote the rest of this section to the proof of its completeness.

Fix a WPN $(\mathcal{N}, \varphi)$, with $\mathcal{N} = (P, T, I, O, m_0)$, and let $\mathcal{E} = (X, B, \alpha, \beta)$ be an MP-exploration of $(\mathcal{N}, \varphi)$. We will use notations X, Act and Inact of Definition 5.

Preliminary properties. Given a node $n \in X$, we define the predicate $\mathsf{disc}(n)$ as $\beta(n) = \alpha(n)$. When this holds, we say that n is discarded as it is immediately added to the set Inact. In that case, no other node is deactivated.

Given two nodes $n, x \in X$ such that $\alpha(n) \le \alpha(x)$ and $n \in \mathsf{Inact}$, we define the predicate $\mathsf{prune}(n, x)$ as $\exists y \in \mathsf{Ancestor}_\mathcal{E}(n). \Lambda(y) \le \Lambda(x) \wedge (y \in \mathsf{Act}_{\beta(n)-1} \vee y \notin \mathsf{Ancestor}_\mathcal{E}(x))$.

One can check that the MP-exploration $\mathcal{E}$ satisfies the following properties. Arbitrary explorations do not satisfy them.

Proposition 3. *Let $n \in \mathsf{Inact}$, then:*

(i) $\mathsf{disc}(n) \iff \Lambda(n) \le \mathsf{Act}_{\alpha(n)-1}$.
(ii) $\neg\mathsf{disc}(n) \Rightarrow \mathsf{prune}(n, x)$, *where* $x = \alpha^{-1}(\beta(n))$.
(iii) $\forall x \in X$ *s.t.* $\alpha(n) \le \alpha(x)$, *if* $\mathsf{prune}(n, x) \wedge \neg\mathsf{disc}(x)$, *then* $\beta(n) \le \alpha(x)$

Covering Function. We introduce a function $\mathsf{Temp\text{-}Cover}$ which explicits why nodes are deactivated. Intuitively, for a node $n \in \mathsf{Inact}$, if we have $\mathsf{Temp\text{-}Cover}(n) = (x, \varrho) \in X \times T^*$, this means that node x is in charge of deactivation of n, and that the firing of the sequence ϱ from $\Lambda(x)$ leads to a state dominating $\Lambda(n)$. Note that to identify the sequence in T^*, we use the path between nodes *in the reachability tree*. This is possible as by definition, the exploration is embedded in the reachability tree.

Definition 7 (Temp-Cover). *The mapping* $\mathsf{Temp\text{-}Cover}$ *is defined from* Inact *to* $X \times T^*$ *as follows. Let* $n \in \mathsf{Inact}$*, and* $i = \beta(n)$*. We distinguish two cases:*

Discarded: *If* $\mathsf{disc}(n)$*, then by Proposition 3.(i), there exists a node* $x \in \mathsf{Act}_{i-1}$ *such that* $\Lambda(n) \leq \Lambda(x)$*, we define* [3] $\mathsf{Temp\text{-}Cover}(n) = (x, \varepsilon)$*.*

Not discarded: *Otherwise,* $\neg\mathsf{disc}(n)$ *holds. By Proposition 3.(ii),* $\mathsf{prune}(n, x)$ *holds, where* $x = \alpha^{-1}(i)$*. We fix* [4] *a witness* y *of property* $\mathsf{prune}(n, x)$*, and let* $\varrho = \mathsf{pathlabel}_{\mathcal{R}}(y, n) \in T^*$*. We define* $\mathsf{Temp\text{-}Cover}(n) = (x, \varrho)$*.*

The following property easily follows from Definition 7 and Lemma 1:

Lemma 3. *Let* $n \in \mathsf{Inact}$*,* $\mathsf{Temp\text{-}Cover}(n) = (x, \varrho)$*. Then* $\Lambda(n) \leq \mathsf{Post}_\varphi(\Lambda(x), \varrho)$*.*

The previous definition is temporary, in the sense that it describes how a node is deactivated. However, active nodes may be deactivated, and thus nodes referenced by mapping $\mathsf{Temp\text{-}Cover}$ may not belong to set Act. In order to recover an active node from which a dominating node can be obtained, we define a mapping which records for each inactivate node the successive covering informations:

Definition 8 (Covering function). *The covering function* Cover *is a mapping from* X *to sequences of pairs in* $\mathsf{Inact} \times T^*$*. It is recursively defined as follows. Let* $n \in X$*.*

1. *if* $n \in \mathsf{Act}$*, then* $\mathsf{Cover}(n) = \varepsilon$ *;*
2. *otherwise, let* $\mathsf{Temp\text{-}Cover}(n) = (x, \varrho)$*. We define* $\mathsf{Cover}(n) = (x, \varrho) \cdot \mathsf{Cover}(x)$*.*

Example 8 (Example 6 continued). We illustrate the definition of the covering function on Example 6. MP Algorithm terminates at step 10. Consider node n_3, deactivated at step 5. We have $\mathsf{Temp\text{-}Cover}(n_3) = (n_5, \varepsilon)$. Indeed, it is directly covered by node n_5. Node n_5 is deactivated at step 6 by node n_6 through node n_4, which is its ancestor by transition t_4, then we have $\mathsf{Temp\text{-}Cover}(n_5) = (n_6, t_4)$. Node n_6 is deactivated at step 8 because it is directly covered by node n_8, thus we have $\mathsf{Temp\text{-}Cover}(n_6) = (n_8, \varepsilon)$. We finally obtain $\mathsf{Cover}(n_3) = (n_5, \varepsilon) \cdot (n_6, t_4) \cdot (n_8, \varepsilon)$. One can verify that $\Lambda(n_3) \leq \mathsf{Post}_\varphi(\Lambda(n_8), t_4)$. ⌟

We state the next property which follows from Lemma 3 by induction:

Lemma 4. *Let* $n \in \mathsf{Inact}$ *be such that* $\mathsf{Cover}(n) = (x_1, \varrho_1) \cdots (x_k, \varrho_k)$*. Then* $\Lambda(n) \leq \mathsf{Post}_\varphi(\Lambda(x_k), \varrho_k \varrho_{k-1} \ldots \varrho_1)$*.*

[3] We choose any such node x.
[4] We could pick any such node y.

