Fundamenta Informaticae 88 (2008) 1–27
IOS Press

From Many Places to Few: Automatic Abstraction Refinement for Petri Nets*

Pierre Ganty†, **Jean-François Raskin**, **Laurent Van Begin**‡
Département d'Informatique
Université Libre de Bruxelles (U.L.B.), Belgium
{pganty,jraskin,lvbegin}@ulb.ac.be

Abstract. Current algorithms for the automatic verification of Petri nets suffer from the explosion caused by the high dimensionality of the state spaces of practical examples. In this paper, we develop an abstract interpretation based analysis that reduces the dimensionality of state spaces that are explored during verification. In our approach, the dimensionality is reduced by trying to gather places that may not be important for the property to establish. If the abstraction that is obtained is too coarse, an automatic refinement is performed and a more precise abstraction is obtained. The refinement is computed by taking into account information about the inconclusive analysis. The process is iterated until the property is proved to be true or false.

1. Introduction

Petri nets (and their monotonic extensions) are well-adapted tools for modeling concurrent and infinite state systems like, for instance, parametrized systems [20]. Even though their state space is infinite, several interesting problems are decidable on Petri nets. The seminal work of Karp and Miller [22] shows that, for Petri nets, an effective representation of the downward closure of the set of reachable markings, the so-called *coverability set*, is constructable. This coverability set is the main tool needed to decide several interesting problems and in particular the *coverability problem*. The coverability problem asks: "given a Petri net N, an initial marking m_0 and a marking m, is there a marking m' reachable from m_0 which is greater or equal to m". The coverability problem was shown decidable in the nineties for the larger class of *well-structured transition systems* [14, 1]. That class of transition systems includes a large number of interesting infinite state models including Petri nets and their monotonic extensions.

A large number of works have been devoted to the study of efficient techniques for the automatic verification of coverability properties of infinite state Petri nets, see for example [10, 26, 2, 21]. Forward and backward algorithms are now available and have been implemented to show their practical relevance. All those methods manipulate, somehow, infinite sets of markings. Sets of markings are subsets of

*This research was supported the Belgian FNRS grant 2.4530.02 of the FRFC project "Centre Fédéré en Vérification" and by the project "MoVES", an Interuniversity Attraction Poles Programme of the Belgian Federal Government.

†Address for correspondence: Département d'Informatique, Université Libre de Bruxelles (U.L.B.)

‡Laurent Van Begin is "Chargé de recherche" at FNRS, Belgium.

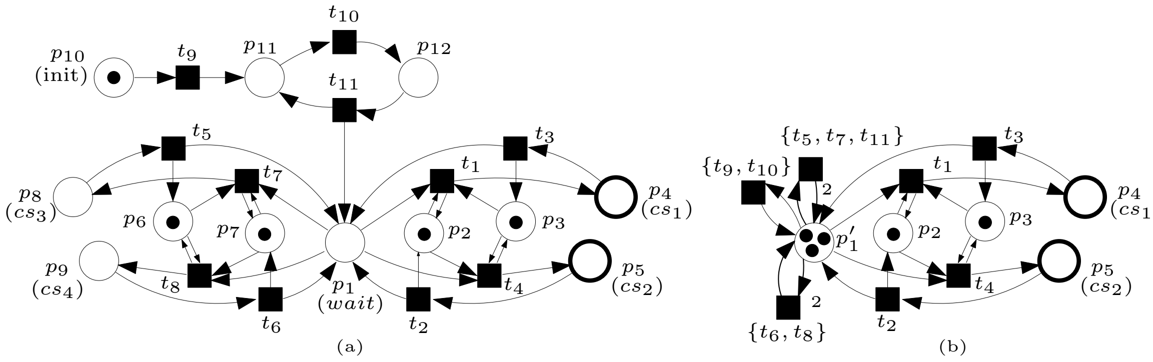

Figure 1. A Petri net with two distinct mutual exclusion properties (a) and its abstraction (b).

$\mathbb{N}^k$ where $\mathbb{N}$ is the set of positive integers and k is the number of places in the Petri net. We call k its *dimension*. When k becomes large the above mentioned methods suffer from the *dimensionality problem*: the sets that have to be handled have large representations that make them hard to manipulate efficiently.

In this paper, we develop an automatic abstraction technique that attacks the dimensionality problem. To illustrate our method, let us consider the Petri net of Fig. 1(a). This Petri net describes abstractly a system that spawns an arbitrary number of processes running in parallel. There are two independent critical sections in the system that correspond to places p_4, p_5 and to places p_8, p_9. One may be interested in proving that mutual exclusion is ensured between p_4 and p_5. That mutual exclusion property is local to a small part of the net, and it is intuitively clear that the places $p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}$ are irrelevant to prove mutual exclusion between p_4 and p_5. Hence, the property can be proved with an abstraction of the Petri net as shown in Fig. 1(b) where the places $\{p_1, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}\}$ are not distinguished and merged into a single place p'_1. However, the current methods for solving coverability, when given the Petri net of Fig. 1(a) will consider the entire net and manipulate subsets of $\mathbb{N}^{12}$. Our method will automatically consider sets of lower dimensionality: in this case subsets of $\mathbb{N}^5$.

Our algorithm is based on two main ingredients: abstract interpretation and automatic refinement. Abstract interpretation [9] is a well-established technique to define, in a systematic way, abstractions of semantics. In our case, we will use the notion of Galois insertion to relate formally subsets in $\mathbb{N}^k$ with their abstract representation in $\mathbb{N}^{k'}$ with $k' < k$. This Galois insertion allows us to systematically design an abstract semantics that leads to efficient semi-algorithms to solve the coverability problem by manipulating *lower dimensional sets*. We will actually show that the original coverability problem reduces to a coverability problem of lower dimensionality and so our algorithm can reuse efficient implementations for the forward and backward analysis of those abstractions. When the abstract interpretation is inconclusive, because it is not precise enough, our algorithm automatically refines the abstract domain. This refinement ensures that the next analysis will be more precise. Moreover it guarantees that the abstract analysis will eventually be precise enough to decide the problem. The abstraction technique that we consider here uses all the information that has been computed by previous steps and is quite different from the technique known as *counterexample guided abstraction refinement* [6].

We have implemented our automatic abstraction technique and we have evaluated our new algorithm on several interesting examples of infinite state Petri nets taken from the literature. It turns out that our technique finds low dimensional systems that are sufficiently precise abstractions to establish the correctness of complex systems. We also have run our algorithm on finite state models of well-known mutual exclusion protocols. On those, the reduction in dimension is less spectacular but our algorithm still finds simplifications that would be very hard to find by hand.

To the best of our knowledge, this work is the first that tries to automatically abstract Petri nets by lowering their dimensionality and which provide an automatic refinement when the analysis is not conclusive. In [3, 12], the authors provide syntactical criterion to simplify Petri nets while our technique is based on semantics. Our technique provides automatically much coarser abstractions than the one we could obtain by applying rules in [3, 12].

2. Preliminaries and Outline

We start this section by recalling Petri nets, their semantics and the coverability problem. Then, we recall the main properties of existing algorithms to solve this coverability problem. We end the section by giving an outline of our new algorithm.

2.1. Petri nets and their (concrete) semantics

In the rest of the paper our model of computation is given by the Petri net formalism. Given a set S we denote by $|S|$ its cardinality.

Definition 2.1. (Petri nets)
A Petri net N is given by a tuple (P, T, F, m_0) where:

- P and T are finite disjoint sets of *places* and *transitions* respectively,
- $F = (\mathcal{I}, \mathcal{O})$ are two mappings: $\mathcal{I}, \mathcal{O} \colon P \times T \mapsto \mathbb{N}$ relating places and transitions. Once a linear order is set on P and T, $\mathcal{I}$ and $\mathcal{O}$ can be seen as $(|P|, |T|)$-matrices over $\mathbb{N}$ ($\mathbb{N}^{|P|\times|T|}$ for short). Let $t \in T$, $\mathcal{I}(t)$ (resp. $\mathcal{O}(t)$) denote the t-column vector in $\mathbb{N}^{|P|}$ of $\mathcal{I}$ (resp. $\mathcal{O}$).
- m_0 is the *initial marking*. A *marking* $m \in \mathbb{N}^{|P|}$ is a column vector giving a number $m(p)$ of tokens for each place $p \in P$. □

Throughout the paper we will use the letter k to denote $|P|$, i.e. the *dimensionality* of the net. We introduce the partial order $\leqslant \subseteq \mathbb{N}^k \times \mathbb{N}^k$ such that for all $m, m' \in \mathbb{N}^k : m \leqslant m'$ iff $m(i) \leq m'(i)$ for all $i \in [1..k]$ (where $[1..k]$ denotes the set $\{1, \ldots, k\}$). It turns out that $\leqslant$ is a well-quasi order (wqo for short) on $\mathbb{N}^k$ meaning that for every infinite sequence of markings $m_1, m_2, \ldots, m_i, \ldots$ there exists indices $i < j$ such that $m_i \leqslant m_j$. We also use $m \lneqq m'$ to denote that $m \leqslant m' \wedge m \neq m'$.

Definition 2.2. (Firing Rules of Petri net)
Given a Petri net $N = (P, T, F, m_0)$ and a marking $m \in \mathbb{N}^k$ we say that the transition t is *enabled* at m, written $m\,[t\rangle$, iff $\mathcal{I}(t) \leqslant m$. If t is enabled at m then the *firing* of t at m leads to a marking m', written $m\,[t\rangle\, m'$, such that $m' = m - \mathcal{I}(t) + \mathcal{O}(t)$. □

Given a Petri net we are interested in the set of markings it can reach. To formalize the set of reachable markings and variants of the reachability problem, we use the following lattice and the following operations on sets of markings.

Definition 2.3. Let $k \in \mathbb{N}$, the powerset lattice associated to $\mathbb{N}^k$ is the complete lattice $(\wp(\mathbb{N}^k), \subseteq, \cup, \cap, \emptyset, \mathbb{N}^k)$ having the powerset of $\mathbb{N}^k$ as a carrier, union and intersection as least upper bound and greatest lower bound operations, respectively and the empty set and $\mathbb{N}^k$ as the $\subseteq$-minimal and $\subseteq$-maximal elements, respectively. □

We use Church's lambda notation (so that F is $\lambda X.\, F(X)$) and use the composition operator $\circ$ on functions given by $(f \circ g)(x) = f(g(x))$. Sometimes we also use logical formulas. Given a logical formula ψ we write $[\![\psi]\!]$ for the set of its satisfying valuations.

Definition 2.4. (The predicate transformers pre, $\widetilde{\mathit{pre}}$, and post)
Let N a Petri net given by (P, T, F, m_0) and let $t \in T$, we define $\mathit{pre}_N[t], \widetilde{\mathit{pre}}_N[t], \mathit{post}_N[t] \colon \wp(\mathbb{N}^k) \mapsto \wp(\mathbb{N}^k)$ as follows,

$$\begin{aligned}
\mathit{pre}_N[t] &\stackrel{\text{def}}{=} \lambda X.\, \{m \in \mathbb{N}^k \mid \exists m' \colon m' \in X \wedge m\,[t\rangle\, m'\} \\
\widetilde{\mathit{pre}}_N[t] &\stackrel{\text{def}}{=} \lambda X.\, \{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m \vee m \in \mathit{pre}_N[t](X)\} \\
\mathit{post}_N[t] &\stackrel{\text{def}}{=} \lambda X.\, \{m \in \mathbb{N}^k \mid \exists m' \colon m' \in X \wedge m'\,[t\rangle\, m\} \ .
\end{aligned}$$

The extension to the set $T' \subseteq T$ of transitions is given by, $\lambda X.\, \bigcup_{t \in T'} f_N[t](X)$ if f_N is $\mathit{pre}_N[T']$ or $\mathit{post}_N[T']$; and $\lambda X.\, \bigcap_{t \in T'} f_N[t](X)$ for $f_N = \widetilde{\mathit{pre}}_N[T']$. □

To simplify notations, we use $\mathit{pre}_N, \widetilde{\mathit{pre}}_N$ and post_N instead of $\mathit{pre}_N[T], \widetilde{\mathit{pre}}_N[T]$ and $\mathit{post}_N[T]$.

In the sequel when the Petri net N is clear from the context we omit to mention N as a subscript. Finally we recall a well-known result which is proved for instance in [8]: for any $X, Y \subseteq \mathbb{N}^k$ we have

$$\mathit{post}(X) \subseteq Y \Leftrightarrow X \subseteq \widetilde{\mathit{pre}}(Y) \ . \tag{Gc}$$

Since all those predicate transformers are monotone functions over the complete lattice $(\wp(\mathbb{N}^k), \subseteq, \cup, \cap, \emptyset, \mathbb{N}^k)$ so they can be used as building blocks to define fixpoints expressions.

In $\wp(\mathbb{N}^k)$, *upward-closed* and *downward-closed* sets are particularly interesting and are defined as follows. We define the operator $\downarrow$ (resp. $\uparrow$) as $\lambda X.\, \{x' \in \mathbb{N}^k \mid \exists x \colon x \in X \wedge x' \leqslant x\}$ (resp. $\lambda X.\, \{x' \in \mathbb{N}^k \mid \exists x \colon x \in X \wedge x \leqslant x'\}$). A set S is $\leqslant$-downward closed ($\leqslant$-dc-set for short), respectively $\leqslant$-upward closed ($\leqslant$-uc-set for short), iff $\downarrow S = S$, respectively $\uparrow S = S$. We define $\mathit{DCS}(\mathbb{N}^k)$ ($\mathit{UCS}(\mathbb{N}^k)$) to be the set of all $\leqslant$-dc-sets ($\leqslant$-uc-sets). A set $M \subseteq \mathbb{N}^k$ is said to be *canonical* if for any distinct $x, y \in M$ we have $x \nleqslant y$. We say that M is a *minor set* of $S \subseteq \mathbb{N}^k$, if $M \subseteq S$ and $\forall s \in S\, \exists m \in M \colon m \leqslant s$.

Lemma 2.1. ([11])
Given $S \subseteq \mathbb{N}^k$, S has exactly one finite canonical minor set.

So, every $\leqslant$-uc-set U can be represented by its finite canonical minor set, written $\min(U)$, because $\uparrow \min(U) = U$. Ad-hoc data structures have been studied to represent compactly and manipulate efficiently minor sets [26]. Since each $\leqslant$-dc-set is the complement of a $\leqslant$-uc-set, we conclude that it has an effective representation. We also refer the interested reader to [22] where the authors define the ω-markings which is an alternative representation for $\leqslant$-dc-set.

We now formally state the coverability problem for Petri nets which corresponds to a fixpoint checking problem. Our formulation follows [8].

Problem 1. Given a Petri net N and a $\leqslant$-dc-set S, we want to check if the inclusion holds:

$$\mathit{lfp}\lambda X.\, \{m_0\} \cup \mathit{post}(X) \subseteq S \tag{1}$$

which, by [8, Thm. 4], is equivalent to

$$\{m_0\} \subseteq \mathit{gfp}\lambda X.\, S \cap \widetilde{\mathit{pre}}(X) \ . \tag{2}$$

We write $\mathit{post}^*(m_0)$ and $\widetilde{\mathit{pre}}^*(S)$ to be the fixpoints of expressions (1) and (2), respectively. They are called the *forward semantics* and the *backward semantics* of the net. They respectively denote the set of *reachable markings* and the set of *markings stuck in* S (or stated otherwise the *markings that cannot escape from the set* S *of markings*). So, with this view in mind, we find that the fixpoint checking problem asks, given a Petri Net N and a set S of markings, whether the reachable markings are included in S or equivalently if the initial marking m_0 belongs to the set of markings that cannot escape from S.

Note also that since S is a $\leqslant$-dc-set, $post^*(m_0) \subseteq S$ if and only if $\downarrow\!(post^*(m_0)) \subseteq S$.

