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

Pierre Ganty, Jean-François Raskin, and Laurent Van Begin

Département d'InformatiqUe, Université Libre de Bruxelles
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 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 unconclusive 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, parameterized systems [1]. Even though their state space is infinite, several interesting problems are decidable on Petri nets. The seminal work of Karp and Miller [2] shows that, for Petri nets, an effective representation of the downward closure of the set of reachable markings, the so-called *coverability set*, is constructible. 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* [3, 4]. 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 [5, 6, 7, 8]. Forward and backward algorithms are now available and have been implemented to show their practical relevance. All those methods manipulate, somehow or other, infinite sets of markings. Sets of markings are subsets of $\mathbb{N}^k$ 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 suffers 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 manipulates subsets of $\mathbb{N}^{12}$. Our method will automatically consider sets of lower dimensionality: in this case subsets of $\mathbb{N}^4$, so, even smaller, for this example, than the ones of the Petri net of Fig. 1(b).

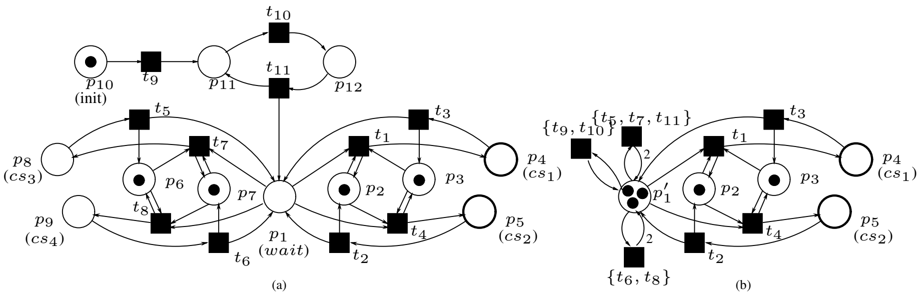

Fig. 1. A Petri net with two distinct mutual exclusion properties (a) and its abstraction (b).

Our algorithm is based on two main ingredients: abstract interpretation and automatic refinements. 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 unconclusive, because it is not precise enough, our algorithm automatically refines the abstract domain. This refinement ensures that the next analysis will be more precise and 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 have been computed by previous steps and is quite different from the technique known as *counterexample guided abstraction refinement* [10].

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 [11], 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 [11].

Due to the lack of space the omitted proofs can be found in a technical report available at `http://www.ulb.ac.be/di/ssd/cfv/publications.html`.

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 based on abstraction and refinement.

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 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: $\mathcal{I}, \mathcal{O} \colon P \times T \mapsto \mathbb{N}$ describing the relationship between places and transitions. Once a linear order has been fixed on P and on T, $\mathcal{I}$ and $\mathcal{O}$ can be seen as $(|P|, |T|)$-matrices over $\mathbb{N}$ ($\mathbb{N}^{|P|\times|T|}$ for short). Let $t \in T$, $\mathcal{I}(t)$ denote the t-column vector in $\mathbb{N}^{|P|}$ of $\mathcal{I}$.*
- *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 any infinite sequence of markings $m_1, m_2, \ldots, m_i, \ldots$ there exists indexes $i < j$ such that $m_i \leqslant m_j$.

Definition 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 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 $\mathbb{N}^k$ are 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 $(f \circ g)(x) \stackrel{\text{def}}{=} f(g(x))$. Also we define $f^{i+1} = f^i \circ f$ and $f^0 = \lambda x.\, x$. Sometimes we also use logical formulas. Given a logical formula ψ we write $[\![\psi]\!]$ for the set of its satisfying valuations.

Definition 4 (The predicate transformers pre, $\widetilde{pre}$, and $post$). *Let N a Petri net given by (P, T, F, m_0) and let $t \in T$, we define $pre_N[t], \widetilde{pre}_N[t], post_N[t] \colon \wp(\mathbb{N}^k) \mapsto \wp(\mathbb{N}^k)$ as follows,*

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

The extension to the set T of transitions is given by,

$$f_N = \begin{cases} \lambda X.\, \bigcup_{t \in T} f_N[t](X) & \text{if } f_N \text{ is } pre_N \text{ or } post_N \\ \lambda X.\, \bigcap_{t \in T} f_N[t](X) & \text{for } f_N = \widetilde{pre}_N. \end{cases}$$

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 [12]: for any $X, Y \subseteq \mathbb{N}^k$ we have