We now state a core property of mapping Cover, holding for MP-explorations. It is fundamental to prove the absence of cycles, and thus the fact that the exploration yields a minimal coverability set. Roughly, it states that intermediary markings skipped by accelerations would not modify activations/deactivations:

Proposition 4. *Let $x \in \mathsf{Inact}$ be such that $\mathsf{Cover}(x) = (x_1, \varrho_1) \cdots (x_k, \varrho_k)$. Define $\varrho = \varrho_k \varrho_{k-1} \ldots \varrho_1$, and let $n \in \mathsf{Act}$ and $\varrho' \in T^*$. Then we have:*

$$(\varrho' \prec \varrho \wedge \Lambda(n) \geq \mathsf{Post}_\varphi(\Lambda(x_k), \varrho')) \Rightarrow \beta(x) \leq \alpha(n)$$

Covering Path. Before turning to the proof of Theorem 5, we introduce an additional definition. Our aim is to prove that any reachable state s is covered by some active node. Therefore we define a notion of covering path, which is intuitively a path through active nodes in which each node is labelled with a sequence (a stack) of transitions that remain to be fired to reach a state s' dominating the desired state s. Formally, a covering path is defined as follows:

Definition 9 (Covering Path). *A* covering path *is a sequence $p = (n_i, \varrho_i)_{i\geq 1} \in (\mathsf{Act} \times T^*)^{\mathbb{N}}$ such that $\Lambda(n_1) \overset{\varrho_1}{\Rightarrow}_\varphi \cdot$ and for any $i \geq 1$, we have either*

(i) *$\varrho_i = \varepsilon$, and then it has no successor, or*

(ii) *$\varrho_i = t_i \varrho'_i$, then let n be such that $(n_i, t_i, n) \in B$ (possible as $\mathcal{E}$ is T-complete). If $n \in \mathsf{Act}$ then $(n_{i+1}, \varrho_{i+1}) = (n, \varrho'_i)$. Otherwise, let $\mathsf{Cover}(n) = (x_1, \eta_1) \cdots (x_k, \eta_k)$, we define $(n_{i+1}, \varrho_{i+1}) = (x_k, \eta_k \ldots \eta_1 \cdot \varrho'_i)$.*

Note that given a node $n \in \mathsf{Act}$ and $\varrho \in T^*$ such that $\Lambda(n) \overset{\varrho}{\Rightarrow}_\varphi \cdot$, there exists a unique covering path $p = (n_i, \varrho_i)_{i\geq 1}$ such that $(n_1, \varrho_1) = (n, \varrho)$. We say that this path is associated with the pair *(n, ϱ).*

Example 9 (Example 8 continued). We illustrate the definition of covering path on Example 6. Consider the covering path associated with the pair $(n_1, t_1 t_3 t_4)$. Successor of node n_1 by transition t_1 is the inactive node n_3. We have already shown in Example 8 that $\mathsf{Cover}(n_3) = (n_5, \varepsilon) \cdot (n_6, t_4) \cdot (n_8, \varepsilon)$. In addition, successor of node n_8 by transition t_4 is the active node n_9. Finally, one can verify that the covering path is: $(n_1, t_1 t_3 t_4), (n_8, t_4 t_3 t_4), (n_9, t_3 t_4), (n_8, t_4), (n_9, \varepsilon)$. Note that the marking $\{p_3, p_5\}$ reached from node n_1 by the sequence $t_1 t_3 t_4$ is covered by the marking $\{p_3, \omega p_5\}$ of node n_9. ⌟

Lemma 5. *Let $p = (n_i, \varrho_i)_{i\geq 1}$ be a covering path. Then we have $\mathsf{Post}_\varphi(\Lambda(n_1), \varrho_1) \leq \mathsf{Post}_\varphi(\Lambda(n_i), \varrho_i)$ for all i. In particular, if for some i we have $\varrho_i = \varepsilon$, we obtain $\mathsf{Post}_\varphi(\Lambda(n_1), \varrho_1) \leq \Lambda(n_i)$.*

Proof. We prove that for any i, we have $\mathsf{Post}_\varphi(\Lambda(n_i), \varrho_i) \leq \mathsf{Post}_\varphi(\Lambda(n_{i+1}), \varrho_{i+1})$. In the definition of covering path, we extend the path only in case (ii). Two cases can occur, in the first one, the property is trivial. In the second one, the property follows from Lemma 4. □

As a consequence of this lemma, to prove the completeness result, it is sufficient to show that for any reachable marking, there exists a finite covering path that covers it.

We introduce a notion of cycle for covering paths:

Definition 10. *Let* $(n,t) \in \mathsf{Act} \times T$ *such that* $\Lambda(n) \overset{t}{\Rightarrow}_\varphi \cdot$, *and* $p = (n_i, \varrho_i)_{i \geq 1}$ *be the covering path associated with* (n,t). *The pair* (n,t) *is said* singular *if there exists* $i > 1$ *such that* $(n_i, \varrho_i) = (n, t\varrho)$, *with* $\varrho \in T^*$.

Proof of Theorem 5. We will prove that any reachable marking of $(\mathcal{N}, \varphi)$ is covered by some active node. Let $m \in \mathsf{Mark}^\omega_\varphi(P)$ be a reachable marking. There exists $\varrho \in T^*$ such that $m_0 \overset{\varrho}{\Rightarrow}_\varphi m$. One can prove (see Appendix C.3) that there exists a node $n'_0 \in \mathsf{Act}$ such that $\Lambda(n_0) \leq \Lambda(n'_0)$ (n'_0 covers the root). As a consequence, there exists $m' \in \mathsf{Mark}^\omega_\varphi(P)$ such that $\Lambda(n'_0) \overset{\varrho}{\Rightarrow}_\varphi m'$ and $m \leq m'$. We can then consider the covering path associated with the pair (n'_0, ϱ).

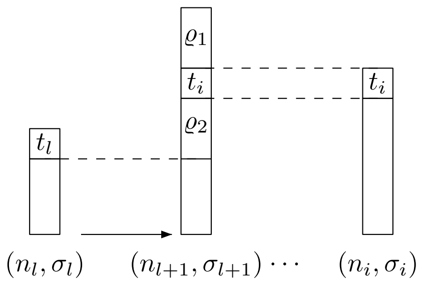


Fig. 5. Stacks of a singular pair.

We now prove that all covering paths are finite. This will conclude the proof by Lemma 5. One can prove (see Appendix C.4) that if a covering path is infinite, then it contains a singular pair.