2.2. Existing algorithms

The solutions to Problem 1 found in the literature (see [16, 18]) iteratively compute finer overapproximations of $\downarrow\!(post^*(m_0))$. All these solutions have in input an effective representation for (i) the initial marking m_0, (ii) the predicate transformer $post_N$ associated to the Petri net N and (iii) the $\leqslant$-dc-set S. They end up with an overapproximation $\mathcal{R}$ satisfying the following properties:

$$post_N^*(m_0) \subseteq \mathcal{R} \tag{A1}$$

$$\mathcal{R} \in DCS(\mathbb{N}^k) \tag{A2}$$

$$post_N(\mathcal{R}) \subseteq \mathcal{R} \tag{A3}$$

$$post_N^*(m_0) \subseteq S \rightarrow \mathcal{R} \subseteq S \tag{A4}$$

The solutions of [16, 18] actually solve Problem 1 for the entire class of well-structured transition systems (WSTS for short) which includes Petri nets and many other interesting infinite state models. In [22, 19] the authors show that $\downarrow\!(post^*(m_0))$ is computable for Petri nets and thus the approximation scheme presented above also encompasses these solutions.

In [1] an algorithm to compute the set $\widetilde{pre}^*(S)$ by evaluating its associated fixpoint (2) is given. The algorithm works for the whole class of WSTS and thus also for Petri net [1]

All these algorithms for Petri nets suffer from the explosion caused by the high dimensionality of the state spaces of practical examples. In this paper, we develop an analysis that reduces the dimensionality of state spaces that are explored during verification.

2.3. Overview of our approach

In order to mitigate the dimensionality problem, we adopt the following strategy. First, we define a parametric abstract domain where subsets of $\mathbb{N}^k$ are abstracted by subsets of $\mathbb{N}^{k'}$ where $k' < k$ (k' being a parameter). More precisely, when each dimension in the concrete domain records the number of tokens contained in each place of the Petri net, in the abstract domain, each dimension records the sum of the number of tokens contained in a set of places. Using this abstract domain, we define abstract forward and abstract backward semantics, and define efficient algorithms to compute them. In those semantics, sets of markings are represented by subsets of $\mathbb{N}^{k'}$. If the abstract semantics is not conclusive (the abstract analysis returned a “don’t know” answer), it is refined automatically using a refinement procedure that is guided by the inconclusive abstract semantics. During the refinement steps, we identify important concrete sets and refine the current abstract domain to allow the exact representation of those sets.

The rest of our paper formalizes those ideas and is organized as follows. In Sect. 3, we define our parametric abstract domain and we specify the abstract semantics. We also show how the precision of different domains of the family can be related. In the section, we also establish the existence of the coarsest abstract domain able to represent exactly a given set M of markings. This result is important for the automatic refinement. In Sect. 4, we define an efficient way to overapproximate the abstract semantics defined in Sect. 3. In Sect. 5, we put all those results together to obtain our algorithm that decides coverability by successive approximations and refinements. Section 5 is also devoted to establishing the correctness of our algorithm. Section 6 identifies the bottlenecks of the algorithm and proposes several improvements which are validated by the experimental results which are given in Sect. 7.

[1] The fixpoint expression considered in [1] is actually different from (2) but coincides with its complement. This is a consequence of the Park’s theorem [25].

3. Abstraction of Sets of Markings

3.1. Partitions

At the basis of our abstraction technique are the *partitions* (of the set of places).

Definition 3.1. Let A be a partition of the set $[1..k]$ into k_A *classes* $\{C_i\}_{i\in[1..k_A]}$. We define the order $\preceq$ over partitions as follows: $A \preceq A'$ iff $\forall C \in A\, \exists C' \in A' \colon C \subseteq C'$. It is well known, see [4], that the set of partitions of $[1..k]$ together with $\preceq$ form a complete lattice where $\{\{1\},\ldots,\{k\}\}$ is the $\preceq$-minimal element, $\{\{1,\ldots,k\}\}$ is the $\preceq$-maximal element and the greatest lower bound of two partitions A_1 and A_2, noted $A_1 \curlywedge A_2$, is the partition given by $\{C \mid \exists C_1 \in A_1\, \exists C_2 \in A_2 \colon C = C_1 \cap C_2 \text{ and } C \neq \emptyset\}$. The least upper bound of two partitions A_1 and A_2, noted $A_1 \curlyvee A_2$, is the finest partition such that given $C \in A_1 \cup A_2$ and $\{a_1, a_2\} \subseteq C$ we have $\exists C' \in A_1 \curlyvee A_2 \colon \{a_1, a_2\} \subseteq C'$. □

Example 3.1. Given the set $\{a, b, c\}$ and two partitions $A_1 = \{\{a, b\}, \{c\}\}$ and $A_2 = \{\{a, c\}, \{b\}\}$. We have that $A_1 \curlywedge A_2 = \{\{a\}, \{b\}, \{c\}\}$ and $A_1 \curlyvee A_2 = \{a, b, c\}$.

Partitions will be used to abstract sets of markings by lowering their dimensionality. Given a marking m (viz. a k-uple) and a partition A of $[1..k]$ into k_A classes we abstract m into a k_A-uple m_A by summing all the coordinates of each class. A simple way to apply the abstraction on a marking m is done by computing the product of a matrix A with the vector of m (noted $A \cdot m$). So we introduce a matrix based definition for partitions.

Definition 3.2. Let A be a partition of $[1..k]$ given by $\{C_i\}_{i\in[1..k_A]}$. We associate to this partition a matrix $A := (a_{ij})_{k_A \times k}$ such that $a_{ij} = 1$ if $j \in C_i$, $a_{ij} = 0$ otherwise. So, $A \in \{0, 1\}^{k_A \times k}$. We write $\mathcal{A}^{k_A \times k}$ to denote the set of matrices associated to the partitions of $[1..k]$ into k_A classes. Given a matrix A, we sometimes use k_A to denote the number of rows in A. □

From the above definition, we deduce that two distinct partitions A_1 and A_2 lead to two different matrices. We also find that a matrix A matches a unique partition. Also note that, since in the matrix based representation a linear order is given on the rows, permuting rows does not modify the associated partition. Hence, whenever a partition A has several classes, several matrices match A. However if a linear order is fixed on the classes of the partitions then the associated matrix is unique. Henceforth we assume that each partition is provided with a linear order on its classes so that we can identify a partition with its unique associated matrix.

3.2. Abstract Semantics

We are now equipped to define an abstraction technique for sets of markings. Then we focus on the abstraction of the predicate transformers involved in the fixpoints of (1) and (2).

Definition 3.3. Let $A \in \mathcal{A}^{k_A \times k}$, we define the abstraction function $\alpha_A \colon \wp(\mathbb{N}^k) \mapsto \wp(\mathbb{N}^{k_A})$ and the concretization function $\gamma_A \colon \wp(\mathbb{N}^{k_A}) \mapsto \wp(\mathbb{N}^k)$ respectively as follows

$$\alpha_A \overset{\text{def}}{=} \lambda X. \{A \cdot x \mid x \in X\} \qquad \gamma_A \overset{\text{def}}{=} \lambda X. \{x \mid A \cdot x \in X\} \ .$$

□

Furthermore, if A is clear from the context, we will write α (resp. γ) instead of α_A (resp. γ_A). Given the posets $\langle L, \leqslant\rangle$ and $\langle M, \sqsubseteq\rangle$ and the maps $\alpha \in L \mapsto M$, $\gamma \in M \mapsto L$, we write $\langle L, \leqslant\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$ if they form a Galois insertion [9], that is $\forall x \in L, \forall y \in M \colon \alpha(x) \sqsubseteq y \Leftrightarrow x \leqslant \gamma(y)$ and $\alpha \circ \gamma = \lambda x. x$.

Proposition 3.1. Let $A \in \mathcal{A}^{k_A \times k}$, we have $(\wp(\mathbb{N}^k), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\wp(\mathbb{N}^{k_A}), \subseteq)$.

Proof:
Let $X \subseteq \mathbb{N}^k$ and $Y \subseteq \mathbb{N}^{k_A}$,

$$\begin{aligned} \alpha(X) \subseteq Y &\Leftrightarrow \{A{\cdot}\, x \mid x \in X\} \subseteq Y && \text{def. 3.3} \\ &\Leftrightarrow \forall x\colon x \in X \rightarrow A{\cdot}\, x \in Y && \\ &\Leftrightarrow X \subseteq \{x \mid A.x \in Y\} && \\ &\Leftrightarrow X \subseteq \gamma(Y) && \text{def. 3.3} \end{aligned}$$

Now, we prove that $\alpha \circ \gamma = \lambda x.\, x$. Given $y \in Y$, we define $x \in \mathbb{N}^k$ such that for each class C_i of A we choose $j \in C_i$ and set $x(j) = y(i)$ and $x(k) = 0$ for $k \in C_i \setminus \{j\}$. It is routine to check that $A \cdot x = y$.

$$\begin{aligned} \alpha \circ \gamma(Y) &= \alpha(\{x \mid A{\cdot}\, x \in Y\}) && \\ &= \{A{\cdot}\, x \mid A{\cdot}\, x \in Y\} && \text{def. 3.3} \\ &= Y && \text{by above} \end{aligned}$$

□

In the sequel we use the property that the abstraction function α is *additive* (i.e. $\alpha(A \cup B) = \alpha(A) \cup \alpha(B)$) and that γ is *co-additive* (i.e. $\gamma(A \cap B) = \gamma(A) \cap \gamma(B)$).

The Petri Net N depicted on the right has three places $\{p_1, p_2, p_3\}$ and three transitions $\{t_1, t_2, t_3\}$, and its initial marking m_0 is given by $\langle 0, 1, 0\rangle$. Let $A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ be the matrix associated to the partition $\{\{p_1, p_3\}, \{p_2\}\}$ of the places of N. We have that $\alpha(m_0) = A{\cdot}\, (0\;1\;0)^T = \langle 0, 1\rangle$ by def. 3.3, hence that $\gamma \circ \alpha(m_0) = \gamma(\langle 0, 1\rangle) = \langle 0, 1, 0\rangle$ by def. 3.3, and so m_0 is represented exactly. However $\gamma \circ \alpha(\langle 1, 1, 0\rangle) = \gamma(\langle 1, 1\rangle) = \{\langle 1, 1, 0\rangle, \langle 0, 1, 1\rangle\}$, hence $\langle 1, 1, 0\rangle$ is not represented exactly by $\alpha(\langle 1, 1, 0\rangle) = \langle 1, 1\rangle$.

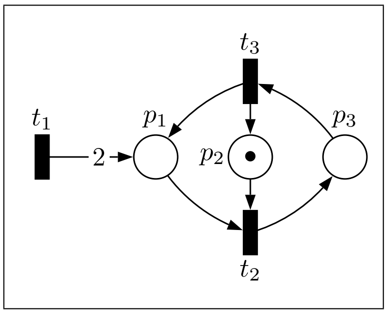

Figure 2. An example of abstraction associated to a Petri Net .

Fig. 2 illustrates the above definitions. Given a Galois insertion, the theory of abstract interpretation [9] provides us with a theoretical framework to systematically derive approximate semantics. The concrete forward semantics of a Petri net N is given by $post^*_N(m_0)$. Since we have a Galois insertion, $post^*_N(m_0)$ has a unique best approximation in the abstract domain. This value is $\alpha(post^*_N(m_0))$.

Unfortunately, there is no general method to compute this approximation without computing $post^*_N(m_0)$ first. So instead of trying to compute this abstract value, we compute an overapproximation. Let $\mathcal{F}$ be an overapproximation of $\alpha(post^*_N(m_0))$ and let $\mathcal{B}$ be an overapproximation $\alpha(\widetilde{pre}^*_N(S))$. The following lemma which easily follows from the results of [7] shows the usefulness of such approximations.

Lemma 3.1. *Given a Petri net N and a $\preccurlyeq$-dc-set S we have $\gamma(\mathcal{F}) \subseteq S \rightarrow post^*_N(m_0) \subseteq S$ and $\{m_0\} \nsubseteq \gamma(\mathcal{B}) \rightarrow post^*_N(m_0) \nsubseteq S$.*

Abstract interpretation [9] tells us that to compute an overapproximation of fixpoints of a concrete function f, what we can do is to define a function $f^\sharp$ over the abstract domain which is such that $f^\sharp$ is an *abstract counterpart* of f. Intuitively, it means that $f^\sharp$ must approximate the function f. In [9] the authors show that, in the context of a Galois insertion, the most precise approximation of f is unique and coincides with $\alpha \circ f \circ \gamma$ which is called the *best abstract counterpart* of f so that each abstract counterpart $f^\sharp$ of f is less precise than $\alpha \circ f \circ \gamma$.

So to approximate $\alpha(\mathit{lfp} f)$ (resp. $\mathit{lfp} f$) a solution is to compute $\mathit{lfp} f^\sharp$ (resp. $\gamma(\mathit{lfp} f^\sharp)$) where $f^\sharp$ is an abstract counterpart of f. A similar reasoning holds for *gfp* expressions. As a possible instantiation for our context, we obtain the following:

$$\alpha(\mathit{post}^*_N(m_0)) \subseteq \mathit{lfp}\lambda X.\,\alpha(\{m_0\} \cup \mathit{post}(\gamma(X))) \text{ and } \alpha(\widetilde{\mathit{pre}}^*_N(S)) \subseteq \mathit{gfp}\lambda X.\,\alpha(S \cap \widetilde{\mathit{pre}}(\gamma(X))) \quad (3)$$

It is worth pointing that using properties of Galois insertions we deduce the following:

$$\mathit{post}^*_N(m_0) \subseteq \gamma\big(\mathit{lfp}\lambda X.\,\alpha(\{m_0\} \cup \mathit{post}(\gamma(X)))\big) \text{ and } \widetilde{\mathit{pre}}^*_N(S) \subseteq \gamma\big(\mathit{gfp}\lambda X.\,\alpha(S \cap \widetilde{\mathit{pre}}(\gamma(X)))\big) \ . \quad (4)$$

Note that the specification $\alpha \circ f \circ \gamma$ of the best abstract counterpart of f naturally suggests to concretize the argument, then apply f and finally to abstract its result. In practice applying this methodology leads to inefficient algorithms. Indeed the explicit computation of γ is in general costly. In our settings it happens that given an effective representation of M the effective representation of the set $\gamma(M)$ could be exponentially larger. In fact, let A be a partition of $[1..k]$ given by $\{C_i\}_{i\in[1..k_A]}$ and let $\hat{m} \in \mathbb{N}^{k_A}$, we have $|\gamma(\hat{m})| = \prod_{i\in[1..k_A]} \binom{\hat{m}(i)+|C_i|-1}{|C_i|-1}$. Section 4 is devoted to the definition of the abstract counterpart of predicate transformers (possibly the best one) without explicitly evaluating γ.

3.3. Refinement

As mentioned in Sect. 2, our algorithm is based on the abstraction refinement paradigm. In that context, if the current abstraction A_i is inconclusive we refine it into an abstraction A_{i+1} which overapproximates sets of markings and predicate transformers more precisely than A_i.

The first result states that the concretization function for the $\preceq$-minimal partition defines a bijection on $\wp(\mathbb{N}^k)$ which implies that every two distinct sets of markings match distinct abstract values.

Lemma 3.2. Let $A_\perp$ denote the partition $\{\{1\},\ldots\{k\}\}$. We have $\gamma_{A_\perp}(\wp(\mathbb{N}^k)) = \wp(\mathbb{N}^k)$.

Proof:
Each class of the partition $A_\perp$ is a singleton. Hence the associated matrix is the identity matrix (up to a permutation of the rows) and so the result follows. □

The next lemma relates partitions and the precision of the abstract domains thereof.