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

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 $DCS(\mathbb{N}^k)$ ($UCS(\mathbb{N}^k)$) to be the set of all $\leqslant$-dc-sets ($\leqslant$-uc-sets). Those closed sets have natural effective representations that are based on the fact that $\leqslant$ is a wqo. A set $M \subseteq \mathbb{N}^k$ is said to be *canonical* if for any distinct $x, y \in \mathbb{N}^k$ 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 1 ([13]). *Given $S \subseteq \mathbb{N}^k$, S has exactly one finite canonical minor set.*

So, any $\leqslant$-uc-set can be represented by its finite set of minimal elements. As any $\leqslant$-dc-set is the complement of a $\leqslant$-upward closed, it has an effective representation also. Sets of omega-markings is an equivalent alternative representation [2].

We now formally state the coverability problem for Petri nets which correspond to a fixpoint checking problem. Our formulation follows [12].

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 [12, 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 relations (1) and (2), respectively. They are called the *forward semantics* and the *backward semantics* of the net. Note also that since S is a $\leqslant$-dc-set, $\mathit{post}^*(m_0) \subseteq S$ if and only if $\downarrow(\mathit{post}^*(m_0)) \subseteq S$.

Existing algorithms The solutions to problem 1 found in the literature (see [14, 15]) iteratively compute finer overapproximations of $\downarrow(\mathit{post}^*(m_0))$. They end up with an overapproximation $\mathcal{R}$ satisfying the following properties:

$$\mathit{post}_N^*(m_0) \subseteq \mathcal{R} \tag{A1}$$
$$\mathcal{R} \in DCS(\mathbb{N}^k) \tag{A2}$$
$$\mathit{post}_N(\mathcal{R}) \subseteq \mathcal{R} \tag{A3}$$
$$\mathit{post}_N^*(m_0) \subseteq S \rightarrow \mathcal{R} \subseteq S \tag{A4}$$

The solutions of [14, 15] 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 [2] the authors show that $\downarrow(\mathit{post}^*(m_0))$ is computable for Petri nets and thus the approximation scheme presented above also encompasses this solution. All these solutions have in input an effective representation for (1) the initial marking m_0, (2) the predicate transformer post_N associated to the Petri net N and (3) the $\leqslant$-dc-set S.

In the literature, see for example [4], there are also solutions which compute the set $\widetilde{\mathit{pre}}^*(S)$ by evaluating its associated fixpoint (see (2)). Since [4], this fixpoint is known to be computable for 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.

[1] The fixpoint expression considered in [4] is actually different from (2) but coincides with its complement.

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 a place of the Petri net, in the abstract domain, each dimension records the sum of the number of tokens contained into 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, it is refined automatically using a refinement procedure that is guided by the unconclusive 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 Sect. 4, we define efficient way to overapproximate the abstract semantics defined in Sect. 3. In Sect. 5, we show how to refine automatically abstract domains. We define there an algorithm that given a concrete set M computes the coarsest abstract domain that is able to represent M exactly. In Sect. 6, we put all those results together to obtain our algorithm that decide coverability by successive approximations and refinements. In Sect. 7, we report on experimentations that show the practical interest of our new algorithm.

3 Abstraction of Sets of Markings

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

Definition 5. *Let A be a partition of the set $[1..k]$ into k'* classes $\{C_i\}_{i\in[1..k']}$. *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 [16], that the set of partitions of $[1..k]$ together with $\preceq$ form a complete lattice where $\{\{1\},\dots,\{k\}\}$ is the bottom element, $\{\{1,\dots,k\}\}$ is the top 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 finer partition such that given $C \in A_1 \cup A_2$ and $\{a_1, a_2\} \subseteq C$ we have $\exists C' \in A_1 \curlyvee A_2 \colon \{a_1, a_2\} \subseteq C'$.*

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' classes we abstract m into a k'-uple m' by taking the sum of 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 6. *Let A be a partition of $[1..k]$ given by $\{C_i\}_{i\in[1..k']}$. We associate to this partition a matrix $A := (a_{ij})_{k'\times k}$ such that $a_{ij} = 1$ if $j \in C_i$, $a_{ij} = 0$ otherwise. So, $A \in \{0,1\}^{k'\times k}$. We write $\mathcal{A}^{k'\times k}$ to denote the set of matrices associated to the partitions of $[1..k]$ into k' classes.*

We sometimes call such a A an *abstraction*.