We will prove that the MP-exploration $\mathcal{E}$ can not admit a singular pair. Consider a singular pair $(n,t) \in \mathsf{Act} \times T$, and denote by $p = (n_i, \sigma_i)_{i \geq 1}$ its infinite covering path. As it is singular, there exists $k > 1$ such that $(n_k, \sigma_k) = (n, t\sigma'_k)$. For any $1 \leq i \leq k$, we write $\sigma_i = t_i \sigma'_i$ (this is possible as the path is infinite and thus never contains empty stacks).

For each $1 < i \leq k$, we define the position $\mathsf{prod}(i) = \max\{1 \leq j < i \mid |\sigma_j| \leq |\sigma_i|\}$. This definition is correct as $|\sigma_1| = |t| = 1$, and for any $1 < i \leq k$, we have $|\sigma_i| \geq 1$ as $\sigma_i \neq \varepsilon$. Intuitively, the value $\mathsf{prod}(i)$ gives the position which is responsible of the addition in the stack of transition t_i. Indeed, let $1 < i \leq k$ and $l = \mathsf{prod}(i)$. As for any position j such that $l < j < i$, we have $|\sigma_j| > |\sigma_i|$, the transition t_i is present in σ_j "at the same height".

Consider now the position $1 < i \leq k$ such that $\alpha(n_i)$ is minimal among $\{\alpha(n_j) \mid 1 \leq j \leq k\}$ (recall that $n_1 = n_k$), and let $l = \mathsf{prod}(i)$. By the T-completeness of $\mathcal{E}$, there exists a node $x \in X$ such that edge (n_l, t_l, x) belongs to B. As we have $|\sigma_l| \leq |\sigma_{l+1}|$, this implies that $x \in \mathsf{Inact}$.

We write the covering function associated with node x as follows: $\mathsf{Cover}(x) = (x_1, \eta_1) \dots (x_k, \eta_k)$. Following the definition of a covering path, we obtain $x_k = n_{l+1}$. In addition, following the above mentionned property of $l = \mathsf{prod}(i)$, there exist two sequences $\varrho_1, \varrho_2 \in T^*$ such that $\eta_k \dots \eta_1 = \varrho_1 t_i \varrho_2$, and verifying:

$$\sigma_{l+1} = \varrho_1 t_i \varrho_2 \sigma'_l \text{ and } \sigma_i = t_i \varrho_2 \sigma'_l$$

This means that the head of the stack σ_l, *i.e.* transition t_l, has been replaced by the sequence $\varrho_1 t_i \varrho_2$, and that between positions $l+1$ and i, transition t_i (which is the head of the stack σ_i), is never consumed. This situation is depicted on Figure 5. In particular, this implies the following property:

$$\Lambda(n_i) \geq \mathsf{Post}_{\varphi}(\Lambda(n_{l+1}), \varrho_1)$$

Indeed, there are two cases, either $i = l+1$, and then we necessarily have $\varrho_1 = \varepsilon$ and the property is trivial, or $l+1 < i$, and then we have that the covering path starting in pair (n_{l+1}, ϱ_1) ends in pair (n_i, ε). The result then follows from Lemma 5.

To conclude, we use the key Proposition 4. Indeed, one can verify that the proposition can be applied on nodes x and n_i using sequences $\varrho = \varrho_1 t_i \varrho_2$ and $\varrho' = \varrho_1$. This result yields the following inequality: $\beta(x) \leq \alpha(n_i)$. As x is the successor of n_l by transition t_l, property (ii) of an exploration implies $\alpha(n_l) < \alpha(x)$. As we always have $\alpha(x) \leq \beta(x)$, we finally obtain $\alpha(n_l) < \alpha(n_i)$, which is a contradiction with our choice of i.

5 Comparison and Experiments

Experimental results are presented in Table 1. The K&M and MP algorithms were implemented in Python and tested on a 3 Ghz Xeon computer. The tests set is the one from [7]. We recall in the last column the values obtained for the CoverProc [7] algorithm. Note that the implementation of [7] also was in Python, and the tests were run on the same computer. We report for each test the number of places and transitions of the net and the size of its MCS, the time the K&M and MP algorithms took and the numbers of nodes each algorithm constructed.

As expected the MP algorithm is a lot faster than K&M algorithm and the tree it constructs is, in some instances, dramatically smaller. K&M algorithm could not compute the MCS for the last five tests (we time out after 20 minutes), while the MP algorithm took less than 20 seconds for all five tests. Note that the time reported for K&M algorithm is the time to build the K&M tree, from this tree one has to extract the minimal coverability set which maybe costly if the set is big (see K&M results in [7]). The MP algorithm directly computes the minimal coverability set (Act), *i.e.* no additional computation is needed.

CoverProc is significantly slower than MP. Moreover, MP algorithm has, in our view, another important advantage over CoverProc. In MP, the order of exploration is totally free, any exploration strategy yields a minimal coverability set while, in CoverProc, when an acceleration is performed the algorithm imposes to first treat the whole subtree of this accelerated node.

References

1. A. Finkel. A generalization of the procedure of Karp and Miller to well structured transition system. In *Proceedings of the 14th International Colloquium on Automata,*

Table 1. K&M and MP algorithm comparison. #P, #T, # MCS : number of places and transitions and size of the MCS of the Petri net. # nodes : number of nodes in the tree constructed by the algorithm.

Test				K&M		MP		CoverProc[7]
name	#P	#T	# MCS	# nodes	time (s)	# nodes	time (s)	time (s)
BasicME	5	4	3	5	< 0.01	5	< 0.01	0.12
Kanban	16	16	1	72226	9.1	114	< 0.01	0.19
Lamport	11	9	14	83	0.02	24	< 0.01	0.17
Manufacturing	13	6	1	81	0.01	30	< 0.01	0.14
Peterson	14	12	20	609	0.2	35	0.02	0.25
Read-write	13	9	41	11139	6.33	76	.06	1.75
Mesh2x2	32	32	256	x	x	6241	18.1	330
Multipool	18	21	220	x	x	2004	4.9	365
pncsacover	31	36	80	x	x	1604	1.6	113
csm	14	13	16	x	x	102	.03	0.34
fms	22	20	24	x	x	809	0.28	2.1

Languages and Programming (ICALP'87), volume 267 of *Lecture Notes in Computer Science*, pages 499–508. Springer-Verlag, 1987.