Lemma 3.3. Let A_1, A_2 be two partitions of $[1..k]$ s.t. $A_1 \preceq A_2$, we have $\gamma_{A_1}(\wp(\mathbb{N}^{k_{A_1}})) \subseteq \gamma_{A_2}(\wp(\mathbb{N}^{k_{A_2}}))$

Proof:
By definition of γ_A, we have that $\gamma_A(X) = \bigcup_{m\in X} \gamma_A(\{m\})$. Hence, it is sufficient to prove that

$$\gamma_{A_1}(\{\{m\} \mid m \in \mathbb{N}^{k_{A_1}}\}) \subseteq \gamma_{A_2}(\{\{m\} \mid m \in \mathbb{N}^{k_{A_2}}\}) \ .$$

Also by the surjectivity of the abstract function (recall that $\alpha \circ \gamma = \lambda x.\,x$ holds by Galois insertion) it is sufficient to prove the following. Let $\widehat{m} \in \mathbb{N}^k$, we denote by m and m' the markings $\alpha_{A_1}(\widehat{m})$ and $\alpha_{A_2}(\widehat{m})$, respectively; we show that $\gamma_{A_1}(m) \subseteq \gamma_{A_2}(m')$.

We conclude from $A_1 \preceq A_2$ that for each $C' \in A_2$ the set $\wp(C') \cap A_1$ is a partition of C', hence that

$$\sum_{\substack{C\in\wp(C')\\ C\in A_1}} m(C) = m'(C') \quad (5)$$

by definition of m', m. Then, we have

$$\begin{aligned}
&\{m_1 \in \mathbb{N}^k \mid \sum_{s' \in C'} m_1(s') = m'(C')\} \\
&= \{m_1 \in \mathbb{N}^k \mid \sum_{s' \in C'} m_1(s') = \sum_{\substack{C \in \wp(C') \\ C \in A_1}} m(C)\} && \text{by (5)} \\
&\supseteq \{m_1 \in \mathbb{N}^k \mid \bigwedge_{\substack{C \in \wp(C') \\ C \in A_1}} \sum_{s \in C} m_1(s) = m(C)\} && \begin{pmatrix} a = b \wedge \\ c = d \end{pmatrix} \to \begin{pmatrix} a + c = \\ b + d \end{pmatrix} \\
&= \bigcap_{\substack{C \in \wp(C') \\ C \in A_1}} \{m_1 \in \mathbb{N}^k \mid \sum_{s \in C} m_1(s) = m(C)\}
\end{aligned} \tag{6}$$

Finally,

$$\begin{aligned}
\gamma_{A_2}(\{m'\}) &= \bigcap_{C' \in A_2} \{m_1 \in \mathbb{N}^k \mid \sum_{s' \in C'} m_1(s') = m'(C')\} && \text{by def. of } \gamma_{A_2} \\
&\supseteq \bigcap_{C' \in A_2} \bigcap_{\substack{C \in \wp(C') \\ C \in A_1}} \{m_1 \in \mathbb{N}^k \mid \sum_{s \in C} m_1(s) = m(C)\} && \text{by (6)} \\
&= \bigcap_{C \in A_1} \{m_1 \in \mathbb{N}^k \mid \sum_{s \in C} m_1(s) = m(C)\} && A_1 \preceq A_2 \\
&= \gamma_{A_1}(\{m\}) && \text{def. of } \gamma_{A_1}
\end{aligned}$$

□

Corollary 3.1. Let A_1, A_2 be two partitions of $[1..k]$ and let $A = A_1 \curlywedge A_2$, we have

$$\gamma_{A_1}(\wp(\mathbb{N}^{k_{A_1}})) \cup \gamma_{A_2}(\wp(\mathbb{N}^{k_{A_2}})) \subseteq \gamma_A(\wp(\mathbb{N}^{k_A})) \ .$$

So by refining partitions, we refine abstractions. The following result tells us that if two partitions are able to represent exactly a set then their *join* is also able to represent that set.

Lemma 3.4. Let A_1, A_2 be two partitions of $[1..k]$ and let $A = A_1 \curlyvee A_2$. For all $M \subseteq \mathbb{N}^k$ we have

$$\text{if } \left\{ \begin{array}{l} \gamma_{A_1} \circ \alpha_{A_1}(M) = M \\ \gamma_{A_2} \circ \alpha_{A_2}(M) = M \end{array} \right\} \text{ then } \gamma_A \circ \alpha_A(M) = M \ . \tag{7}$$

Proof:
We first observe that by property of Galois insertion we have that $\gamma'_A \circ \alpha'_A(M) \supseteq M$ for each partition A' and each set of markings M. We thus concentrate on the reverse inclusion property. Given an abstraction A, we define $\mu_A = \gamma_A \circ \alpha_A$. Let $m \in M$ and $m' \in \mu_A(\{m\})$. We will show that there exists a finite sequence $\mu_{A_{i_1}}, \mu_{A_{i_2}}, \ldots, \mu_{A_{i_n}}$ such that $m' \in \mu_{A_{i_1}} \circ \mu_{A_{i_2}} \circ \ldots \circ \mu_{A_{i_n}}(\{m\})$ and $\forall j \in [1..n] \colon i_j \in [1..2]$. Then we will conclude that $m' \in M$ by the left hand side of (7).

It is well known that given a set S, the set of partitions of S coincides with the set of equivalence classes in S. So we denote by $\equiv_A$ the equivalence relation defined by the partition A.

We thus get $m' \in \mu_A(\{m\})$ iff m' may be obtained from m by moving tokens inside the equivalence classes of $\equiv_A$. More precisely, let $v \in \mathbb{N}$, and a, b be two distinct elements of $[1..k]$ such that $\langle a, b\rangle \in \equiv_A$

and two markings $m_1, m_2 \in \mathbb{N}^k$ such that

$$m_2(q) = \begin{cases} m_1(q) + v & \text{if } q = a \\ m_1(q) - v & \text{if } q = b \\ m_1(q) & \text{otherwise.} \end{cases}$$

Intuitively the marking m_2 is obtained from m_1 by moving v tokens from b into a. So, since on the one hand b and a belong to the same equivalence class and, on the other hand m_2 and m_1 contain an equal number of tokens, we find that $m_2 \in \mu_A(\{m_1\})$.

Now, we use the result of [4, Thm. 4.6] over the equivalence classes of a set. The theorem states that $\langle a, b\rangle \in \equiv_A$ iff there is a sequence of elements $c_1, \ldots, c_{n'}$ of $[1..k]$ such that

$$\langle c_i, c_{i+1}\rangle \in \equiv_{A_1} \qquad \text{or} \qquad \langle c_i, c_{i+1}\rangle \in \equiv_{A_2} \tag{8}$$

for $i \in [1..n'-1]$ and $a = c_1$, $b = c_{n'}$. From $c_1, \ldots, c_{n'}$ we define a sequence of n' moves whose global effect is to move v tokens from b into a. So given m_1, the marking obtained by applying this sequence of n' moves is m_2. Moreover, by (8) we have that each move of the sequence is defined inside an equivalence class of $\equiv_{A_1}$ or $\equiv_{A_2}$. Hence each of those moves can be done using operator μ_{A_1} or μ_{A_2}.

Repeated application of the above reasoning shows that m' is obtained by moving tokens of m where moves are given by operators μ_{A_1} and μ_{A_2}. Formally this finite sequence of moves $\mu_{A_{i_1}}, \mu_{A_{i_2}}, \ldots, \mu_{A_{i_n}}$ is such that $\forall j \in [1..n]\colon i_j \in [1..2]$ and $m' \in \mu_{A_{i_1}} \circ \mu_{A_{i_2}} \circ \ldots \circ \mu_{A_{i_n}}(\{m\})$. Finally, left hand side of (7) and monotonicity of μ_{A_1}, μ_{A_2} shows that $m' \in M$. □

Corollary 3.2. Let A_1, A_2 be two partitions of $[1..k]$ and let $A = A_1 \curlyvee A_2$, we have

$$\gamma_{A_1}(\wp(\mathbb{N}^{k_{A_1}})) \cap \gamma_{A_2}(\wp(\mathbb{N}^{k_{A_2}})) \subseteq \gamma_A(\wp(\mathbb{N}^{k_A})) \ .$$

Proposition 3.2. Given $k \in \mathbb{N}$ and $M \subseteq \mathbb{N}^k$, the coarsest partition of $[1..k]$ which represents exactly M exists and is given by $\bigcurlyvee\{A \mid \gamma_A \circ \alpha_A(M) = M\}$.

Proof:
From Lem. 3.2 there is at least one partition that represents exactly M. Moreover Cor. 3.2 shows that $\bigcurlyvee\{A \mid \gamma_A \circ \alpha_A(M) = M\}$ is unique by definition of the lattice of partitions, represents exactly M and is the coarsest. □

We end this section by a series a technical results that are needed in the sequel.

Lemma 3.5. Let $A \in \mathcal{A}^{k_A \times k}$ and $X, Y \subseteq \mathbb{N}^k$,

$$\gamma \circ \big\downarrow(X) = \downarrow \circ\, \gamma(X), \gamma \circ \big\uparrow(X) = \uparrow \circ\, \gamma(X)$$

$$\alpha \circ \big\downarrow(X) = \downarrow \circ\, \alpha(X), \alpha \circ \big\uparrow(X) = \uparrow \circ\, \alpha(X)$$

$$\forall m_1, m_2 \in \mathbb{N}^k\colon m_1 \lneqq m_2 \text{ implies } \alpha(m_1) \lneqq \alpha(m_2) \qquad \text{(strict monotonicity)}$$

$$\forall m_3 \in \gamma(m_1)\, \exists m_4 \in \gamma(m_2)\colon m_4 \gneqq m_3 \qquad \text{for each } m_1, m_2 \in \mathbb{N}^{k_A} \text{ such that } m_1 \lneqq m_2$$

$$\gamma(\neg X) = \neg \circ \gamma(X)$$

$$\alpha(\neg X) = \neg \circ \alpha(X) \qquad \text{provided } \gamma \circ \alpha(X) = X$$

$$\alpha(Y \cap X) = \alpha(Y) \cap \alpha(X) \qquad \text{provided } \gamma \circ \alpha(X) = X$$

Lemma 3.6. Let $A \in \mathcal{A}^{k_A \times k}$, the set $\gamma_A(\mathbb{N}^{k_A})$ is closed under union, intersection and complement.

Proof:
We know that each Galois connection satisfies the Moore closure property, which means that the set $\gamma_A(\mathbb{N}^{k_A})$ is closed to intersection. Next we have that

$$\begin{aligned} \gamma_A \circ \alpha_A(S) = S &\Leftrightarrow \neg \circ \gamma_A \circ \alpha_A(S) = \neg S \\ &\Leftrightarrow \gamma_A \circ \neg \circ \alpha_A(S) = \neg S && \text{Lem. 3.5} \\ &\Rightarrow \gamma_A \circ \alpha_A(\neg S) = \neg S && \text{Lem. 3.5} \end{aligned}$$

Finally, note that $S_1 \cup S_2 = \neg(\neg S_1 \cap \neg S_2)$. Hence, the closure under intersection and complement implies the closure under union. □

Moreover, Lemma 3.6 implies that the concretization function is additive as shown below.

Lemma 3.7. Let $A \in \mathcal{A}^{k_A \times k}$ and $S_1, S_2 \subseteq \mathbb{N}^{k_A}$, we have that $\gamma_A(S_1 \cup S_2) = \gamma_A(S_1) \cup \gamma_A(S_2)$.

Proof:

$$\begin{aligned} \gamma_A(S_1 \cup S_2) &= \gamma_A(\alpha_A \circ \gamma_A(S_1) \cup \alpha_A \circ \gamma_A(S_2)) && S_i = \alpha_A \circ \gamma_A(S_i) \text{ by Galois insertion} \\ &= \gamma_A \circ \alpha_A(\gamma_A(S_1) \cup \gamma_A(S_2)) && \alpha_A \text{ is additive} \end{aligned}$$

Finally, applying Lemma 3.6, we conclude that $\gamma_A \circ \alpha_A(\gamma_A(S_1) \cup \gamma_A(S_2)) = \gamma_A(S_1) \cup \gamma_A(S_2)$. □

4. Efficient Abstract Semantics

In this section, we show how to compute a precise overapproximation of the abstract semantics efficiently without evaluating the concretization function γ. For that, we show that to each Petri net N of dimensionality k and each abstraction $A \in \mathcal{A}^{k_A \times k}$, we can associate a Petri net $\hat{N}$ of dimensionality k_A whose concrete forward and backward semantics gives precise overapproximations of the abstraction by A of the semantics of N as we previously stated in (3), p. 8.

4.1. Abstract net

To evaluate the best abstract counterpart of $post_N[t]$ and $\widetilde{pre}_N[t]$ for each $t \in T$, without explicitly evaluating γ, we associate for each Petri net N and abstraction A a Petri net $\hat{N}$.

Definition 4.1. Let N be a Petri net given by (P, T, F, m_0) and let $A \in \mathcal{A}^{k_A \times k}$. We define the tuple $(\hat{P}, T, \hat{F}, \widehat{m_0})$ where $\hat{P}$ is a set of k_A places (one for each class of the partition A); $\hat{F} = (\widehat{\mathcal{I}}, \widehat{\mathcal{O}})$ is such that $\widehat{\mathcal{I}} \overset{\text{def}}{=} A \cdot \mathcal{I}$ and $\widehat{\mathcal{O}} \overset{\text{def}}{=} A \cdot \mathcal{O}$; $\widehat{m_0}$ is given by $A \cdot m_0$. The tuple is a Petri net since $\widehat{m_0} \in \mathbb{N}^{|\hat{P}|}$, and $\widehat{\mathcal{I}}, \widehat{\mathcal{O}} \in \mathbb{N}^{(|\hat{P}|, |T|)}$. We denote by $\hat{N}$ this Petri net. □

Fig. 3 illustrates the definition of abstract net given the partition $\{\{p_1, p_3\}, \{p_2\}\}$. At this early stage we already can noticed that the forward semantics of the abstracted net $\hat{N}$ fails to establish that the place p_2 is 1-safe, which holds on the original net N. In fact, $post^*_{\hat{N}}(\widehat{m_0}) = \{(2i, j) \mid 2i + j \geq 1\}$. Accordingly a refinement step has to take place. In this paper, our refinement refines the partition defining the abstract net, i.e. it splits classes of the partition. Hence a new abstrated net is defined.

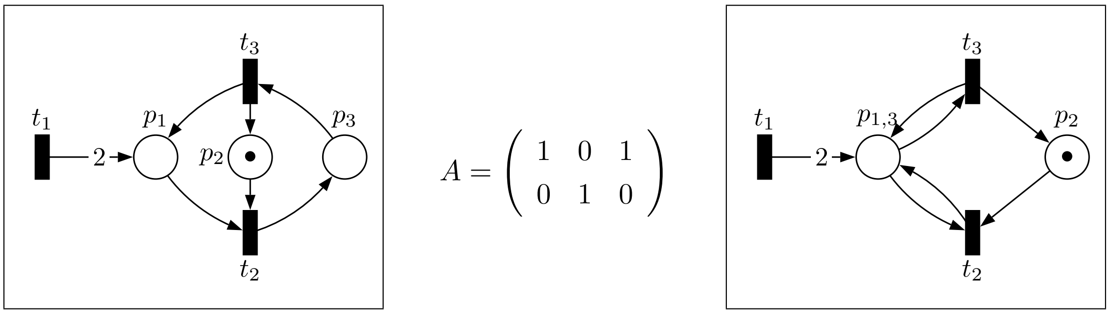

Figure 3. A Petri net N, a partition A of its places, and the associated abstract Petri net $\hat{N}$.

4.2. Forward overapproximation

The next proposition states that the best abstract counterpart of the predicate transformer $post_N$ is given by the predicate transformer $post_{\hat{N}}$ of the abstract net.