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

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

In the following, 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 1. *Let $A \in \mathcal{A}^{k' \times k}$, we have $(\wp(\mathbb{N}^k), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\wp(\mathbb{N}^{k'}), \subseteq)$.*

Proof. Let $X \subseteq \mathbb{N}^k$ and $Y \subseteq \mathbb{N}^{k'}$,

$$\begin{array}{ll} \alpha(X) \subseteq Y & \\ \Leftrightarrow \{A \cdot x \mid x \in X\} \subseteq Y & \text{def. 7} \\ \Leftrightarrow \forall x \colon x \in X \rightarrow A \cdot x \in Y & \\ \Leftrightarrow X \subseteq \{x \mid A.x \in Y\} & \\ \Leftrightarrow X \subseteq \gamma(Y) & \text{def. 7} \end{array}$$

Now given $y \in Y$ we define $x \in \mathbb{N}^k$ such that for each class C_i of A and 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 $A \cdot x = y$.

$$\begin{array}{rll} \alpha \circ \gamma(Y) & = \alpha(\{x \mid A \cdot x \in Y\}) & \\ & = \{A \cdot x \mid A \cdot x \in Y\} & \text{def. 7} \\ & = Y & \text{by above} \quad \square \end{array}$$

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 shows the usefulness of such approximations.

Lemma 2. *Given a Petri net N and a $\leqslant$-dc-set S we have*

$$\gamma(\mathcal{F}) \subseteq S \rightarrow post^*_N(m_0) \subseteq S$$
$$\{m_0\} \not\subseteq \gamma(\mathcal{B}) \rightarrow post^*_N(m_0) \not\subseteq S$$

Abstract interpretation [9] tells us that to compute an overapproximation of fixpoints of a concrete function, we must first approximate this function by an abstract function and compute the fixpoint of this abstract function in the abstract domain. Among the abstractions of a function f is the most precise one. In [9] the authors show that, in the context of a Galois insertion, the most precise approximation of f is unique and given by $\alpha \circ f \circ \gamma$. So to approximate $\alpha(post^*_N(m_0))$ and $\alpha(\widetilde{pre}^*_N(S))$ we obtain the following fixpoint expression in the abstract domain:

$$\begin{aligned} &lfp\lambda X.\, \alpha(\{m_0\} \cup post(\gamma(X))) \enspace, \\ &gfp\lambda X.\, \alpha(S \cap \widetilde{pre}(\gamma(X))) \enspace, \end{aligned} \tag{3}$$

respectively. This definition 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 $\{C_i\}_{i\in[1..k']}$ and let $\hat{m} \in \mathbb{N}^{k'}$, we have $|\gamma(\hat{m})| = \prod_{i\in[1..k']} \binom{\hat{m}(i)+|C_i|-1}{|C_i|-1}$. Section 4 is devoted to the definition of most precise approximation without explicitly evaluating γ.

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 unconclusive we refine it into an abstraction A_{i+1} which overapproximates sets of markings and predicate transformers more precisely than A_i. Here follows a result relating the precision of abstractions with their underlying partitions.

Lemma 3. *Let A, A' be two partitions of $[1..k]$ such that $A \preceq A'$ and $M \subseteq \mathbb{N}^k$,*

$$\gamma_A \circ \alpha_A(M) \subseteq \gamma_{A'} \circ \alpha_{A'}(M) \enspace .$$

So by refining partitions, we refine abstractions. We will see in Sect. 5 how to use this result systematically in our algorithm and how to take into account previous computation when a refinement is done. The following result tells us that if two partitions are able to represent exactly a set then their *lub* is also able to represent that set. So, for any set M there is a coarsest partition which is able to represent it.

Lemma 4. *Let A, A_1, A_2 be three partitions of $[1..k]$ such that $A = A_1 \curlyvee A_2$ and $M \subseteq \mathbb{N}^k$, we have*

$$\textit{if} \begin{Bmatrix} \gamma_{A_1} \circ \alpha_{A_1}(M) \subseteq M \\ \gamma_{A_2} \circ \alpha_{A_2}(M) \subseteq M \end{Bmatrix} \textit{ then } \gamma_A \circ \alpha_A(M) \subseteq M \enspace . \tag{4}$$

Proof. First 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 left hand side of (4).