2. A. Finkel. The minimal coverability graph for Petri nets. In *Papers from the 12th International Conference on Applications and Theory of Petri Nets (APN'91)*, volume 674 of *LNCS*, pages 210–243. Springer-Verlag, 1993.
3. A. Finkel and J. Goubault-Larrecq. Forward analysis for wsts, part i: Completions. In *Proc. 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*, volume 3 of *LIPIcs*, pages 433–444, 2009.
4. A. Finkel and J. Goubault-Larrecq. Forward analysis for wsts, part ii: Complete wsts. In *Proc. 36th Internatilonal Colloquium on Automata, Languages and Programming (ICALP 2009)*, volume 5556 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2009.
5. A. Finkel, J.-F. Raskin, M. Samuelides, and L. V. Begin. Monotonic extensions of petri nets: Forward and backward search revisited. *Electr. Notes Theor. Comput. Sci.*, 68(6), 2002.
6. G. Geeraerts. *Coverability and Expressiveness Properties of Well-structured Transitions Systems.* Thèse de doctorat, Université Libre de Bruxelles, Belgique, June 2007.
7. G. Geeraerts, J.-F. Raskin, and L. Van Begin. On the efficient computation of the coverability set for petri nets. *International Journal of Foundations of Computer Science*, 21(2):135–165, 2010.
8. R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.
9. K. Luttge. Zustandsgraphen von petri-netzen. Master's thesis, Humboldt-Universitat, 1995.

A MP Algorithm for WPN

Algorithm 3 Monotone Pruning Algorithm for Widened Petri Nets.

```
Require: A widened Petri net $(\mathcal{N},\varphi)$ with $\mathcal{N}=(P,T,I,O,m_0)$.
Ensure: A labelled tree $\mathcal{C}=(X,x_0,B,\Lambda)$ and a partition $X=\mathsf{Act}\uplus\mathsf{Inact}$
     such that $\Lambda(\mathsf{Act})=\mathrm{MCS}(\mathcal{N},\varphi)$.
 1: Let $x_0$ be a new node such that $\Lambda(x_0)=m_0$;
 2: $X:=\{x_0\}$; $\mathsf{Act}:=X$; $\mathsf{Wait}:=\{(x_0,t)\mid\Lambda(x_0)\overset{t}{\Rightarrow}_\varphi\cdot\}$; $B:=\emptyset$;
 3: while $\mathsf{Wait}\neq\emptyset$ do
 4:   Pop $(n',t)$ from $\mathsf{Wait}$.
 5:   if $n'\in\mathsf{Act}$ then
 6:     $m:=\mathsf{Post}_\varphi(\Lambda(n'),t)$;
 7:     Let $n$ be a new node such that $\Lambda(n)=\mathsf{Acc}(\Lambda(\mathsf{Ancestor}_\mathcal{C}(n')\cap\mathsf{Act}),m)$;
 8:     $X+=\{n\}$; $B+=\{(n',t,n)\}$;
 9:     if $\Lambda(n)\not\leq\Lambda(\mathsf{Act})$ then
10:       $\mathsf{Act}-=\{x\mid\exists y{\in}\mathsf{Ancestor}_\mathcal{C}(x).\Lambda(y){\leq}\Lambda(n)\wedge(y{\in}\mathsf{Act}\vee y\notin\mathsf{Ancestor}_\mathcal{C}(n))\}$;
11:       $\mathsf{Act}+=\{n\}$; $\mathsf{Wait}+=\{(n,u)\mid n\overset{u}{\Rightarrow}_\varphi\cdot\}$;
12:     end if
13:   end if
14: end while
15: Return $\mathcal{C}=(X,x_0,B,\Lambda)$ and $(\mathsf{Act},\mathsf{Inact})$.
```

B Properties of the concretization function

In this section, as the WPN $(\mathcal{N},\varphi)$ is fixed, we omit the subscript φ in the operator Post.

B.1 Proof of Lemma 2

Lemma 2. *Let $x,y\in X$ such that $x\in\mathsf{Ancestor}_\mathcal{C}(y)$, and let $w=\mathsf{path}_\mathcal{C}(x,y)$. Then we have $\mathsf{Post}_\varphi(\Lambda(x),\gamma(w))=\Lambda(y)$.*

Proof. We prove the result for the case where w is a single edge, that is $w=(x,t,n)\in B$. The general result follows easily. We let $m=\mathsf{Post}(\Lambda(x),t)$.

We distinguish two cases :

If $\Lambda(n)=m$: then by definition we have $\gamma(w)=t$ and the result is trivial.

If $\Lambda(n)>m$: then an acceleration has been applied. We prove the property by induction on $\alpha(x)$.

- if $\alpha(x)=1$, then the acceleration is necessarily applied w.r.t. node x, that is $\mathsf{Post}(\Lambda(x),t)>\Lambda(x)$. Let $add\in\mathsf{Mark}^\omega_\varphi(P)$ be defined by $add(p)=\mathsf{Post}(\Lambda(x),t)(p)-\Lambda(x)(p)$ for any $p\in P$. Naturally, the vector is positive exactly for places p which have strictly increased, and which will thus be accelerated. For these places, after M iterations of t, the value in these places has exceeded the maximum value, *i.e.* value $\varphi(p)$, and thus is equal to ω. In the other places, the value is let unchanged. As a concequence, we exactly obtain $\mathsf{Post}(\Lambda(x), t^M) = \Lambda(n)$.

– otherwise, we have $\alpha(x) > 1$. Following the definition of the concretization function, let $x_1, \ldots, x_k$ denote the nodes used to compute the concretized path $\gamma(t)$, and $w_1, \ldots, w_k$ the paths associated. By induction hypothesis, we have $\mathsf{Post}(\Lambda(x_i), \gamma(w_i)) = \Lambda(x)$ for any $i = 1 \ldots k$. By definition, we have $\gamma(t) = t.(\gamma(w_1).t)^M \ldots (\gamma(w_k).t)^M$. For $i \in \{0, \ldots, k\}$, let m_i^{acc} denote the marking reached from $\Lambda(x)$ by the sequence $t.(\gamma(w_1).t)^M \ldots (\gamma(w_i).t)^M$, *i.e.* such that $m_i^{acc} = \mathsf{Post}(\Lambda(x), t.(\gamma(w_1).t)^M \ldots (\gamma(w_i).t)^M)$. We prove, by induction on $i \in \{0, \ldots, k\}$, the following property:

$$\forall p \in P, m_i^{acc}(p) = \begin{cases} \omega & \text{if } \exists 1 \leq j \leq i \text{ s.t. } \Lambda(x_j)(p) < m(p) < \omega \\ m(p) & \text{otherwise.} \end{cases}$$