Proposition 4.1. Given a Petri net $N = (P, T, F, m_0)$, $A \in \mathcal{A}^{k_A \times k}$ and $\hat{N}$ the Petri net given by def. 4.1, we have that $\lambda X.\, \alpha \circ post_N \circ \gamma(X) = \lambda X.\, post_{\hat{N}}(X)$.

Proof:
Definition 2.4 states that $post_N = \lambda X. \bigcup_{t \in T} post_N[t](X)$. Thus, for $t \in T$, we show that $\alpha \circ post_N[t] \circ \gamma = post_{\hat{N}}[t]$. Then the additivity of α shows the desired result.

For each $t \in T$, for each $\hat{m} \in \mathbb{N}^{k_A}$,

$$\begin{aligned}
&\alpha \circ post_N[t] \circ \gamma(\hat{m}) \\
&= \alpha \circ post_N[t](\{m \mid m \in \gamma(\hat{m})\}) \\
&= \alpha(\{m - \mathcal{I}(t) + \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\}) && \text{def. 2.2} \\
&= \{A \cdot (m - \mathcal{I}(t) + \mathcal{O}(t)) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} && \text{def. 3.3} \\
&= \{A \cdot m - A \cdot \mathcal{I}(t) + A \cdot \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} \\
&= \{\alpha(m) - A \cdot \mathcal{I}(t) + A \cdot \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} && \text{def. of } \alpha \\
&= \{\hat{m} - A \cdot \mathcal{I}(t) + A \cdot \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} && \overset{\gamma}{\underset{\alpha}{\leftrightarrows}} \\
&= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} && \text{def. 4.1} \\
&= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \{\mathcal{I}(t)\} \subseteq {\downarrow} \circ \gamma(\hat{m})\} && \text{def. of } {\downarrow}
\end{aligned}$$

$$\begin{aligned}
&= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \{\mathcal{I}(t)\} \subseteq \gamma \circ {\downarrow}(\hat{m})\} && \text{Lem. 3.5} \\
&= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \alpha(\{\mathcal{I}(t)\}) \subseteq {\downarrow}(\hat{m})\} && \overset{\gamma}{\underset{\alpha}{\leftrightarrows}} \\
&= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \widehat{\mathcal{I}}(t) \leqslant \hat{m}\} && \text{def. 4.1} \\
&= post_{\hat{N}}[t](\hat{m}) && \text{def. 2.2}
\end{aligned}$$

□

The consequences of Prop 4.1 are twofold. First, it gives a way to compute $\alpha \circ post_N \circ \gamma$ without computing explicitly γ and second since $post_{\hat{N}} = \alpha \circ post_N \circ \gamma$ and $\widehat{N}$ is a Petri net we can use any state of the art tool to check whether $post^*_{\hat{N}}(\widehat{m_0}) \subseteq \alpha(S)$ and conclude, provided $\gamma \circ \alpha(S) = S$, that $\gamma(post^*_{\hat{N}}(\widehat{m_0})) \subseteq S$, hence that $post^*_N(m_0) \subseteq S$ by Lem. 3.1.

4.3. Backward overapproximation

Below we show that given a Petri Net $N = (P, T, F, m_0)$ the best abstract counterpart of the predicate transformer $\widetilde{pre}_N[t]$ (for $t \in T$) can be computed using $\widetilde{pre}_{\hat{N}}[t]$ where $\hat{N}$ is given as in def. 4.1. However, as we will see this result does not extend to $\widetilde{pre}_N[T]$. To obtain those results, we need some intermediary lemmas (i.e. Lem. 4.1, 4.2 and 4.3).

Lemma 4.1. Given a Petri net $N = (P, T, F, m_0)$, $A \in \mathcal{A}^{k_A \times k}$ and $\hat{N}$ the Petri net given by def. 4.1, we have that $\lambda X.\, \alpha \circ pre_N \circ \gamma(X) = \lambda X.\, pre_{\hat{N}}(X)$.

Proof:
The proof is similar to the proof of Prop. 4.1 with $\mathcal{O}$ (resp. $\widehat{\mathcal{O}}$) replaced by $\mathcal{I}$ (resp. $\widehat{\mathcal{I}}$) and vice versa. □

Lemma 4.2. Given a Petri net $N = (P, T, F, m_0)$ and a partition $A = \{C_j\}_{j \in [1..k_A]}$ of $[1..k]$, if $\exists i \in [1..k] \colon \mathcal{I}(i, t) > 0$ and $\{i\} \notin A$ then $\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) = \mathbb{N}^{k_A}$.

Proof:
Besides the hypothesis assume $i \in C_j$ and consider $l \in [1..k]$ such that $l \in C_j$ and $l \neq i$. The set $\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}$ is a $\leqslant$-dc-set given by the following formula:

$$\bigvee_{\substack{p \in [1..k] \\ \mathcal{I}(p,t) > 0}} x_p < \mathcal{I}(p, t) \enspace .$$

We conclude from $i \in [1..k]$ and $\mathcal{I}(i, t) > 0$ that $[\![x_i < \mathcal{I}(i, t)]\!] = \{\langle v_1, \dots, v_i, \dots, v_k\rangle \mid v_i < \mathcal{I}(i, t)\}$, hence that $\alpha([\![x_i < \mathcal{I}(i, t)]\!]) = \mathbb{N}^{k_A}$ by $\{i, l\} \subseteq C_j \in A$, and finally that $\alpha([\![x_i < \mathcal{I}(i, t)]\!]) \subseteq \alpha([\![\bigvee_{\substack{p \in [1..k] \\ \mathcal{I}(p,t) > 0}} x_p < \mathcal{I}(p, t)]\!])$ by additivity of α. It follows that $\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) = \alpha([\![\bigvee_{\substack{p \in [1..k] \\ \mathcal{I}(p,t) > 0}} x_p < \mathcal{I}(p, t)]\!]) = \mathbb{N}^{k_A}$. □

Lemma 4.3. Given a Petri net $N = (P, T, F, m_0)$, a partition $A = \{C_j\}_{j \in [1..k_A]}$ of $[1..k]$ and $\hat{N}$ the Petri net given by def. 4.1, if for each $i \in [1..k] \colon \mathcal{I}(i, t) > 0$ implies $\{i\} \in A$, then $\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) = \{m \in \mathbb{N}^{k_A} \mid \widehat{\mathcal{I}}(t) \nleqslant m)\}$.

Proof:

$$\begin{aligned}
&\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) \\
&= \{A \cdot m \mid m \in \mathbb{N}^k \wedge \mathcal{I}(t) \nleqslant m\} && \text{def. of } \alpha \\
&= \{A \cdot m \mid m \in \mathbb{N}^k \wedge A \cdot \mathcal{I}(t) \nleqslant A \cdot m\} && \text{if } \mathcal{I}(i,t) > 0 \text{ then } \{i\} \in A \text{ for each } i \in [1..k] \\
&= \{A \cdot m \mid m \in \mathbb{N}^k \wedge \widehat{\mathcal{I}}(t) \nleqslant A \cdot m\} && \text{def. of } \widehat{\mathcal{I}} \\
&= \{\widehat{m} \in \mathbb{N}^{k_A} \mid \exists m \in \mathbb{N}^k \colon \widehat{m} = A \cdot m \wedge \widehat{\mathcal{I}}(t) \nleqslant \widehat{m}\} \\
&= \{\widehat{m} \in \mathbb{N}^{k_A} \mid \widehat{\mathcal{I}}(t) \nleqslant \widehat{m}\} && \text{tautology}
\end{aligned}$$

□

We are now ready to state and prove that $\widetilde{pre}_{\hat{N}}[t]$ is the best abstract counterpart of $\widetilde{pre}_N[t]$.

Proposition 4.2. Given a Petri net $N = (P, T, F, m_0)$, a partition $A = \{C_j\}_{j \in [1..k_A]}$ of $[1..k]$ and $\hat{N}$ the Petri net given by def. 4.1, we have

$$\lambda X.\, \alpha \circ \widetilde{pre}_N[t] \circ \gamma(X) = \begin{cases} \mathbb{N}^{k_A} & \text{if } \exists i \in [1..k_A]\colon\ |C_i| > 1 \wedge \widehat{\mathcal{I}}(i,t) > 0 \\ \lambda X.\, \widetilde{pre}_{\hat{N}}[t](X) & \text{otherwise.} \end{cases}$$

Proof:

$$\begin{aligned}
&\alpha \circ \widetilde{pre}_N[t] \circ \gamma(S) \\
&= \alpha \circ \widetilde{pre}_N[t](\{m \in \mathbb{N}^k \mid m \in \gamma(S)\}) \\
&= \alpha(\{m \mid (\mathcal{I}(t) \not\leqslant m) \vee (\mathcal{I}(t) \leqslant m \wedge m - \mathcal{I}(t) + \mathcal{O}(t) \in \gamma(S))\}) && \text{def. of } \widetilde{pre}_N[t] \\
&= \alpha(\{m \mid \mathcal{I}(t) \not\leqslant m\}) \cup \alpha(\{m \mid \mathcal{I}(t) \leqslant m \wedge m - \mathcal{I}(t) + \mathcal{O}(t) \in \gamma(S)\}) && \text{additivity of } \alpha \\
&= \alpha(\{m \mid \mathcal{I}(t) \not\leqslant m\}) \cup \alpha \circ pre_N[t] \circ \gamma(S) && \text{def. of } pre_N[t] \\
&= \alpha(\{m \mid \mathcal{I}(t) \not\leqslant m\}) \cup pre_{\hat{N}}[t](S) && \text{by Lem. 4.1}
\end{aligned}$$

We now consider two cases:

- $\exists i \in [1..k]\colon \mathcal{I}(i,t) > 0$ and $\{i\} \notin A$. Lemma 4.2 shows that $\alpha \circ \widetilde{pre}_N[t] \circ \gamma(S) = \mathbb{N}^{k_A}$;
- $\forall i \in [1..k]\colon \mathcal{I}(i,t) > 0$ implies $\{i\} \in A$. In this case we have

$$\begin{aligned}
\alpha \circ \widetilde{pre}_N[t] \circ \gamma(S) &= \{m \in \mathbb{N}^{k_A} \mid \widehat{\mathcal{I}}(t) \not\leqslant m\} \cup pre_{\hat{N}}[t](S) && \text{by Lem. 4.3} \\
&= \widetilde{pre}_{\hat{N}}[t](S) && \text{def. of } \widetilde{pre}_{\hat{N}}[t]
\end{aligned}$$

□

Definition 4.2. Given a Petri Net $N = (P, T, F, m_0)$, a partition $A = \{C_j\}_{j \in [1..k_A]}$ of $[1..k]$ and $t \in T$ we denote by ϕ_t^A the following formula: $\exists i \in [1..k_A]\colon\ |C_i| > 1 \wedge \widehat{\mathcal{I}}(i,t) > 0$. We also use T_A to denote the set of transitions $\{t \in T \mid \phi_t^A \text{ is false}\}$. □

From the result of Prop. 4.2, let us see how to approximate $\widetilde{pre}_N^*(S)$ for $S \subseteq \mathbb{N}^k$ without evaluating γ explicitly. To this end we recall the result of (3), p. 8 which is based on the best abstract counterpart of $\widetilde{pre}_N$ and states that: $\alpha(\widetilde{pre}_N^*(S)) \subseteq \mathit{gfp}(\lambda X.\, \alpha(S \cap \widetilde{pre}_N \circ \gamma(X)))$. This latter fixpoint is in turn approximated as follows:

$$\begin{aligned}
&\mathit{gfp}\lambda X.\, \alpha(S \cap \widetilde{pre}_N \circ \gamma(X)) \\
&= \mathit{gfp}\lambda X.\, \alpha(S) \cap \alpha \circ \widetilde{pre}_N \circ \gamma(X) && \text{Lem. 3.5 provided } \gamma \circ \alpha(S) = S^{2} \\
&= \mathit{gfp}\lambda X.\, \alpha(S) \cap \alpha \circ \bigcap_{t \in T} \widetilde{pre}_N[t] \circ \gamma(X) && \text{def. of } \widetilde{pre}_N \\
&\subseteq \mathit{gfp}\lambda X.\, \alpha(S) \cap \bigcap_{t \in T} \alpha \circ \widetilde{pre}_N[t] \circ \gamma(X) && \alpha(A \cap B) \subseteq \alpha(A) \cap \alpha(B) \\
&= \mathit{gfp}\lambda X.\, \alpha(S) \cap \widetilde{pre}_{\hat{N}}[T_A](X) && \text{Prop. 4.2, def. of } T_A
\end{aligned}$$

Note that if $T_A = \emptyset$ then $\mathit{gfp}\lambda X.\, S \cap \widetilde{pre}_{\hat{N}}[T_A](X)$ is defined to be S.

We see that this result for the backward semantics is weaker then what we obtained for the forward semantics. This difference stems from the fact that α is not *co-additive* (i.e. $\alpha(A \cap B) \neq \alpha(A) \cap \alpha(B)$).

[2] As we will see later, this hypothesis is going verified in the context of our algorithm.

5. The Algorithm

The algorithm we propose is given in Alg. 1. In that algorithm, we use the following notations: α_i and γ_i instead of α_{A_i} and γ_{A_i}, and given a set $S \subseteq \mathbb{N}^k$, we denote by $\lceil S \rceil$ the partition of $[1..k]$ given by $(\curlyvee\{A \mid \gamma_A \circ \alpha_A(S) = S\})$, i.e. the coarsest partition that allows to represent exactly S. Given a Petri net N and a $\leqslant$-dc-set S, the algorithm builds abstract nets $\hat{N}$ with smaller dimensionality than N (line 4), analyses them (lines 5–10), and refines them (line 13) until it concludes. To analyse an abstraction $\hat{N}$, the algorithm first uses a model-checker that answers the coverability problem for $\hat{N}$ and the $\leqslant$-dc-set $\alpha_i(S)$ using any algorithm proposed in [22, 18, 16]. Besides an answer those algorithms return an overapproximation of the fixpoint $post^*_{\hat{N}}(\widehat{m_0})$ that satisfies A1–4. If the model-checker returns a positive answer then, following the abstract interpretation theory, Alg. 1 concludes that $post^*_N(m_0) \subseteq S$ (line 6). Otherwise, Alg. 1 tries to decide if $\{m_0\} \not\subseteq \widetilde{pre}^*_N(S)$ checking the inclusion given by (2) (line 9–13). The fixpoint of (2) is computable by [1] but practically difficult to build for the net N and S. Hence, our algorithm only builds an overapproximation by evaluating the fixpoint on the abstract net $\hat{N}$ instead of N, i.e. we evaluate the fixpoint $gfp\lambda X.\, \alpha_i(S) \cap \widetilde{pre}_{\hat{N}}[T_A](X)$ which overapproximates $\alpha\big(gfp\lambda X.\, S \cap \widetilde{pre}_N(X)\big)$ by (3), p. 8. Since the abstractions $\hat{N}$ have a smaller dimensionality than N, the greatest fixpoint can be evaluated more efficiently on $\hat{N}$. Moreover, at the ith iteration of the algorithm (i) we restrict the fixpoint to the overapproximation $\mathcal{R}_i$ of $post^*_{\hat{N}}(\alpha_i(m_0))$ computed at line 5, and (ii) we consider $\alpha_i(Z_i)$ instead of $\alpha_i(S)$. Point (i) allows the algorithm to use the information given by the forward analysis of the model-checker, and point (ii) is motivated by the fact that at each step i we have $gfp\lambda X.\, \alpha_i(S) \cap \mathcal{R}_i \cap \widetilde{pre}_{\hat{N}}[T_A](X) \subseteq \alpha_i(Z_i) \subseteq \alpha_i(S)$. That allows us to consider $\alpha_i(Z_i)$ instead of $\alpha_i(S)$ without changing the fixpoint, leading to a more efficient computation of it (see [1] for more details). Those optimisations are safe in the sense that if the fixpoint we evaluate at line 9 does not contain $\alpha_i(m_0)$ then $post^*_N(m_0) \not\subseteq S$, hence its usefulness to detect negative instances (line 10).