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' is obtained from m by moving tokens inside the equivalence classes of $\equiv_A$. More precisely, let $v \in \mathbb{N}$, and a, b 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 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 [16, 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

$$\langle c_i, c_{i+1}\rangle \in \equiv_{A_1} \qquad \text{or} \qquad \langle c_i, c_{i+1}\rangle \in \equiv_{A_2} \tag{5}$$

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 eq. (5) we have that each move of the sequence is defined inside an equivalence class of $\equiv_{A_1}$ or $\equiv_{A_2}$. Hence each move of the sequence 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

$$\forall j \in [1..n] \colon i_j \in [1..2] \quad \text{and} \quad m' \in \mu_{A_{i_1}} \circ \mu_{A_{i_2}} \circ \ldots \circ \mu_{A_{i_n}}(\{m\}) \ .$$

Finally, left hand side of (4) and monotonicity of μ_{A_1}, μ_{A_2} shows that $m' \in M$. $\square$

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 any Petri net N of dimensionality k and abstraction $A \in \mathcal{A}^{k' \times k}$, we can associate a Petri net $\hat{N}$ of dimensionality k' whose concrete forward and backward semantics gives precise overapproximations of the abstraction by A of the semantics of N given by (3).

Abstract net In order to efficiently evaluate the best approximation of $post_N$ and $\widetilde{pre}_N[t]$ for each $t \in T$, without explicitly evaluating γ, we associate for each N and A a Petri net $\hat{N}$.

Definition 8. *Let N be a Petri net given by (P, T, F, m_0) and let $A \in \mathcal{A}^{k' \times k}$. We define the tuple $(\hat{P}, T, \hat{F}, \widehat{m_0})$ where*

- *$\hat{P}$ is a set of k' places (one for each class of the partition A),*
- *$\hat{F} = (\widehat{\mathcal{I}}, \widehat{\mathcal{O}})$ is such that $\widehat{\mathcal{I}} \stackrel{\mathrm{def}}{=} A \cdot \mathcal{I}$ and $\widehat{\mathcal{O}} \stackrel{\mathrm{def}}{=} A \cdot \mathcal{O}$,*
- *$\widehat{m_0}$ 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.

To establish properties of the semantics of this abstract net (given in Prop. 2 and Prop. 3 below), we need the technical results:

Lemma 5. *Let $A \in \mathcal{A}^{k' \times k}$, $f \in \{\uparrow, \downarrow, \neg\}$ and $X \subseteq \mathbb{N}^k$,*

$$\gamma \circ f(X) = f \circ \gamma(X) \qquad \alpha \circ \uparrow(X) = \uparrow \circ\, \alpha(X) \qquad \alpha \circ \downarrow(X) = \downarrow \circ\, \alpha(X) \ .$$

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. $\gamma(A \cap B) = \gamma(A) \cap \gamma(B)$).

Forward overapproximation The next proposition states that the most precise approximation of the predicate transformer $post_N$ is given by the predicate transformer $post_{\hat{N}}$ of the abstract net.

Proposition 2. *Given a Petri net $N = (P, T, F, m_0)$, $A \in \mathcal{A}^{k' \times k}$ and $\hat{N}$ the Petri net given by def. 8, we have*

$$\lambda X.\, \alpha \circ post_N \circ \gamma(X) = \lambda X.\, post_{\hat{N}}(X) \ .$$

Proof. Definition 4 states that $post_N = \lambda X. \bigcup_{t \in T} post_N[t](X)$. Thus, for $t \in T$, we show that $\alpha \circ post_N[t] \circ \gamma(\hat{m}) = post_{\hat{N}}[t](\hat{m})$. Then the additivity of α shows the desired result.