- For $i = 0$, the property is trivial by definition of m.
- Let $i < k$, assume property holds for i and let prove it for $i+1$. To prove the result we split the set of places P into three parts and successively prove that for each case the property is satisfied:
  - *(i)* P_1: $\exists 1 \leq j \leq i \mid \Lambda(x_j)(p) < m(p) < \omega$. Intuitively, P_1 represents places accelerated by one of the nodes $x_1, \ldots, x_i$. By the induction hypothesis, we have $m_i^{acc}(p) = \omega$, and we thus we will still have $m_{i+1}^{acc}(p) = \omega$, as expected.
  - *(ii)* P_2: $p \notin P_1 \wedge \Lambda(x_{i+1})(p) < m(p) < \omega$. Intuitively, P_2 denotes places not accelerated by one of $x_1, \ldots, x_i$, but that should be accelerated by x_{i+1}. By induction hypothesis, we have $m_i^{acc}(p) = m(p)$, and we have to prove that $m_{i+1}^{acc}(p) = \omega$. By the induction hypothesis of the external induction, we have $\Lambda(x) = \mathsf{Post}(\Lambda(x_{i+1}), \gamma(w_{i+1}))$. Thus we obtain $m = \mathsf{Post}(\Lambda(x_{i+1}), \gamma(w_{i+1}).t)$. Let $p \in P_2$. Following properties stated above, we have $\Lambda(x_{i+1})(p) < m(p) = m_i^{acc}(p)$. Thus, the iteration of the sequence $\gamma(w_{i+1}).t$ will increase the value of place p. By the choice of M, the value $\varphi(p)$ will be exceeded and we obtain $m_{i+1}^{acc}(p) = \omega$ as expected.
  - *(iii)* P_3: $p \notin P_1 \wedge p \notin P_2$. This last case concerns places that should not be accelerated by any of the x_j's, with $j \leq i+1$. Thus induction hypothesis entails that $m_i^{acc}(p) = m(p)$ and we have to prove that $m_{i+1}^{acc}(p) = m(p)$. As in the previous case, we have $m = \mathsf{Post}(\Lambda(x_{i+1}), \gamma(w_{i+1}).t)$. Let $p \in P_2$, then we have $\Lambda(x_{i+1})(p) = m(p) = m_i^{acc}(p)$. Here, the iteration of the sequence $\gamma(w_{i+1}).t$ will let the value of place p unchanged. We thus obtain $m_{i+1}^{acc}(p) = m(p)$ as expected.

  It is then trivial to verify that the application of this property for $i = k$ leads to the result.

This concludes the proof. □

B.2 Details on the embedding of $\mathcal{C}$ in $\mathcal{R}$

To embed $\mathcal{C}$ in $\mathcal{R}$, we define the mapping η from X to N which maps a node $x \in X$ to a node $n \in N$ that is labelled with the same marking. η is recursively defined by:

- $\eta(x_0) = n_0$,
- let $b = (x, t, y) \in B$, $n = \eta(x)$, and $\varrho = \gamma(b)$. By Lemma 2, we have $\Lambda(n) \overset{\varrho}{\Rightarrow} \cdot$. Thus there exists a unique node $n' \in N$ such that $n \in \mathsf{Ancestor}_{\mathcal{R}}(n')$ and $\mathsf{pathlabel}_{\mathcal{R}}(n, n') = \varrho$. We define $\eta(y) = n'$.

One can verify that using this definition, the $\mathcal{C}$-labelling of a node $x \in X$ coincides with the $\mathcal{R}$-labelling of the node $\eta(x) \in N$. As a consequence, we identify in the sequel node $x \in X$ with node $\eta(x) \in N$.

To conclude, we check properties (i) to (v):

(i) the property is trivial,

(ii) first, note that $(x, t, y) \in B$ entails $\alpha(x) < \alpha(y)$. Second, the definition of η implies that $\forall x, y \in X, x \in \mathsf{Ancestor}_{\mathcal{R}}(y) \Rightarrow x \in \mathsf{Ancestor}_{\mathcal{C}}(y)$. This proves the property.

(iii) This is a consequence of test of Line 5.

(iv) This is a consequence of the termination of the MP Algorithm. Indeed, when the algorithm terminates, the set Wait is empty, what implies that all possible successors of active nodes have been built.

(v) Let $b = (x, t, y) \in B$. By Lemma 2 applied on b, one can verify that the sequence $\gamma(t) \in T^*$ starts with transition t and thus the successor in the reachability tree of node x by the transition t satisfies all constraints.

B.3 Minimal Completeness

We prove an additional property of MP Algorithm related to its accelerations. Intuitively, some nodes may be hidden when an acceleration is performed. The property states that if a node y is hidden, then it owns an ancestor y' which has been explored ($y' \in X$), and whose label is strictly less than the label of y. We call this property the minimal completeness of the exploration.

Lemma 6 (Minimal Completeness). *For any edge $b = (x, t, x') \in B$ corresponding to an acceleration (*i.e. *such that $\Lambda(x') > \mathsf{Post}_{\varphi}(\Lambda(x), t)$), there exists a node z such that:*

(i) *$z \in \mathsf{Ancestor}_{\mathcal{E}}(x)$ and $\beta(z) = \alpha(x')$,*

(ii) *for any node $y \in \mathsf{Ancestor}_{\mathcal{E}}(x') \setminus \mathsf{Ancestor}_{\mathcal{E}}(x)$, there exists a node $y' \in \mathsf{Ancestor}_{\mathcal{R}}(y) \cap X$ such that $\Lambda(y') < \Lambda(y)$ and $z \in \mathsf{Ancestor}_{\mathcal{E}}(y')$.*

Proof. The proof proceeds by induction on $\alpha(x)$. If $\alpha(x) = 1$ (x is the root), then one can easily verify that one can choose $z = x$, and for any node y that is skipped by the acceleration, $y' = x$ is a correct candidate.

We now consider x such that $\alpha(x) > 1$, and consider an edge $b = (x, t, x') \in B$. We consider the notations introduced in the definition of the concretization function, and let $\gamma(b) = t(\gamma(w_1)t)^M \dots (\gamma(w_k)t)^M$, where w_i is the path in B^* associated with node x_i. We assume that nodes x_i's are ordered w.r.t. α, and thus x_1 is an ancestor of all x_i's. We let $z = x_1$, it verifies property (i).