If the algorithm cannot conclude, it refines the abstraction. The main property of the refinement is that the sequences of $Z_i's$ computed at line 12 is strictly decreasing and converge in a finite number of steps to $\widetilde{pre}^*_N(S) \cap \mathcal{R}$ where $\mathcal{R}$ is an inductive overapproximation of $post^*_N(m_0)$. Suppose that at step i, we have $Z_{i+1} = \widetilde{pre}^*_N(S) \cap \mathcal{R}$. Hence, $\gamma_{i+1} \circ \alpha_{i+1}(\widetilde{pre}^*_N(S) \cap \mathcal{R}) = \widetilde{pre}^*_N(S) \cap \mathcal{R}$. If $post^*_N(m_0) \subseteq S$ then $post^*_N(m_0) \subseteq \widetilde{pre}^*_N(S) \cap \mathcal{R}$ and the abstract interpretation theory guarantees that $post^*_{\hat{N}}(\alpha_{i+1}(m_0)) \subseteq \alpha_{i+1}(\widetilde{pre}^*_N(S) \cap \mathcal{R}) \subseteq \alpha_{i+1}(S)$, hence the Checker will return the answer OK at iteration $i+1$. Moreover, if $post^*_N(m_0) \not\subseteq S$ then $m_0 \notin \widetilde{pre}^*_N(S)$, hence $m_0 \notin \widetilde{pre}^*_N(S) \cap \mathcal{R}$, and the algorithm will return KO at step $i+1$ because we have $Z_{i+1} = \widetilde{pre}^*_N(S) \cap \mathcal{R}$, hence $\widehat{m_0} \notin \alpha_{i+1}(Z_{i+1})$ by monotonicity of α_{i+1} and Z_{i+1} does not include $\widehat{m_0}$. Again, we do not evaluate the greatest fixpoint $\widetilde{pre}^*_N(S)$ because the dimensionality of N is too high and the evaluation is in general too costly in practice. Hence, we prefer to build overapproximations that can be computed more efficiently.

We now formally prove that our algorithm is sound, complete and terminates.

Lemma 5.1. In Alg. 1, at the ith iteration, we have $\gamma_i \circ \alpha_i(Z_i) = Z_i$.

Proof:
We prove, by induction on i, that $Z_i \in \gamma_i(\wp(\mathbb{N}^{k_i}))$ where k_i is the number of elements in the partition A_i, hence that there exists $a \subseteq \wp(\mathbb{N}^{k_i})$ such that $\gamma_i(a) = Z_i$, and finally that $\gamma_i \circ \alpha_i(Z_i) = Z_i$ by property of Galois insertion. For the base case, line 1 ($Z_0 = S$) and assumption $S \in \gamma_0(\wp(\mathbb{N}^{k_0}))$ show that $Z_0 \in \gamma_0(\wp(\mathbb{N}^{k_0}))$. The inductive case follows immediately by line 13. □

Lemma 5.2. In Alg. 1, at the ith iteration, we have

$$Z_{i+1} \subseteq \gamma_i(\mathcal{S}_i) \subseteq Z_i \subseteq \cdots \subseteq Z_1 \subseteq \gamma_i(\mathcal{S}_0) \subseteq Z_0 \subseteq S \ .$$

Algorithm 1: Algorithm for the coverability problem, assume $\{m_0\} \subseteq S$

```
   Data: A Petri net $N = (P, T, F, m_0)$, a $\preccurlyeq$-dc-set $S$
 1 $Z_0 = S$
 2 $A_0 = \lceil Z_0 \rceil$
 3 for $i = 0, 1, 2, 3, \ldots$ do
 4     $\boxed{\textbf{Abstract:}}$ Given $A_i$, compute $\hat{N}$ given by def. 4.1.
 5     $\boxed{\textbf{Verify:}}$ $(answer, \mathcal{R}_i) = \mathsf{Checker}(\widehat{m_0}, post_{\hat{N}}, \alpha_i(Z_i))$
 6     if $answer == OK$ then
 7         return $OK$
 8     else
 9         Let $\mathcal{S}_i = gfp\lambda X. \alpha_i(Z_i) \cap \mathcal{R}_i \cap \widetilde{pre}_{\hat{N}}[T_{A_i}](X)$
10         if $\widehat{m_0} \notin \mathcal{S}_i$ then  return $KO$
11     end
12     Let $Z_{i+1} = \gamma_i(\mathcal{S}_i) \cap \widetilde{pre}_N(\gamma_i(\mathcal{S}_i))$
13     $\boxed{\textbf{Refine:}}$ Let $A_{i+1} = A_i \curlywedge \lceil Z_{i+1} \rceil$
14 end
```

Proof:
We establish the inclusions from right to left. First, line 1 shows that $Z_0 = S$ which concludes the first case. Then we show that for each value of i we have $\gamma_i(\mathcal{S}_i) \subseteq Z_i$ and finally $Z_{i+1} \subseteq \gamma_i(\mathcal{S}_i)$.

We conclude from line 9 that $\mathcal{S}_i \subseteq \alpha_i(Z_i)$, hence that $\gamma_i(\mathcal{S}_i) \subseteq \gamma_i \circ \alpha_i(Z_i)$ by monotonicity of γ_i and finally that $\gamma_i(\mathcal{S}_i) \subseteq Z_i$ by Lem. 5.1. Finally, it is clear from line 12 that $Z_{i+1} \subseteq \gamma_i(\mathcal{S}_i)$. □

Lemma 5.3. In Alg. 1, at the ith iteration, we have $post_N^*(m_0) \subseteq \gamma_i(\mathcal{R}_i)$.

Proof:
Eq. (3), p. 8 shows that $\alpha_i(post_N^*(m_0)) \subseteq lfp\lambda X. \alpha_i(m_0 \cup post_N(\gamma_i(X)))$, hence that $\alpha_i(post_N^*(m_0)) \subseteq post_{\hat{N}}^*(\widehat{m_0})$ by Prop. 4.1, and finally that $\alpha_i(post_N^*(m_0)) \subseteq \mathcal{R}_i$ by (A_1). The last step concludes, by $\underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}}$, that $post_N^*(m_0) \subseteq \gamma_i(\mathcal{R}_i)$. □

Proposition 5.1. (Soundness)
If Alg. 1 says "OK" then we have $post^*(m_0) \subseteq S$.

Proof:
If Alg. 1 says "OK" then

$$
\begin{aligned}
\mathcal{R}_i \subseteq \alpha_i(Z_i) &\Rightarrow \gamma_i(\mathcal{R}_i) \subseteq \gamma_i \circ \alpha_i(Z_i) && \gamma_i \text{ is monotonic} \\
&\Rightarrow \gamma_i(\mathcal{R}_i) \subseteq Z_i && \text{Lem. 5.1} \\
&\Rightarrow \gamma_i(\mathcal{R}_i) \subseteq S && \text{Lem. 5.2} \\
&\Rightarrow post_N^*(m_0) \subseteq S && \text{Lem. 5.3}
\end{aligned}
$$

□

We need the intermediary result of Lem. 5.4 to establish the completeness of Alg. 1.

Lemma 5.4. In Alg. 1 if at some iteration i we have $post_N^*(m_0) \not\subseteq Z_i$ then $post_N^*(m_0) \not\subseteq S$.

Proof:
The proof is by induction on i.

base case. Trivial since line 1 defines Z_0 to be S.

inductive case.

$$\begin{array}{ll}
\mathit{post}^*_N(m_0) \not\subseteq Z_{i+1} & \\
\Leftrightarrow \mathit{post}^*_N(m_0) \not\subseteq \gamma_i(\mathcal{S}_i) \cap \widetilde{\mathit{pre}}_N(\gamma_i(\mathcal{S}_i)) & \text{def. of } Z_{i+1} \\
\Rightarrow \mathit{post}^*_N(m_0) \not\subseteq \widetilde{\mathit{pre}}^*_N(\gamma_i(\mathcal{S}_i)) & \widetilde{\mathit{pre}}^*_N(\gamma_i(\mathcal{S}_i)) \subseteq \gamma_i(\mathcal{S}_i) \cap \widetilde{\mathit{pre}}_N(\gamma_i(\mathcal{S}_i)) \\
\Rightarrow \mathit{post}^*_N(m_0) \not\subseteq \gamma_i(\mathcal{S}_i) & \widetilde{\mathit{pre}}^*_N(Y)\text{: the markings stuck in } Y \\
\Leftrightarrow \mathit{post}^*_N(m_0) \not\subseteq \gamma_i\big(\mathit{gfp}\lambda X.\, \alpha_i(Z_i) \cap \mathcal{R}_i \cap \widetilde{\mathit{pre}}_{\hat{N}}[T_A](X)\big) & \text{def. of } \mathcal{S}_i \\
\Rightarrow \mathit{post}^*_N(m_0) \not\subseteq \mathit{gfp}\lambda X.\, Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{\mathit{pre}}_N(X) & \text{(4), p. 8} \\
\Leftrightarrow \mathit{post}^*_N(m_0) \not\subseteq \widetilde{\mathit{pre}}^*_N(Z_i \cap \gamma_i(\mathcal{R}_i)) & \text{def. of } \widetilde{\mathit{pre}}^*_N \\
\Rightarrow \mathit{post}^*_N(m_0) \not\subseteq Z_i \cap \gamma_i(\mathcal{R}_i) & \widetilde{\mathit{pre}}^*_N(Y)\text{: the markings stuck in } Y \\
\Rightarrow \mathit{post}^*_N(m_0) \not\subseteq Z_i & \text{Lem. 5.3} \\
\Rightarrow \mathit{post}^*_N(m_0) \not\subseteq S & \text{induction hypothesis}
\end{array}$$

□

Proposition 5.2. (Completeness)
If Alg. 1 says “KO” then we have $\mathit{post}^*_N(m_0) \subseteq S$.

Proof:
If Alg. 1 says “KO” then

$$\begin{array}{ll}
\widehat{m_0} \notin \mathcal{S}_i \Leftrightarrow \alpha_i(m_0) \not\subseteq \mathcal{S}_i & \text{def. of } \widehat{m_0} \\
\qquad \Rightarrow \alpha_i(m_0) \not\subseteq \alpha_i(\mathit{gfp}\lambda X.\, Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{\mathit{pre}}_N(X)) & \text{(3)} \\
\qquad \Rightarrow m_0 \notin \mathit{gfp}\lambda X.\, Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{\mathit{pre}}_N(X) & \alpha_i \text{ monotonicity} \\
\qquad \Leftrightarrow \mathit{lfp}\lambda X.\, m_0 \cup \mathit{post}(X) \not\subseteq Z_i \cap \gamma_i(\mathcal{R}_i) & \text{(1) iff (2)} \\
\qquad \Rightarrow \mathit{lfp}\lambda X.\, m_0 \cup \mathit{post}(X) \not\subseteq Z_i & \text{Lem. 5.3} \\
\qquad \Rightarrow \mathit{lfp}\lambda X.\, m_0 \cup \mathit{post}(X) \not\subseteq S & \text{Lem. 5.4}
\end{array}$$

□

Proposition 5.3. (Termination)
Algorithm 1 terminates.

Proof:
We start by recalling the result of Lem. 5.2 which shows that

$$Z_0 \supseteq Z_1 \supseteq \cdots \supseteq Z_i \supseteq Z_{i+1} \supseteq \cdots$$

Consider the sequence of Z_i’s and assume that from index i we have $Z_{i+1} = Z_i$.

$$\begin{aligned}
&Z_{i+1} = \gamma_i(\mathcal{S}_i) \cap \widetilde{pre}_N(\gamma_i(\mathcal{S}_i)) && \text{def. of } Z_{i+1}\\
&\Rightarrow Z_{i+1} \subseteq \widetilde{pre}_N(\gamma_i(\mathcal{S}_i)) && \\
&\Rightarrow Z_{i+1} \subseteq \widetilde{pre}_N(Z_i) && \text{Lem. 5.2, } \widetilde{pre}_N \text{ monotonicity}\\
&\Leftrightarrow Z_{i+1} \subseteq \widetilde{pre}_N(Z_{i+1}) && Z_{i+1} = Z_i\\
&\Leftrightarrow Z_j \subseteq \widetilde{pre}_N(Z_j) && \text{let } j = i+1\\
&\Leftrightarrow post_N(Z_j) \subseteq Z_j && \text{(Gc)}\\
&\Rightarrow \alpha_j \circ post_N(Z_j) \subseteq \alpha_j(Z_j) && \alpha_j \text{ is monotonic}\\
&\Rightarrow \alpha_j \circ post_N(\gamma_j \circ \alpha_j(Z_j)) \subseteq \alpha_j(Z_j) && \text{Lem. 5.1}\\
&\Rightarrow post_{\hat{N}}(\alpha_j(Z_j)) \subseteq \alpha_j(Z_j) && \text{Prop. 4.1}
\end{aligned}$$

Then, provided $\widehat{m_0} \in \alpha_j(Z_j)$ holds, so does $lfp\lambda X.\, \widehat{m_0} \cup post_{\hat{N}}(X) \subseteq \alpha_j(Z_j)$ and line 6 shows that the Alg. 1 terminates by property A4 of the Checker called at line 5. Or we have, $\widehat{m_0} \notin \alpha_j(Z_j)$ which yields $\widehat{m_0} \notin \mathcal{S}_j$ as follows. We conclude from Lem. 5.2 that $\gamma_j(\mathcal{S}_j) \subseteq Z_j$, hence that $\alpha_j \circ \gamma_j(\mathcal{S}_j) \subseteq \alpha_j(Z_j)$ by monotonicity of α_j and finally that $\mathcal{S}_j \subseteq \alpha_j(Z_j)$ since $\alpha_j \circ \gamma_j = \lambda x.\, x$ by Galois insertion. To conclude this part of the proof, we see that the test of line 10 succeeds and Alg. 1 terminates.

Now we assume that the sequence of Z_i's strictly decreases, i.e. $Z_{i+1} \subset Z_i$. First recall that the ordered set $\langle DCS(\mathbb{N}^k), \subseteq\rangle$ is a wqo. We conclude from A2, Lem. 3.5, $\preccurlyeq$-dc-set are closed to $\widetilde{pre}$ and $\cap$ that for each value of i in Alg. 1 we have $Z_i \in DCS(\mathbb{N}^k)$. However $\preccurlyeq$ defines a wqo and following [16, Lem. 2] there is no infinite strictly decreasing sequence of $\langle DCS(\mathbb{N}^k), \subseteq\rangle$, hence a contradiction. □

6. Efficient implementation

We implemented Alg. 1 with the checker of [18] in a naïve way and presented experimental results in [17]. Those results showed that our algorithm is able to build Petri nets $\hat{N}$ that are small but precise enough to conclude. Hence, preliminary experiments validated our new approach. However, the resources (time/memory) needed by our prototype was huge for large Petri nets. In particular the refinement step (line 13) and the $\widetilde{pre}_N$ operator are not efficient. In the sequel, we present improvements that dramatically decrease the resources used by our algorithm. We assume from now that Checker does not take in input a $\preccurlyeq$-dc-set S but its complement[3]. To simplify the correction proof of our improved algorithm, we also assume the following for every Petri net $N = (P, T, F, m_0)$ and every $\preccurlyeq$-uc-sets U and I such that $post^*_N(m_0) \cap U \neq \emptyset$ and $post^*_N(m_0) \cap I = \emptyset$: The overapproximations $\mathcal{R}_1$ and $\mathcal{R}_2$ returned by Checker when applied, resp., to the target sets U and $U \cup I$ are such that

$$\mathcal{R}_1 = \mathcal{R}_2 \ . \tag{A5}$$

This last assumption is verified by most of the model checking algorithms [22, 18, 19].