For each $t \in T$, for each $\hat{m} \in \mathbb{N}^{k'}$,

$$
\begin{array}{ll}
\alpha \circ post_N[t] \circ \gamma(\hat{m}) & \\
= \alpha \circ post_N[t](\{m \mid m \in \gamma(\hat{m})\}) & \\
= \alpha(\{m - \mathcal{I}(t) + \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\}) & \text{def. 2} \\
= \{A\cdot (m - \mathcal{I}(t) + \mathcal{O}(t)) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} & \text{def. 7} \\
= \{A\cdot m - A\cdot \mathcal{I}(t) + A\cdot \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} & \\
= \{\alpha(m) - A\cdot \mathcal{I}(t) + A\cdot \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} & \text{def of } \alpha \\
= \{\hat{m} - A\cdot \mathcal{I}(t) + A\cdot \mathcal{O}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} & \xrightleftharpoons[\alpha]{\gamma} \\
= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid m \in \gamma(\hat{m}) \wedge \mathcal{I}(t) \leqslant m\} & \text{def. 8} \\
= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \{\mathcal{I}(t)\} \subseteq \downarrow \circ\, \gamma(\hat{m})\} & \text{def of } \downarrow \\
= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \{\mathcal{I}(t)\} \subseteq \gamma \circ \big\downarrow(\hat{m})\} & \text{Lem. 5} \\
= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \alpha(\{\mathcal{I}(t)\}) \subseteq \big\downarrow(\hat{m})\} & \xrightleftharpoons[\alpha]{\gamma} \\
= \{\hat{m} - \widehat{\mathcal{I}}(t) + \widehat{\mathcal{O}}(t) \mid \widehat{\mathcal{I}}(t) \leqslant \hat{m}\} & \text{def. 8} \\
= post_{\hat{N}}[t](\hat{m}) & \text{def. 2} \quad \square
\end{array}
$$

The consequences of Prop 2 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. 2.

Backward overapproximation The backward semantics of the transitions of the abstract net are the best approximation of the backward semantics of the transitions of the concrete net. However, the best abstraction of the predicate transformer $\widetilde{pre}_N$ does not coincide with $\widetilde{pre}_{\hat{N}}$ as we will see later. To obtain those results, we need some intermediary lemmas (i.e. Lem. 6, 7 and 8).

Lemma 6. *Given a Petri net $N = (P, T, F, m_0)$ and $A \in \mathcal{A}^{k' \times k}$ we have*

$$\lambda X.\, \alpha \circ pre_N \circ \gamma(X) = \lambda X.\, pre_{\hat{N}}(X) \enspace .$$

Proof. The proof is similar to the proof of Prop. 2 with $\mathcal{O}$ (resp. $\widehat{\mathcal{O}}$) replaced by $\mathcal{I}$ (resp. $\widehat{\mathcal{I}}$) and vice versa. $\square$

Lemma 7. *Given a Petri net $N = (P, T, F, m_0)$ and a partition $A = \{C_j\}_{j\in[1..k']}$ of $[1..k]$, if $\exists i \in [1..k] \colon \mathcal{I}(i, t) > 0$ and $\{i\} \notin A$ then $\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) = \mathbb{N}^{k'}$.*

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:

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

We conclude from $i \in [1..k]$ and $\mathcal{I}(i,t) > 0$ that $[\![x_i < \mathcal{I}(i,t)]\!] = \{\langle v_1, \ldots, v_i, \ldots, v_k\rangle \mid v_i < \mathcal{I}(i,t)\}$, hence that $\alpha([\![x_i < \mathcal{I}(i,t)]\!]) = \mathbb{N}^{k'}$ 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 $\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) = \alpha([\![\bigvee_{\substack{p\in[1..k]\\ \mathcal{I}(p,t)>0}} x_p < \mathcal{I}(p,t)]\!]) = \mathbb{N}^{k'}$. □

Lemma 8. *Given a Petri net $N = (P,T,F,m_0)$, a partition $A = \{C_j\}_{j\in[1..k']}$ of $[1..k]$ and $\hat{N}$ the Petri net given by def. 8, if for any $i \in [1..k]$: $\mathcal{I}(i,t) > 0$ implies $\{i\} \in A$, then $\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) = \{m \in \mathbb{N}^{k'} \mid \widehat{\mathcal{I}}(t) \nleqslant m)\}$.*

Proof.

$$\begin{aligned}
&\alpha(\{m \in \mathbb{N}^k \mid \mathcal{I}(t) \nleqslant m\}) && \\
&= \{A\cdot m \mid m \in \mathbb{N}^k \wedge \mathcal{I}(t) \nleqslant m\}) && \text{def. of } \alpha\\
&= \{A\cdot m \mid m \in \mathbb{N}^k \wedge A\cdot\mathcal{I}(t) \nleqslant A\cdot m\} && \text{hyp.}\\
&= \{A\cdot m \mid m \in \mathbb{N}^k \wedge \widehat{\mathcal{I}}(t) \nleqslant A\cdot m\} && \text{def. of } \widehat{\mathcal{I}}\\
&= \{\widehat{m} \in \mathbb{N}^{k'} \mid \exists m \in \mathbb{N}^k : \widehat{m} = A\cdot m \wedge \widehat{\mathcal{I}}(t) \nleqslant \widehat{m}\} && \\
&= \{\widehat{m} \in \mathbb{N}^{k'} \mid \widehat{\mathcal{I}}(t) \nleqslant \widehat{m}\} && \text{tautology} \quad \square
\end{aligned}$$