Let us prove property (ii). Let $y \in N$ be a node of the reachabilty tree such that there exists a word $\varepsilon \neq \varrho \preceq \gamma(b)$ verifying $\Lambda(y) = \mathsf{Post}(\Lambda(x), \varrho)$. As $\varrho \preceq \gamma(b)$, there exists a unique i such that $t(\gamma(w_1)t)^M \dots (\gamma(w_i)t)^M \prec \varrho$ and $\varrho \preceq t(\gamma(w_1)t)^M \dots (\gamma(w_{i+1})t)^M$.

We can thus decompose ϱ as $\varrho = t(\gamma(w_1)t)^M \dots (\gamma(w_i)t)^M (\gamma(w_{i+1})t)^l \eta$ where $0 \leq l < M$ and $\varepsilon \prec \eta \preceq \gamma(w_{i+1})t$.

Consider the node y' in the reachability tree defined as follows: it is the successor of node x_{i+1} by the sequence η.

We first prove that $\Lambda(y') \leq \Lambda(y)$. Following notations introduced in the proof of Lemma 2, we have that $\Lambda(y)$ can be reached from marking m_i^{acc} by the sequence $(\gamma(w_{i+1})t)^l \eta$. According to previous properties, we obtain $m \leq m_i^{acc} \leq \mathsf{Post}(m_i^{acc}, (\gamma(w_{i+1})t)^l)$ and thus $\Lambda(y') = \mathsf{Post}(\Lambda(x_{i+1}), \eta) \leq \mathsf{Post}(m, \eta) \leq \Lambda(y)$.

Now, we prove that the inequality is strict. By contradiction, if $\Lambda(y') = \Lambda(y)$, according to previous inequalities, we obtain $\mathsf{Post}(\Lambda(x_{i+1}), \eta) = \mathsf{Post}(m, \eta)$. By completing η to obtain the sequence $\gamma(w_{i+1})t$, we obtain $\mathsf{Post}(\Lambda(x_{i+1}), \gamma(w_{i+1})t) = \mathsf{Post}(m, \gamma(w_{i+1})t)$. By Lemma 2, we have $\mathsf{Post}(\Lambda(x_{i+1}), \gamma(w_{i+1})) = \Lambda(x)$. By definition of m, we obtain $m = \mathsf{Post}(m, \gamma(w_{i+1})t)$. This is a contradiction with our choice of x_{i+1}! Indeed, in Definition 6, we require the following property: $\exists p. \Lambda(x_{i+1})(p) < m(p) < \omega$. One can prove that this implies the following strict inequality: $m(p) < \mathsf{Post}(m, \gamma(w_{i+1})t)(p)$, yielding the contradiction

Then, we distinguish two cases:

- if $y' \in X$, then we are done ($z = x_1$ is an ancestor y').
- otherwise ($y' \notin X$), this implies that y' is skipped by an acceleration on the path between x_{i+1} and x, related to an edge $b' = (n, u, n')$. But then we can apply the induction hypothesis on this edge as we have $\alpha(n) < \alpha(x)$, and obtain two nodes z' and y'' verifying properties (i) and (ii). By transitivity, we trivially obtain $y'' \in \mathsf{Ancestor}_{\mathcal{R}}(y) \cap X$ and $\Lambda(y'') < \Lambda(y)$. It remains to prove that $z \in \mathsf{Ancestor}_{\mathcal{E}}(y'')$. As x_{i+1} is an ancestor of node n, active at step $\alpha(x)$, it can not be deactivated by the acceleration related to edge b', what implies that z' must be "below" x_{i+1}, *i.e.* $x_{i+1} \in \mathsf{Ancestor}_{\mathcal{E}}(z')$. This yields the result.

This concludes the proof. □

C Complements on Subsection 4.4

C.1 Additional preliminary properties

Proposition 5. *Let $\mathcal{E}$ an* MP*-exploration, and $i \in \{1, \dots, |X|\}$. Let three distinct nodes $x, y, n \in X$ such that $n \in \mathsf{Act}$, $\Lambda(y) \leq \Lambda(n)$, $y \in \mathsf{Ancestor}_{\mathcal{E}}(x)$ and $y \notin \mathsf{Ancestor}_{\mathcal{E}}(n)$. Then we have $\beta(x) \leq \alpha(n)$.*

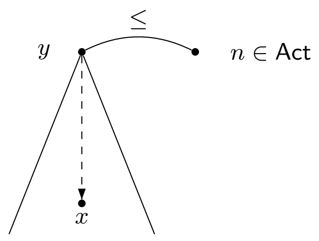


Fig. 6. Illustration of Proposition 5.

Proof. The property which is illustrated on Figure 6. First note that $\alpha(n) < \alpha(y)$ cannot hold. Indeed, if it was the case, then n would stop y as $n \in \mathsf{Act}$ and $\Lambda(y) \leq \Lambda(n)$. This is impossible as y is a strict ancestor of x. Thus, we have $\alpha(n) > \alpha(y)$ (the equality is impossible as y and n are distinct). Then, one can verify that the introduction of n will deactivate node x, if it has not yet been deactivated: node y is an ancestor of x, covered by n, and y is not an ancestor of n. We obtain $\beta(x) \leq \alpha(n)$. □

Lemma 7. *Let $n \in \mathsf{Inact}$, $\mathsf{Temp\text{-}Cover}(n) = (x, \varrho)$ and c such that $\mathsf{pathlabel}_{\mathcal{R}}(c, n) = \varrho$. Then we have:*

(i) $\beta(n) < \beta(x)$;
(ii) $\varrho \neq \varepsilon \Rightarrow \beta(n) = \alpha(x)$;
(iii) *if* $\varrho \neq \varepsilon$, *then* $\forall y \in X, c \in \mathsf{Ancestor}_{\mathcal{E}}(y) \wedge x \notin \mathsf{Ancestor}_{\mathcal{E}}(y) \Rightarrow \beta(y) \leq \alpha(x)$.

Proof. Property (i) is a consequence of the following property: we have $x \in \mathsf{Act}_{\beta(n)}$. Intuitively, this means that x is active when it deactivates node n.