For the same reason, we will also assume that Checker returns refined results over refined domains: if $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$ are computed by Algorithm 1, then we have that

$$\gamma_i(\mathcal{R}_i) \subseteq \gamma_{i+1}(\mathcal{R}_{i+1}) \ . \tag{A6}$$

If A6 is not enforced by Checker itself then it can be enforced at each iteration i (for $i \geq 1$) by defining $\mathcal{R}_i$ to be the intersection of the set returned by Checker$(\widehat{m_0}, post_{\hat{N}}, \alpha_i(Z_i))$ with $\alpha_i \circ \gamma_{i-1}(\mathcal{R}_{i-1})$.

[3] Notice that it is in general the case in practice.

Algorithm 2: solves the coverability problem without $\widetilde{pre}$ and complement, assume $\{m_0\} \subseteq S$

```
Data: A Petri net $N = (P, T, F, m_0)$, a $\preccurlyeq$-dc-set $S$
$Z'_0 = \neg S$
$A'_0 = \lceil Z'_0 \rceil$
for $i = 0, 1, 2, 3, \ldots$ do
    $\boxed{\textbf{Abstract:}}$ Given $A'_i$, compute $\hat{N}'$ given by def. 4.1.
    $\boxed{\textbf{Verify:}}$ $(\mathit{answer}, \mathcal{R}'_i) = \mathsf{Checker}(\widehat{m_0}, \mathit{post}_{\hat{N}'}, \alpha'_i(Z'_i))$
    if $\mathit{answer} == \mathit{OK}$ then
        return $\mathit{OK}$
    else
        Let $\mathcal{S}'_i = \mathit{lfp}\lambda X.\, {\uparrow}\big((\alpha'_i(Z'_i) \cup \mathit{pre}_{\hat{N}'}[T_{A_i}](X)\big) \cap \mathcal{R}'_i\big)$
        if $\widehat{m_0} \in \mathcal{S}'_i$ then return $\mathit{KO}$
    end
    Let $Z'_{i+1} = {\uparrow}((\gamma'_i(\mathcal{S}'_i) \cup \mathit{pre}_N(\gamma'_i(\mathcal{S}'_i))) \cap \gamma'_i(\mathcal{R}'_i))$
    $\boxed{\textbf{Refine:}}$ Let $A'_{i+1} = A'_i \curlywedge \lceil Z'_{i+1} \rceil$
end
```

6.1. Replacing the greatest fixpoint by a least fixpoint

In practice, the abstract greatest fixpoint $\mathcal{S}_i$ given by $\mathit{gfp}\lambda X.\, \alpha_i(Z_i) \cap \mathcal{R}_i \cap \widetilde{pre}_{\hat{N}}[T_{A_i}](X)$ is costly to compute, and, in particular, the operator $\widetilde{pre}$. To avoid that problem, we can compute its complement instead. Following the same idea, we could avoid to compute the set Z_{i+1} but computing its complement instead. This is equivalent since Z_{i+1} can be represented exactly with an abstraction if and only if its complement can be represented exactly with the same abstraction as shown in Lem. 3.6. For the sake of clarity, the proofs of the subsection are given in appendix.

We first give a characterization of the complement of the sets $\mathcal{S}_i$ and Z_i that uses pre instead of $\widetilde{pre}$.

Proposition 6.1. In Alg. 1, at the ith iteration, we have

$$\neg\mathcal{S}_i = \mathit{lfp}\lambda X.\, \alpha_i(\neg Z_i) \cup \neg\mathcal{R}_i \cup \mathit{pre}_{\hat{N}}[T_{A_i}](X) \quad \text{and} \quad \neg Z_i = \neg\gamma_i(\mathcal{S}_i) \cup \mathit{pre}_N(\neg\gamma_i(\mathcal{S}_i)) \ .$$

The formulation of the set $\neg\mathcal{S}_i$ given by Prop. 6.1 contains the negation of the set $\mathcal{R}_i$ that may be costly to compute in practice. Hence, that formulation is not satisfactory to design an efficient algorithm. Since the set $\mathcal{R}_i$ is computed at each iteration, we propose to take implicitly into account the set $\neg\mathcal{R}_i$ when computing $\neg\mathcal{S}_i$. In other words, we do not compute $\neg\mathcal{S}_i$ but a set $\mathcal{S}'_i$ such that $\neg\mathcal{S}_i = \mathcal{S}'_i \cup \neg\mathcal{R}_i$.

Algorithm 2 computes such sets without using $\widetilde{pre}$ and complement. In that algorithm, we use the notation α'_i and γ'_i instead of $\alpha_{A'_i}$ and $\gamma_{A'_i}$. Instead of computing the sets $\mathcal{S}_i$ and Z_i, the algorithm computes, respectively, the sets $\mathcal{S}'_i$ and Z'_i. Intuitively, the sets $\mathcal{S}'_i$ are ($\preccurlyeq$-upward closed) over-approximations of the set of markings of the Petri net $\hat{N}'$ that are in $\mathcal{R}_i$ allowing to reach $\alpha'_i(Z'_i)$. To prove its correction, we first prove Lem. 6.1 that states the relation between the sets computed by Alg. 1 and Alg. 2.

Lemma 6.1. Let N be a Petri net given by (P, T, F, m_0) and a $\preccurlyeq$-dc-set $S \subseteq \mathbb{N}^k$. Assume that Algorithm 1 and 2 compute the sets $\mathcal{S}_i, \mathcal{S}'_i, Z_{i+1}, Z'_{i+1}$ and for all $0 \leq j \leq i, A_j = A'_j$. Then, we have that $\neg\mathcal{S}_i = \mathcal{S}'_i \cup \neg\mathcal{R}_i$ and $\neg Z_{i+1} = Z'_{i+1} \cup \gamma_i(\neg\mathcal{R}_i)$.

The next lemma validates the assumption that the abstract domains computed by the two algorithms are the same.

Lemma 6.2. Let a Petri net $N = (P, T, F, m_0)$ and a $\leqslant$-dc-set $S \subseteq \mathbb{N}^k$. Assume that Algorithm 1 and 2 compute at least i iterations. Then, we have that $A_i = A'_i$.

We are now ready to prove that Algorithm 1 and Algorithm 2 returns the same answer for all the instances of the coverability problem.

Theorem 6.1. For any Petri net $N = (P, T, F, m_0)$ and $\leqslant$-dc-set $S \subseteq \mathbb{N}^k$ the following holds:

1. If Alg. 1 answers OK at iteration i then so does Alg. 2;
2. If Alg. 1 answers KO at iteration i then so does Alg. 2.

Finally, note that termination of Algorithm 2 is directly implied by Theorem 6.1.

6.2. Efficient computation of a new abstraction

In this section we give efficient algorithms to implement the operation needed at lines 2 and 13. Indeed, the naïve approach to build $A'_i (i \geq 0)$ enumerates the set of all the partitions of P, which is exponential in the number of places.

Instead, we present the algorithm refinement that given a set of markings M computes the coarsest partition A which is able to represent exactly M without enumerating the partitions. The algorithm starts from the $\preceq$-minimal partition then the algorithm chooses non-deterministically two candidate classes and merge them in a unique class. If this new partition still represents exactly M, we iterate the procedure. Otherwise the algorithm tries choosing different candidates. The algorithm is presented in Alg. 3 in which we use the following notation: given a partition $A = \{C_i\}_{i \in [1..k_A]}$ of $[1..k]$, we denote by A_{C_i} the partition $\{C_i\} \cup \{\{s\} \mid s \in [1..k] \wedge s \notin C_i\}$.

Algorithm 3: refinement

```
Input: $M \subseteq \mathbb{N}^k$
Let $A$ be $\{\{1\}, \{2\}, \dots, \{k\}\}$ ;
while $\exists C_i, C_j \in A : C_i \neq C_j$ and $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq M$ do
    Let $C_i, C_j \in A$ such that $C_i \neq C_j$ and $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq M$;
    $A \leftarrow (A \setminus \{C_i, C_j\}) \cup \{C_i \cup C_j\}$ ;
```

We first prove the following lemma.

Lemma 6.3. Let $A = \{C_i\}_{i \in [1..k_A]}$ be a partition of $[1..k]$, $M \subseteq \mathbb{N}^k$, we have:

$$\gamma_A \circ \alpha_A(M) \subseteq M \Leftrightarrow \left(\forall C_i \in A \colon \gamma_{A_{C_i}} \circ \alpha_{A_{C_i}}(M) \subseteq M\right) \ .$$

Proof:
We conclude from $A_{C_i} \preceq A$ and Lem. 3.3 that the implication $\Rightarrow$ holds. For the other direction the result follows from Lem. 3.4 and the following equality: $A = \curlyvee\{A_{C_i}\}_{C_i \in A}$ which follows by definition of $\curlyvee$. Indeed, $A_{C_i} \curlyvee A_{C_j}$ is given by $\{C_i, C_j\} \cup \{\{s\} \mid s \in [1..k] \wedge s \notin (C_i \cup C_j)\}$. □

The following two lemmas and the corollary state the correctness and the optimality of Alg. 3.

Lemma 6.4. Given $M \subseteq \mathbb{N}^k$, the partition A returned by refinement(M) is such that $\gamma_A \circ \alpha_A(M) = M$.

Proof:
Initially $A = \{\{1\}, \ldots, \{k\}\}$ so that $\gamma_A \circ \alpha_A(M) = M$, hence $\gamma_A \circ \alpha_A(M) \subseteq M$ which is an invariant maintained by the iterations. Indeed, suppose the invariant holds before executing line 1: $\gamma_A \circ \alpha_A(M) \subseteq M$. We conclude from line 1 that $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq M$, hence that the new value for A satisfies $\gamma_A \circ \alpha_A(M) \subseteq M$ by Lem. 6.3 and line 2. □

Lemma 6.5. Given $M \subseteq \mathbb{N}^k$, let A be the partition returned by refinement(M). There is no partition A' with $A \preceq A'$ and $A \neq A'$ such that $\gamma_{A'} \circ \alpha_{A'}(M) = M$.

Proof:
Suppose that such a partition A' exists. Since $A \preceq A'$, $\exists C_i, C_j \in A \, \exists C' \in A'$: $(C_i \neq C_j) \wedge C_i \cup C_j \subseteq C'$. We conclude from Lem. 6.3 and $\gamma_{A'} \circ \alpha_{A'}(M) \subseteq M$ (hence $\gamma_{A'} \circ \alpha_{A'}(M) = M$ by $\underset{\alpha_{A'}}{\overset{\gamma_{A'}}{\leftrightarrows}}$), that $\gamma_{A_{C'}} \circ \alpha_{A_{C'}}(M) = M$.

Moreover, $A_{C_i \cup C_j} \preceq A_{C'}$ shows that $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq \gamma_{A_{C'}} \circ \alpha_{A_{C'}}(M) = M$ by Lem. 3.3 and property of Galois insertions. Hence, the equality $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) = M$ holds.

It follows that the condition of the while loop of the refinement algorithm is verified by A, hence the algorithm should execute the loop at least once more before termination and return a partition A'' such that $A \preceq A''$ and $A \neq A''$, a contradiction. □

Putting together Prop. 3.2 and Lem. 6.4 and 6.5 we get:

Corollary 6.1. Given $M \subseteq \mathbb{N}^k$, the partition A returned by refinement(M) is the coarsest partition representing exactly M, that is refinement(M)$= \lceil M \rceil$.

In the algorithm refinement, the termination test of the **while** loop asks if $\gamma_{C_i \cup C_j} \circ \alpha_{C_i \cup C_j}(M) \subseteq M$ holds. As shown in Alg. 2, the sets of markings involved at lines 2 and 13 are $\leqslant$-uc-sets of markings which contains infinitely many markings. Hence the test is to decide whether $\gamma_{C_i \cup C_j} \circ \alpha_{C_i \cup C_j}(M) \subseteq M$ is far from being trivial. However, by taking into account the particular structure of $\leqslant$-uc-sets, we restrict the reasoning to the minimal elements of M only. Now, we can prove that a $\leqslant$-uc-set U is exactly representable with an abstraction iff $\min(U)$ is exactly representable with that abstraction.

Lemma 6.6. Let $A \in \mathcal{A}^{k_A \times k}$, $U \in UCS(\mathbb{N}^k)$ and $\min(U)$ be the finite canonical minor set of U. We have that $\gamma_A \circ \alpha_A(U) \subseteq U$ if and only if $\gamma_A \circ \alpha_A(min(U)) \subseteq min(U)$.

Proof:
$\boxed{\Rightarrow}$ In order to ease its understanding, the following proof is given together with Fig. 4.

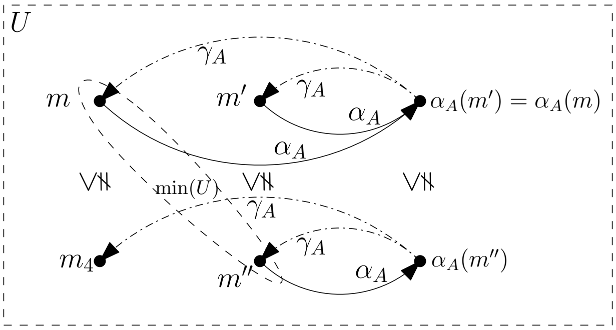

Figure 4. Graphical depiction of the reasoning given in the proof of Lem. 6.6.

$$\begin{aligned}
&\gamma_A \circ \alpha_A(\min(U)) \not\subseteq \min(U) \\
&\Rightarrow \exists m\, \exists m' \colon m \in \min(U) \wedge m' \notin \min(U) \wedge m' \in \gamma_A \circ \alpha_A(\{m\}) \\
&\Rightarrow \exists m\, \exists m' \colon m \in \min(U) \wedge m' \in U \setminus \min(U) \wedge m' \in \gamma_A \circ \alpha_A(\{m\}) \qquad \gamma_A \circ \alpha_A(U) \subseteq U
\end{aligned} \tag{9}$$

Consider any such markings m, m' satisfying (9) as depicted in Fig. 4. Using the fact that U is a $\preccurlyeq$-uc-set we deduce that

$$\begin{aligned}
&\exists m'' \colon m'' \in \min(U) \wedge m'' \lneqq m' \\
&\Rightarrow \exists m'' \colon m'' \in \min(U) \wedge \alpha_A(m'') \lneqq \alpha_A(m') \\
&\qquad\qquad\qquad\qquad \text{Lem. 3.5} \\
&\Rightarrow \exists m'' \colon m'' \in \min(U) \wedge \alpha_A(m'') \lneqq \alpha_A(m') \wedge \exists m_4 \colon m_4 \lneqq m \\
&\qquad\qquad\qquad\qquad \text{Lem. 3.5 with } m_4 \in \gamma_A \circ \alpha_A(m'') \\
&\Rightarrow \exists m'' \colon m'' \in \min(U) \wedge \alpha_A(m'') \lneqq \alpha_A(m') \wedge \exists m_4 \colon m_4 \lneqq m \wedge m_4 \in U \\
&\qquad\qquad\qquad\qquad \gamma_A \circ \alpha_A(U) \subseteq U
\end{aligned}$$

Hence a contradiction since $m \in \min(U) \wedge m_4 \lneqq m \wedge m_4 \in U$.

$\boxed{\Leftarrow}$

$$\begin{aligned}
U &= \uparrow\! min(U) \\
&\supseteq \uparrow\! \gamma_A \circ \alpha_A(min(U)) && \text{by hyp., monotonicity of } \uparrow \\
&= \gamma_A \circ \alpha_A(\uparrow\! min(U)) && \text{Lem. 3.5}
\end{aligned}$$

Since $\uparrow\! min(U) = U$, we conclude that $\gamma_A \circ \alpha_A(U) \subseteq U$. □

From Lem. 6.6 and 6.3, we concentrate on the problem to decide if $\gamma_{A_S} \circ \alpha_{A_S}(M) \subseteq M$ where M is a finite set of marking and S a finite set of places.