Property (ii): by definition of $\mathsf{Temp\text{-}Cover}(n)$, $\varrho \neq \varepsilon$ implies that property $\neg\mathsf{disc}(n)$ holds. Then we obtain $x = \alpha^{-1}(\beta(n))$, as expected.

Last, consider property (iii). As for the previous property, by definition of $\mathsf{Temp\text{-}Cover}(n)$, $\varrho \neq \varepsilon$ implies that $\mathsf{prune}(n, x)$ holds and that c is a witness of the property $\mathsf{prune}(n, x)$. Then, by definition of the pruning of MP Algorithm, the whole subtree rooted in c is deactivated by node x, except node x itself if it belongs to the subtree. Let $y \in X$ such that $c \in \mathsf{Ancestor}_{\mathcal{E}}(y) \wedge x \notin \mathsf{Ancestor}_{\mathcal{E}}(y)$. Then y belongs to the subtree rooted in c, but not to the subtree rooted in x. As a consequence, it is deactivated by node x, and we obtain $\beta(y) \leq \alpha(x)$. □

C.2 Proof of Proposition 4

Proposition 4. *Let $x \in \mathsf{Inact}$ be such that $\mathsf{Cover}(x) = (x_1, \varrho_1) \cdots (x_k, \varrho_k)$. Define $\varrho = \varrho_k \varrho_{k-1} \ldots \varrho_1$, and let $n \in \mathsf{Act}$ and $\varrho' \in T^*$. Then we have:*

$$(\varrho' \prec \varrho \wedge n \geq \mathsf{Post}(x_k, \varrho')) \Rightarrow \beta(x) \leq \alpha(n)$$

Proof. As $\varrho' \prec \varrho$, there exists a unique index j such that $1 \leq j \leq k$ and $\varrho' = \varrho_k \varrho_{k-1} \dots \varrho_{j+1} \varrho''$ with $\varepsilon \preceq \varrho'' \prec \varrho_j$. In particular, this yields that $\varrho_j \neq \varepsilon$.

By definition of Cover, we have that for any $\ell \in \{1, \dots, k\}$, $(x_\ell, \varrho_\ell) = \mathsf{Temp\text{-}Cover}(x_{\ell-1})$ (where we let $x_0 = x$). By Lemma 7. (i), this implies $\beta(x_{\ell-1}) < \beta(x_\ell)$. We thus obtain the inequality $\beta(x) \leq \beta(x_{j-1})$, as $x_0 = x$ (the inequality is non strict as we may have $j = 1$).

Consider the peculiar case $x_j \in \mathsf{Ancestor}_{\mathcal{E}}(n)$. This implies $\alpha(x_j) \leq \alpha(n)$. As $\varrho_j \neq \varepsilon$, we have by Lemma 7. (ii) the equality $\beta(x_{j-1}) = \alpha(x_j)$, which yields the result. In the sequel we thus assume $x_j \notin \mathsf{Ancestor}_{\mathcal{E}}(n)$.

Node x_{j-1} is deactivated by node x_j, and with sequence ϱ_j. This means the ancestor of node x_{j-1} by the sequence ϱ_j in the reachability tree, which we denote by c, belongs to X and is covered by x_j. As $\varrho'' \prec \varrho_j$, we can consider the successor of c by the sequence ϱ'' (in the reachability tree), and denote this node by y, which is thus a (strict) ancestor of $x_{j-1} \in X$. We now distinguish two cases: either $y \in X$ or $y \notin X$. Let detail more precisely the second case: node y lies inbetween nodes c and x_{j-1} and as it does not belong to X, it is "skipped" by an acceleration. We denote by $(y_1, t, y_2) \in B$ the edge of the exploration that skipps the node y. By the minimal-completeness property of the exploration $\mathcal{E}$ applied on edge (y_1, t, y_2) and node y, there exist two nodes z and y' in X verifying the following properties:

(i) $z \in \mathsf{Ancestor}_{\mathcal{E}}(y_1)$ and $\beta(z) = \alpha(y_2)$,
(ii) $y' \in \mathsf{Ancestor}_{\mathcal{R}}(y) \cap X$, $\Lambda(y') < \Lambda(y)$ and $z \in \mathsf{Ancestor}_{\mathcal{E}}(y')$.

We will prove that in the first case ($y \in X$), and respectively in the second one ($y \notin X$), we can apply Proposition 5 to nodes x_{j-1}, y and n (resp. x_{j-1}, y' and n in the second case). Therefore, we prove each of the hypotheses:

- First, properties $n \in \mathsf{Act}$ and $y \in \mathsf{Ancestor}_{\mathcal{E}}(x_{j-1})$ (resp. $y' \in \mathsf{Ancestor}_{\mathcal{E}}(x_{j-1})$) are trivial. In addition, we obviously have $x_{j-1} \neq y$ (resp. $x_{j-1} \neq y'$) as y is a strict ancestor of x_{j-1} (and y' is itself a strict ancestor of y). We also have $x_{j-1} \neq n$ as $n \in \mathsf{Act}$ while x_{j-1} is deactivated by x_j.
- Second, we prove that $\Lambda(n) \geq \Lambda(y)$ (resp. $\Lambda(n) \geq \Lambda(y')$ in the second case). Indeed, we can prove using Lemma 3 that $\mathsf{Post}(\Lambda(x_k), \varrho_k \varrho_{k-1} \dots \varrho_{j+1}) \geq \Lambda(x_j)$. By definition of c, we have $\Lambda(x_j) \geq \Lambda(c)$. By definition of y, this yields $\mathsf{Post}(\Lambda(x_k), \varrho) \geq \Lambda(y)$, and thus $\Lambda(n) \geq \Lambda(y)$. In the second case, the property follows from $\Lambda(y) > \Lambda(y')$.
- Third, we prove that $y \notin \mathsf{Ancestor}_{\mathcal{E}}(n)$ (resp. $y' \notin \mathsf{Ancestor}_{\mathcal{E}}(n)$), which also entails $y \neq n$ (resp. $y' \neq n$). Consider the first case and proceed by contradiction: assume that y is an ancestor of node n. This implies that $c \in \mathsf{Ancestor}_{\mathcal{E}}(n)$, and then by Lemma 7.$(iii)$, as $x_j \notin \mathsf{Ancestor}_{\mathcal{E}}(n)$ and $\varrho_j \neq \varepsilon$, we obtain $\beta(n) \leq \alpha(x_j)$ which is impossible as $n \in \mathsf{Act}$.
  Consider now the second case and proceed by contradiction: assume that $y' \in \mathsf{Ancestor}_{\mathcal{E}}(n)$. Then y_2 is necessarily an ancestor of n, otherwise n is deactivated at step $\alpha(y_2)$ (see Lemma 7.(iii)). But then we can apply a reasoning similar to that of the first case and prove that n is deactivated by node x_j, what yields a contradiction.