It is important to note that a direct implementation of the test $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq M$ is not polynomial time because the number of elements of the set $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M)$ may be exponential with respect to the number of elements of M. To overcome that problem, we define a polynomial algorithm to test if $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq M$ without computing explicitly the set $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M)$.

Let us assume in the rest of the section a finite set M of markings. Given a marking m over the set P of places and a set $S \subseteq P$, we define $m(S)$ to be $\Sigma_{p \in S} m(p)$ and $m_{/S}$ to be the marking over the places $P \setminus S$ such that $m_{/S}(p) = m(p)$ for all $p \in P \setminus S$.

Alg. 4 decides if $\gamma_{A_S} \circ \alpha_{A_S}(M) \subseteq M$. It proceeds by enumerating the markings of M, and for each marking m it compares the number of markings of M which contains as many tokens in S as does m to $\binom{m(S)+|S|-1}{|S|-1}$, the number of possible distributions of $m(S)$ tokens into $\mid S \mid$ places, i.e. the number of markings in $\gamma_{A_S} \circ \alpha_{A_S}(\{m\})$. If the two values are not equal for some marking $m \in M$, then we conclude that $\gamma_{A_S} \circ \alpha_{A_S}(M) \not\subseteq M$.

We define the size of the set M, noted $\mathsf{size}(M)$, given as parameter of Alg. 4 as the sum of: the number of elements in M, the number k of places, and the number of bits needed to encode the maximal value, denoted max, appearing into a markings of M. Let us note that the test at line 4 can be evaluated in time polynomial in $\mathsf{size}(M)$. Indeed, let us consider the algorithm given in [23] to compute $\binom{m(S)+|S|-1}{|S|-1}$. That algorithm computes $n = min(|S|-1, m(S))$ multiplications and n divisions. Hence, the number of operations computed by the algorithm is bounded by a polynomial in $\mathsf{size}(M)$ since $|S|$ is bounded by k. Remember that multiplication/division can be computed in time polynomial in the number of bits used to encode the arguments. Furthermore, the number of bits to encode the result does not increase for division, i.e. it is bounded by the number of bits of the first argument, and

```
Algorithm 4: CanIRepresentExactlyTheFiniteSet
   Input: A finite set $M \subseteq \mathbb{N}^k$ and a finite set of places $S \subseteq P$
1  while ($M \neq \emptyset$) do
2      Let $m \in M$;
3      $M' \leftarrow \{m' \in M \mid m'(S) = m(S)$ and $m'_{/S} = m_{/S}\}$;
4      if $|M'| < \binom{m(S)+|S|-1}{|S|-1}$ then return false;
5      $M \leftarrow M \setminus M'$;
6  return true;
```

the number of bits to encode the result of a multiplication is bounded by the sum of the bits needed for the arguments. Hence, the number of bits to encode intermediate results may increase when applying multiplications. However, a careful examination of the algorithm reveals it multiplies n values bounded by $m(S) + |S| - 1$ whose encoding uses at most $\lceil log_2(m(S) + |S| - 1)\rceil$ bits. Hence, the intermediate results are encoded with at most $n\lceil(log_2(m(S) + |S| - 1))\rceil$ bits. Note that $log_2(m(S) + |S| - 1) \leq log_2(k \cdot max + k) \leq log_2(k \cdot max + k \cdot max) = log_2(2k \cdot max) = 1 + log_2(k) + log_2(max)$ for all $max > 0$. Hence, $log_2(m(S) + |S| - 1)$ is bounded by a polynomial in size(M), i.e. each multiplication/division is applied on arguments whose size of the encoding is bounded by a polynomial in size(M). We conclude that there exists a polynomial in size(M) that bounds the execution time of the algorithm that computes $\binom{m(S)+|S|-1}{|S|-1}$.

Proposition 6.2. Algorithm 4 has time complexity polynomial in size(M).

Proof:
First, notice that each line of the algorithm can be computed in time polynomial in size(M). Then, from lines 2,3 and 5 we know that the number of elements in the set M decreases at each iteration of the **while** loop. Since M is a finite set at the beginning of the algorithm, we conclude that M becomes empty after a finite number of iterations bounded by size(M). Hence, the time complexity of Alg. 4 is polynomial in size(M). □

Theorem 6.2. Algorithm 4 returns false iff $\gamma_{A_S} \circ \alpha_{A_S}(M) \nsubseteq M$.

Proof:
$\boxed{\Rightarrow}$ Suppose that Alg. 4 returns false. From lines 3 and 4, we know that there exists a marking $m \in M$ such that the size of the set $\{m' \in M \mid m'(S) = m(S)$ and $m'_{/S} = m_{/S}\}$ is smaller than $\binom{m(S)+|S|-1}{|S|-1}$. This means that there exists a marking m' such that $m'(S) = m(S), m'_{/S} = m_{/S}$ and $m' \notin M$. Indeed, $\binom{m(S)+|S|-1}{|S|-1}$ is the number of possible distributions of $m(S)$ tokens into the places S. From the definition of m and m', we conclude that $m' \in \gamma_{A_S} \circ \alpha_{A_S}(M)$, hence $\gamma_{A_S} \circ \alpha_{A_S}(M) \nsubseteq M$.

$\boxed{\Leftarrow}$ Suppose that $\gamma_{A_S} \circ \alpha_{A_S}(M) \nsubseteq M$ but Alg. 4 returns true. If the algorithm returns true then, at line 4, it always considers markings m from M such that the set of markings $M_m = \{m' \in M \mid m'(S) = m(S)$ and $m'_{/S} = m_{/S}\}$ has size $s = \binom{m(S)+|S|-1}{|S|-1}$. Note that if the algorithm took another marking $m' \in M_m$ from M instead of m, $M_{m'}$ would also have size s, since $M_{m'} = M_m$, and the algorithm would also return true. We conclude that for each marking $m \in M$ we have that the set $\{m' \mid m'(S) = m(S)$ and $m'_{/S} = m_{/S}\}$ is included into M. Hence, $\gamma_{A_S} \circ \alpha_{A_S}(M) \subseteq M$, which is a contradiction with our hypothesis. □

Table 1. **Var**: number of places of the Petri net; **Cvar**: number of places of the abstraction that allow to conclude; **Ref**: number of refinements before conclusion; **time**: execution time in seconds on Intel Xeon 3Ghz; **EEC**: execution time of the algorithm defined in [18].

	Example	Var	Cvar	Ref	time	EEC
Unbounded PN	ME	5	4	1	<0.01s	<0.01s
	MultiME	12	5	1	<0.01s	<0.01s
	FMS	22	7	1	0.03s	6.3s
	CSM	14	9	2	0.02s	0.06s
	PNCSA	31	20	5	1s	2m42s
	mesh2x2	32	9	2	0.04s	17m29s
	mesh3x2	52	9	2	0.07s	>60m
Bounded PN	**Example**	**Var**	**Cvar**	**Ref**	**time**	**EEC**
	lamport	11	9	2	0.02s	<0.01s
	dekker	16	15	2	0.14s	0.03s
	peterson	14	13	2	0.06s	0.01s

7. Experimental results

We implemented Alg. 2 in `C` with the efficient refinement procedure given by Alg. 3 and 4. We used the symbolic data structure of [15] to represent and manipulate sets of markings; and for the model-checker referenced at line 5, we used the algorithm of [18].

We tested our method against a large set of examples. The properties we consider are mutual exclusions and the results we obtained are shown in Table 1. We distinguish two kind of examples. *Parametrized systems* describe systems where we have a parametrized number of resources: ME [10, Fig. 1], MultiME (Fig. 1 of Sect. 2), FMS [5], CSM [24, Fig. 76, page 154], the mesh 2x2 of [24, Fig. 130, page 256] and its extension to the 3x2 case. For all those infinite state Petri nets, the mutual exclusion properties depend only on a small part of the nets. We also considered the Petri net given in [13] encoding the PNCSA protocol in a high-level way. For that example, we do not test mutual exclusion but if a particular transition is firable from a reachable marking. Notice that the property holds, i.e. the PNCSA example is a negative instance while the other examples are positive instances.

The mesh 2x2 (resp. 3x2) examples corresponds to 4 (resp. 6) processors running in parallel with a load balancing mechanism that allow tasks to move from one processor to another. The mutual exclusion property says that one processor never processes two tasks at the same time. That property is local to one processor and our algorithm builds an abstraction where the behaviour of the processor we consider is exactly described and the other places are totally abstracted into one place. In that case, we manipulate subsets of $\mathbb{N}^9$ instead of subsets of $\mathbb{N}^{32}$ for mesh 2x2 or $\mathbb{N}^{52}$ for mesh 3x2.

For the other examples, we have a similar phenomenon: only a small part of the Petri nets is relevant to prove the mutual exclusion property. The rest of the net describes other aspects of the parametrized system and is abstracted by our algorithm. Hence, all the parametrized systems are analysed building an abstract Petri net with few places.

The bounded Petri Net examples are classical *algorithms to ensure mutual exclusion* of critical sections for two processes. In those cases, our method concludes building very precise abstractions, i.e. only few places are merged. The reasons are twofold: (i) the algorithms are completely dedicated to mutual exclusion, and (ii) the nets have been designed by hand in a "*optimal*" manner. However and quite surprisingly, we noticed that our algorithm found for those examples places that can be merged. In our opinion, this shows that our algorithm found reductions that are (too) difficult to find by hand.

Finally, concerning the execution times we see that our method is dramatically faster than the classical method that consists to directly analyse Petri nets without first decrease their size.

References

[1] Abdulla, P. A., Cerans, K., Jonsson, B., Tsay, Y.-K.: General decidability theorems for infinite-state systems, *LICS '96: Proc. 11th Annual IEEE Symp. on Logic in Computer Science*, IEEE Computer Society, 1996.

[2] Abdulla, P. A., Iyer, S. P., Nylén, A.: Unfoldings of Unbounded Petri Nets., *CAV '00: Proc. 12th Int. Conf. on Computer Aided Verification*, 1855, Springer, 2000.

[3] Berthelot, G., Roucairol, G., Valk, R.: Reductions of Nets and Parallel Prgrams, *Advanced Course: Net Theory and Applications*, 84, Springer, 1975.

[4] Burris, S., Sankappanavar, H. P.: *A Course in Universal Algebra*, Springer, New York, 1981.

[5] Ciardo, G., Miner, A.: Storage Alternatives for Large Structured State Space, *Proc. Computer Performance Evaluation 1997: 9th Int. Conf. on Modelling Techniques and Tools*, 1245, Springer, 1997.

[6] Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement for symbolic model checking, *J. ACM*, **50**(5), 2003, 752–794.

[7] Cousot, P.: Semantic Foundations of Program Analysis, in: *Program Flow Analysis: Theory and Applications* (S. Muchnick, N. Jones, Eds.), chapter 10, Prentice-Hall, Inc., 1981, 303–342.

[8] Cousot, P.: Partial Completeness of Abstract Fixpoint Checking, invited paper, *SARA '00: Proc. 4th Int. Symp. on Abstraction, Reformulations and Approximation*, 1864, Springer, 2000.

[9] Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, *POPL '77: Proc. 4th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages*, ACM Press, 1977.

[10] Delzanno, G., Raskin, J.-F., Van Begin, L.: Attacking Symbolic State Explosion, *CAV '01: Proc. 13th Int. Conf. on Computer Aided Verification*, 2102, Springer, 2001.

[11] Dickson, L. E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors, *Amer. J. Math.*, **35**, 1913, 413–422.

[12] Evangelista, S., Haddad, S., Pradat-Peyre, J.-F.: Syntactical Colored Petri Nets Reductions, *ATVA '05: Proc. 3rd IntṠympȯn Automated Technology for Verification and Analysis*, 3707, Springer, 2005.

[13] Finkel, A.: The minimal coverability graph for Petri nets, *APN '93: Proc. 14th Int. Conf. on Application and Theory of Petri Nets*, 674, Springer, 1993.

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

[15] Ganty, P., Meuter, C., Delzanno, G., Kalyon, G., Raskin, J.-F., Van Begin, L.: *Symbolic Data Structure for sets of k-uples*, Technical report, Université Libre de Bruxelles, Belgium, 2007.

[16] Ganty, P., Raskin, J.-F., Van Begin, L.: A Complete Abstract Interpretation Framework for Coverability Properties of WSTS, *VMCAI '06: Proc. 7th Int. Conf. on Verification, Model Checking and Abstract Interpretation*, 3855, Springer, 2006.

[17] Ganty, P., Raskin, J.-F., Van Begin, L.: From Many Places to Few: Automatic Abstraction Refinement for Petri Nets, *ICATPN '07: Proc. of 28th Int. Conf. on Application and Theory of Petri Nets and Other Models of Concurrency*, 4546, Springer, 2007.

[18] Geeraerts, G., Raskin, J.-F., Van Begin, L.: Expand, Enlarge and Check: new algorithms for the coverability problem of WSTS, *FSTTCS '04: Proc. 24th Int. Conf. on Fondation of Software Technology and Theoretical Computer Science*, 3328, Springer, 2004.

[19] Geeraerts, G., Raskin, J.-F., Van Begin, L.: On the efficient Computation of the Minimal Coverability set of Petri nets, *ATVA '07: Proc. 5th Int. Symp. on Automated Technology for Verification and Analysis*, 4762, Springer, 2007.

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

[21] Grahlmann, B.: The PEP Tool, *CAV'97: Proc. 9th Int. Conf. on Computer Aided Verification*, 1254, Springer, 1997.

[22] Karp, R. M., Miller, R. E.: Parallel Program Schemata., *Journal of Comput. Syst. Sci.*, **3**(2), 1969, 147–195.

[23] Manolopoulos, Y.: Binomial coefficient computation: Recursion or Iteration, *ACM SIGCSE Bulletin*, **34**(4), 2002, 65–67.

[24] Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: *Modelling with Generalized Stochastic Petri Nets*, Wiley series in parallel computing, Wiley, 1995.

[25] Park, D.: Fixpoint Induction and Proofs of Program Properties, in: *Machine Intelligence*, vol. 5, American

Elsevier, 1969, 59–78.

[26] Van Begin, L.: *Efficient Verification of Counting Abstractions for Parametric systems*, Ph.D. Thesis, Université Libre de Bruxelles, Belgium, 2003.

A. Proofs

Here follows the proof of Proposition 6.1.

Proof:

$$
\begin{aligned}
\neg\mathcal{S}_i &= \neg \mathit{gfp}\lambda X.\, \alpha_i(Z_i) \cap \mathcal{R}_i \cap \widetilde{\mathit{pre}}_{\hat{N}}[T_{A_i}](X) && \text{def. of } \mathcal{S}_i\\
&= \mathit{lfp}\lambda X.\, \neg(\alpha_i(Z_i) \cap \mathcal{R}_i \cap \widetilde{\mathit{pre}}_{\hat{N}}[T_{A_i}](\neg X)) && \text{Park's theorem [25]}\\
&= \mathit{lfp}\lambda X.\, \neg\alpha_i(Z_i) \cup \neg\mathcal{R}_i \cup \neg(\widetilde{\mathit{pre}}_{\hat{N}}[T_{A_i}](\neg X)) && \\
&= \mathit{lfp}\lambda X.\, \neg\alpha_i(Z_i) \cup \neg\mathcal{R}_i \cup \neg \circ \neg \circ \mathit{pre}_{\hat{N}}[T_{A_i}] \circ \neg \circ \neg(X) && \text{def. of } \widetilde{\mathit{pre}}\\
&= \mathit{lfp}\lambda X.\, \neg\alpha_i(Z_i) \cup \neg\mathcal{R}_i \cup \mathit{pre}_{\hat{N}}[T_{A_i}](X) && \neg \circ \neg = \lambda x.\, x\\
&= \mathit{lfp}\lambda X.\, \alpha_i(\neg Z_i) \cup \neg\mathcal{R}_i \cup \mathit{pre}_{\hat{N}}[T_{A_i}](X) && \text{Lem. 3.5, Lem. 5.1}
\end{aligned}
$$

The second assertion is obvious. □

Here follows the proof of Lemma 6.1

Proof:
The proof is by induction on i.

$\boxed{i = 0}$ First, $A_0 = A'_0$, and $\neg Z_0 = Z'_0$ implies that $\mathcal{R}_0 = \mathcal{R}'_0$, $\alpha'_0 = \alpha_0$, and $\gamma'_0 = \gamma_0$ (lines 4-5 of Alg. 1 and 2 and (A5)).

$$
\begin{aligned}
\mathcal{S}'_0 \cup \neg\mathcal{R}_0 &= \neg\mathcal{R}_0 \cup \mathit{lfp}\lambda X.\, {\uparrow}\big(\big(\alpha_0(Z'_0) \cup \mathit{pre}_{\hat{N}}[T_{A_0}](X)\big) \cap \mathcal{R}_0\big) && \mathcal{R}_0 = \mathcal{R}'_0,\ \alpha'_0 = \alpha_0\\
&= \mathit{lfp}\lambda X.\, {\uparrow}\big(\big(\alpha_0(Z'_0) \cup \mathit{pre}_{\hat{N}}[T_{A_0}](X)\big) \cap \mathcal{R}_0\big) \cup \neg\mathcal{R}_0 && \mathit{pre}(\neg\mathcal{R}_0) \subseteq \neg\mathcal{R}_0,\ \mathit{pre} \text{ is additive}\\
&= \mathit{lfp}\lambda X.\, {\uparrow}\big(\big(\big(\alpha_0(Z'_0) \cup \mathit{pre}_{\hat{N}}[T_{A_0}](X)\big) \cap \mathcal{R}_0\big) \cup \neg\mathcal{R}_0\big) && \neg\mathcal{R}_0\text{: } \leqslant\text{-uc-set}\\
&= \mathit{lfp}\lambda X.\, {\uparrow}\big(\alpha_0(\neg Z_0) \cup \mathit{pre}_{\hat{N}}[T_{A_0}](X) \cup \neg\mathcal{R}_0\big) && \text{Alg. 1 line 1, Alg. 2, line 1}\\
&= \mathit{lfp}\lambda X.\, \alpha_0(\neg Z_0) \cup \mathit{pre}_{\hat{N}}[T_{A_0}](X) \cup \neg\mathcal{R}_0 && \alpha_0(\neg Z_0), \neg\mathcal{R}_0\text{: } \leqslant\text{-uc-set}\\
&= \neg\mathcal{S}_0 && \text{Prop. 6.1}
\end{aligned}
$$

and

$$
\begin{aligned}
Z'_1 \cup \gamma_0(\neg\mathcal{R}_0) &= \gamma_0(\neg\mathcal{R}_0) \cup {\uparrow}((\gamma_0(\mathcal{S}'_0) \cup \mathit{pre}_N(\gamma_0(\mathcal{S}'_0))) \cap \gamma_0(\mathcal{R}_0)) && \mathcal{R}_0 = \mathcal{R}'_0,\ \gamma'_0 = \gamma_0\\
&= \Big\uparrow\Big(\Big(\big(\gamma_0(\mathcal{S}'_0) \cup \mathit{pre}_N(\gamma_0(\mathcal{S}'_0))\big) \cap \gamma_0(\mathcal{R}_0)\Big) \cup \gamma_0(\neg\mathcal{R}_0)\Big) && \gamma_0(\neg\mathcal{R}_0)\text{: } \leqslant\text{-uc set}\\
&= {\uparrow}\big(\gamma_0(\mathcal{S}'_0) \cup \mathit{pre}_N(\gamma_0(\mathcal{S}'_0)) \cup \gamma_0(\neg\mathcal{R}_0)\big) && \text{Lem. 3.5}\\
&= \gamma_0(\mathcal{S}'_0) \cup \mathit{pre}_N(\gamma_0(\mathcal{S}'_0)) \cup \gamma_0(\neg\mathcal{R}_0) && \gamma_0(\mathcal{S}'_0), \gamma_0(\neg\mathcal{R}_0)\text{: } \leqslant\text{-uc-set}\\
&= \gamma_0(\mathcal{S}'_0) \cup \mathit{pre}_N(\gamma_0(\mathcal{S}'_0)) \cup \mathit{pre}_N(\gamma_0(\neg\mathcal{R}_0)) \cup \gamma_0(\neg\mathcal{R}_0) && \mathit{pre}(\gamma_0(\neg\mathcal{R}_0)) \subseteq \gamma_0(\neg\mathcal{R}_0)\\
&= \gamma_0(\mathcal{S}'_0) \cup \mathit{pre}_N(\gamma_0(\mathcal{S}'_0) \cup \gamma_0(\neg\mathcal{R}_0)) \cup \gamma_0(\neg\mathcal{R}_0) && \mathit{pre} \text{ is additive}\\
&= \gamma_0(\mathcal{S}'_0 \cup \neg\mathcal{R}_0) \cup \mathit{pre}_N(\gamma_0(\mathcal{S}'_0 \cup \neg\mathcal{R}_0)) && \text{Lem. 3.7}\\
&= \gamma_0(\neg\mathcal{S}_0) \cup \mathit{pre}_N(\gamma_0(\neg\mathcal{S}_0)) && \text{see above}\\
&= \neg Z_1 && \text{Lem. 3.5, Prop. 6.1}
\end{aligned}
$$

$\boxed{i>0}$ By ind. hyp., we have that $\neg Z_i = Z'_i \cup \gamma_{i-1}(\neg\mathcal{R}_{i-1})$. Note that $post^*_N(m_0) \subseteq \gamma_{i-1}(\mathcal{R}_{i-1})$ and Lemma 3.5 implies that $post^*_N(m_0) \cap \gamma_{i-1}(\neg\mathcal{R}_{i-1}) = \emptyset$. Furthermore, $A_i = A'_i$ by hyp. Hence, we have that $\mathcal{R}_i = \mathcal{R}'_i$ (lines 4-5 of Alg. 1 and 2 and (A5)).

$$\begin{aligned}
\mathcal{S}'_i \cup \neg\mathcal{R}_i &= \neg\mathcal{R}_i \cup \mathit{lfp}\lambda X.\,{\uparrow}\big((\alpha_i(Z'_i) \cup pre_{\tilde N}[T_{A_i}](X)) \cap \mathcal{R}_i\big) && \mathcal{R}_i = \mathcal{R}'_i\\
&= \mathit{lfp}\lambda X.\,{\uparrow}\big((\alpha_i(Z'_i) \cup pre_{\tilde N}[T_{A_i}](X)) \cap \mathcal{R}_i\big) \cup \neg\mathcal{R}_i && pre(\neg\mathcal{R}_i) \subseteq \neg\mathcal{R}_i\\
& && pre \text{ is additive}\\
&= \mathit{lfp}\lambda X.\,{\uparrow}\big(((\alpha_i(Z'_i) \cup pre_{\tilde N}[T_{A_i}](X)) \cap \mathcal{R}_i) \cup \neg\mathcal{R}_i\big) && \neg\mathcal{R}_i\text{: } \leqslant\text{-uc-set}\\
&= \mathit{lfp}\lambda X.\,{\uparrow}\big(\alpha_i(Z'_i) \cup pre_{\tilde N}[T_{A_i}](X) \cup \neg\mathcal{R}_i\big) && \text{then we have}
\end{aligned}$$

(A6) says $\gamma_i(\mathcal{R}_i) \subseteq \gamma_{i-1}(\mathcal{R}_{i-1})$ iff $\gamma_{i-1}(\neg\mathcal{R}_{i-1}) \subseteq \gamma_i(\neg\mathcal{R}_i)$ (by Lem. 3.5) iff $\alpha_i \circ \gamma_{i-1}(\neg\mathcal{R}_{i-1}) \subseteq \neg\mathcal{R}_i$ (Gc)

$$\begin{aligned}
&= \mathit{lfp}\lambda X.\,{\uparrow}\big(\alpha_i(Z'_i) \cup pre_{\tilde N}[T_{A_i}](X) \cup \neg\mathcal{R}_i \cup \alpha_i \circ \gamma_{i-1}(\neg\mathcal{R}_{i-1})\big) && \\
&= \mathit{lfp}\lambda X.\,{\uparrow}\big(\alpha_i(Z'_i \cup \gamma_{i-1}(\neg\mathcal{R}_{i-1})) \cup pre_{\tilde N}[T_{A_i}](X) \cup \neg\mathcal{R}_i\big) && \alpha_i \text{ is additive}\\
&= \mathit{lfp}\lambda X.\,{\uparrow}(\alpha_i(\neg Z_i) \cup pre_{\tilde N}[T_{A_i}](X) \cup \neg\mathcal{R}_i) && \text{ind. hyp.}\\
&= \mathit{lfp}\lambda X.\,\alpha_i(\neg Z_i) \cup pre_{\tilde N}[T_{A_i}](X) \cup \neg\mathcal{R}_i && \alpha_i(Z'_i), \neg\mathcal{R}_i\text{: } \leqslant\text{-uc-sets}\\
&= \neg\mathcal{S}_i && \text{Prop. 6.1}
\end{aligned}$$

The equality $\neg Z_i = Z'_{i+1} \cup \gamma_i(\neg\mathcal{R}_i)$ is proved similarly as for $i = 0$. □

Lemma A.1. Let a Petri net $N = (P, T, F, m_0)$ and a $\leqslant$-dc-set $S \subseteq \mathbb{N}^k$. Assume that Algorithm 1 and 2 compute at least $i > 0$ iterations and for all $0 \le j < i$, $A_j = A'_j$. Then, for each partition $A \in \mathcal{A}^{k_A \times k}$ such that $\gamma_A \circ \alpha_A(\gamma_{i-1}(\mathcal{R}_{i-1})) = \gamma_{i-1}(\mathcal{R}_{i-1})$ we find that $\gamma_A \circ \alpha_A(Z_i) = Z_i$ iff $\gamma_A \circ \alpha_A(Z'_i) = Z'_i$.

Proof:
By hyp., we have that $A_{i-1} = A'_{i-1}$. Furthermore, if $i - 1 = 0$ then $\neg Z_{i-1} = Z'_{i-1} = \neg S$ (line 1 of Alg. 1 and 2). Otherwise, by Lemma 6.1, we have that $\neg Z_{i-1} = Z'_{i-1} \cup \gamma_{i-2}(\neg\mathcal{R}_{i-2})$. Since $post^*_N(m_0) \subseteq \gamma_{i-2}(\mathcal{R}_{i-2})$, by Lemma 3.5, we obtain that $post^*_N(m_0) \cap \gamma_{i-2}(\neg\mathcal{R}_{i-2}) = \emptyset$. Hence, we have that $\mathcal{R}_{i-1} = \mathcal{R}'_{i-1}$ (lines 4-5 of Alg. 1 and 2 and (A5)), i.e. $\gamma_{i-1}(\mathcal{R}_{i-1}) = \gamma'_{i-1}(\mathcal{R}'_{i-1}) = \mathcal{R}$. Lem. 3.6, also shows that $\gamma_A \circ \alpha_A(\neg\mathcal{R}) = \neg\mathcal{R}$. $\boxed{\Leftarrow}$ Lem. 6.1 shows that $\neg Z_i = Z'_i \cup \neg\mathcal{R}$, hence, using hyp. and applying Lem. 3.6 we conclude that $\gamma_A \circ \alpha_A(Z_i) = Z_i$.

$\boxed{\Rightarrow}$ We conclude from Lem. 6.1 that $\neg Z_i = Z'_i \cup \neg\mathcal{R}$, hence that $\neg Z_i \cap \mathcal{R} = Z'_i \cap \mathcal{R}$ and finally that $\gamma_A \circ \alpha_A(Z'_i \cap \mathcal{R}) = Z'_i \cap \mathcal{R}$ by applying Lem. 3.6 and hyp. By def., $Z'_i = {\uparrow}(Z'_i \cap \mathcal{R})$. Then, Lem. 3.5 shows that $Z'_i = {\uparrow}(Z'_i \cap \mathcal{R}) = {\uparrow}(\gamma_A \circ \alpha_A(Z'_i \cap \mathcal{R})) = \gamma_A \circ \alpha_A({\uparrow}(Z'_i \cap \mathcal{R})) = \gamma_A \circ \alpha_A(Z'_i)$. □

Here follows the proof of Lemma 6.2

Proof:
By induction on i. $\boxed{i=0}$ Remark that $Z_0 = S$, $Z'_0 = \neg S$. Following Lem. 3.6, for each partition $A \in \mathcal{A}^{k_A \times k}$, $\gamma_A \circ \alpha_A(S) = S$ iff $\gamma_A \circ \alpha_A(\neg S) = \neg S$. By def. of A_0 and A'_0 we conclude that $A_0 = A'_0$.

$\boxed{i>0}$ By ind. hyp., we have that $A_{i-1} = A'_{i-1}$. Furthermore, if $i - 1 = 0$ then $\neg Z_{i-1} = Z'_{i-1} = \neg S$. Otherwise, by Lemma 6.1, we have that $\neg Z_{i-1} = Z'_{i-1} \cup \gamma_{i-2}(\neg\mathcal{R}_{i-2})$. Since $post^*_N(m_0) \subseteq \gamma_{i-2}(\mathcal{R}_{i-2})$, by Lemma 3.5, we obtain that $post^*_N(m_0) \cap \gamma_{i-2}(\neg\mathcal{R}_{i-2}) = \emptyset$. Hence, $\mathcal{R}_{i-1} = \mathcal{R}'_{i-1}$ (lines 4-5 of Alg. 1 and 2 and (A5)), i.e. $\gamma_{i-1}(\mathcal{R}_{i-1}) = \gamma'_{i-1}(\mathcal{R}'_{i-1}) = \mathcal{R}$. By def., we have that $A_i = A_{i-1} \curlywedge \lceil Z_i \rceil$ and $A'_i = A'_{i-1} \curlywedge \lceil Z'_i \rceil$, hence $A'_i = A_{i-1} \curlywedge \lceil Z'_i \rceil$ by induction hyp. Furthermore, by construction we see that $A_{i-1} = A_{i-1} \curlywedge \lceil \mathcal{R} \rceil$ and $A'_{i-1} = A'_{i-1} \curlywedge \lceil \mathcal{R} \rceil$, hence that $A_i = A_{i-1} \curlywedge \tilde{A}$ and $A'_i = A_{i-1} \curlywedge \tilde{A}'$ where $\tilde{A} = \lceil Z_i \rceil \curlywedge \lceil \mathcal{R} \rceil$ and $\tilde{A}' = \lceil Z'_i \rceil \curlywedge \lceil \mathcal{R} \rceil$. To conclude the proof we show that $\tilde{A} = \tilde{A}'$. Lemma A.1 shows that $\gamma_{\tilde{A}} \circ \alpha_{\tilde{A}}(Z'_i) = Z'_i$, hence that $\tilde{A} = \tilde{A} \curlywedge \lceil Z'_i \rceil$. Similarly we find that $\tilde{A}' = \tilde{A}' \curlywedge \lceil Z_i \rceil$. Finally, $\tilde{A} = \lceil Z_i \rceil \curlywedge \lceil \mathcal{R} \rceil = \lceil Z_i \rceil \curlywedge \lceil \mathcal{R} \rceil \curlywedge \lceil Z'_i \rceil = \lceil Z_i \rceil \curlywedge \tilde{A}' = \tilde{A}'$. □