Finally, we obtain by Proposition 5 the inequality $\beta(x_{j-1}) \leq \alpha(n)$. Combined with a previous inequality, this entails $\beta(x) \leq \alpha(n)$ as expected. □

C.3 An additional Lemma on the root

We now prove a Lemma stating that the root of the reachability tree is always covered by an active node, which in addition is the root of the current tree (restricted to the active nodes).

Lemma 8. *Let $\mathcal{E}$ be an* MP*-exploration with root n_0. Then there exists a node $n'_0 \in \mathsf{Act}$ such that $\Lambda(n_0) \leq \Lambda(n'_0)$.*

Proof. We prove that the property holds for the set Act_i all $i \leq |X|$. For all i such that $n_0 \in \mathsf{Act}_i$, there is nothing to prove. Consider, if it exists, the smallest i such that $n_0 \in \mathsf{Act}_i$ and $n_0 \notin \mathsf{Act}_{i+1}$. Let $n = \alpha^{-1}(i+1)$. Following definition of Act_{i+1}, as n_0 is the root of the tree, we must have $\Lambda(n_0) < \Lambda(n)$. As i has been chosen to be minimal, n can not be covered by another node, and thus $n \in \mathsf{Act}_{i+1}$. As a consequence, the property is true at step $i+1$. Moreover, note that as n_0 is the root of the tree, n is necessarily a descendant of n_0. As a consequence, the definition of active and inactive nodes yields that the only remaining active node after step $i+1$ is the node n. We thus have $\mathsf{Act}_{i+1} = \{n\}$. In other terms, this means that from this step, the exploration will start from a new root, and thus by the all new (active or node) nodes are descendant of n. Then, we can inductively apply the same reasoning to node n, and conclude by the transitivity of relation $\leq$. □

C.4 An additional Lemma on singular pairs

Lemma 9. *Let $(n, \varrho) \in \mathsf{Act} \times T^*$ such that $\Lambda(n) \overset{\varrho}{\Rightarrow} \cdot$, and $p = (n_i, \varrho_i)_{i \geq 1}$ be the coverability path s.t. $(n_1, t_1) = (n, \varrho)$. If p is infinite, then there exists a position $i \geq 1$ such that the pair (n_i, t_i) is singular, where $\varrho_i = t_i \varrho'_i$.*

Proof. We distinguish two cases:

- If there exists a bound $k \in \mathbb{N}^*$ such that infinitely often, the length of the sequence ϱ_i is smaller than k. Then, as the number of sequences of length bounded by k and the number of active nodes are finite, there exist two positions $1 \leq j < l$ such that $(n_j, \varrho_j) = (n_l, \varrho_l)$. Let i be an index in the interval $[j, l]$ such that the length of the sequence ϱ_i is minimal. This implies that the construction of the coverability path from (n_i, ϱ_i) only depends on the first transition t_i of ϱ_i: the pair (n_i, t_i) is singular.
- Otherwise, for any bound k, there exists a position after which all sequences of transitions have a length larger than k. For each k, we note $l(k)$ the first position verifying this property: $\forall l \geq l(k), |\varrho_l| \geq k$. Note that the sequence starting at $l(k)$ only depends on the node $n_{l(k)}$ and the transition $t_{l(k)}$ such that $\varrho_{l(k)} = t_{l(k)} \varrho'_{l(k)}$. As the number of active nodes and the set of transitions are finite, there exist $k < k'$ such that $l(k) < l(k')$, $n_{l(k)} = n_{l(k')}$ and $t_{l(k)} = t_{l(k)}$. Then the pair $(n_{l(k)}, t_{l(k)})$ is singular. □

D Running MP Algorithm on the counter-example of [7]

The Petri net considered in [7] to prove the incompleteness of the MCT Algorithm is depicted on Figure 7. We represent on Figure 8 an execution of the MP Algorithm on it.

The difference with the execution of the MCT Algorithm occurs at step 8: node n_7 is deactivated because node n_6 is covered by node n_8. This deactivation does not happen in MCT because node n_6 is inactive. This exactly corresponds to the difference between the two algorithm.

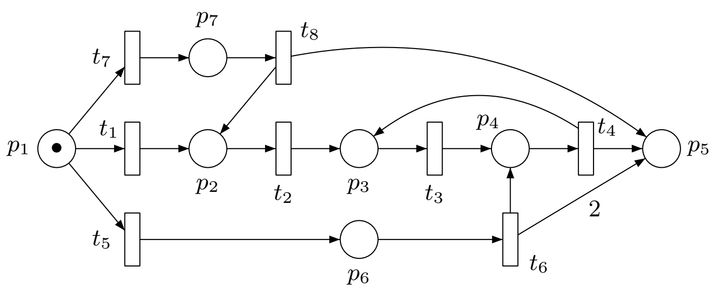


Fig. 7. The counter-example of [7]: Petri net $\mathcal{N}_{cex}$.

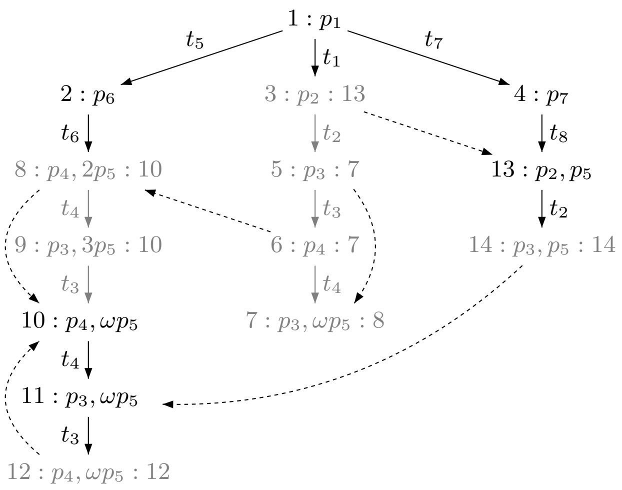


Fig. 8. An execution of MP Algorithm on $\mathcal{N}_{cex}$.