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

Proposition 3. *Given a Petri net $N = (P,T,F,m_0)$, a partition $A = \{C_j\}_{j\in[1..k']}$ of $[1..k]$ and $\hat{N}$ the Petri net given by def. 8, we have*

$$\lambda X.\, \alpha \circ \widetilde{pre}_N[t] \circ \gamma(X) = \begin{cases} \mathbb{N}^{k'} & \text{if } \exists i \in [1..k'] \colon |C_i| > 1 \wedge \widehat{\mathcal{I}}(i,t) > 0\\ \lambda X.\, \widetilde{pre}_{\hat{N}}[t](X) & \text{otherwise.}\end{cases}$$

Proof.

$$\begin{aligned}
&\alpha \circ \widetilde{pre}_N[t] \circ \gamma(S) && \\
&= \alpha \circ \widetilde{pre}_N[t](\{m \in \mathbb{N}^k \mid m \in \gamma(S)\}) && \\
&= \alpha(\{m \mid (\mathcal{I}(t) \nleqslant m) \vee (\mathcal{I}(t) \leqslant m \wedge m - \mathcal{I}(t) + \mathcal{O}(t) \in \gamma(S))\}) && \text{def. of } \widetilde{pre}_N[t]\\
&= \alpha(\{m \mid \mathcal{I}(t) \nleqslant m\}) \cup \alpha(\{m \mid \mathcal{I}(t) \leqslant m \wedge m - \mathcal{I}(t) + \mathcal{O}(t) \in \gamma(S))\}) && \text{additivity of } \alpha\\
&= \alpha(\{m \mid \mathcal{I}(t) \nleqslant m\}) \cup \alpha \circ pre_N[t] \circ \gamma(S) && \text{def of } pre_N[t]\\
&= \alpha(\{m \mid \mathcal{I}(t) \nleqslant m\}) \cup pre_{\hat{N}}[t](S) && \text{by lem. 6}
\end{aligned}$$

We now consider two cases:

- $\exists i \in [1..k] \colon \mathcal{I}(i,t) > 0$ and $\{i\} \notin A$. From Lemma 7, we conclude that $\alpha \circ \widetilde{pre}_N[t] \circ \gamma(S) = \mathbb{N}^{k'}$;
- $\forall i \in [1..k] \colon \mathcal{I}(i,t) > 0$ implies $\{i\} \in A$. In this case we have

$$\begin{aligned}
\alpha \circ \widetilde{pre}_N[t] \circ \gamma(S) &= \{m \in \mathbb{N}^{k'} \mid \widehat{\mathcal{I}}(t) \nleqslant m\} \cup pre_{\hat{N}}[t](S) && \text{by Lem. 8}\\
&= \widetilde{pre}_{\hat{N}}[t](S) && \text{def of } \widetilde{pre}_{\hat{N}}[t] \quad \square
\end{aligned}$$

Now, let us see how to approximate $\widetilde{pre}_N$. We can do that by distributing α over $\cap$ as shown below at the cost of an overapproximation:

$$\begin{aligned}
&\mathit{gfp}(\lambda X.\, \alpha(S \cap \widetilde{pre}_N \circ \gamma(X))) && (3)\\
&= \mathit{gfp}(\lambda X.\, \alpha(S \cap \bigcap_{t\in T} \widetilde{pre}_N[t] \circ \gamma(X))) && \text{def of } \widetilde{pre}_N\\
&\subseteq \mathit{gfp}(\lambda X.\, \alpha(S) \cap \bigcap_{t\in T} \alpha \circ \widetilde{pre}_N[t] \circ \gamma(X)) && \alpha(A\cap B) \subseteq \alpha(A) \cap \alpha(B)
\end{aligned}$$

Thus, this weaker result for the backward semantics stems from the definition of $\widetilde{pre}_N$ given by $\bigcap_{t\in T} \widetilde{pre}_N[t]$ and the fact that α is not *co-additive* (i.e. $\alpha(A \cap B) \neq \alpha(A) \cap \alpha(B)$).

5 Abstraction refinement

In the abstraction refinement paradigm, if the current abstraction A_i is unconclusive it is refined. A refinement step will then produce an abstraction A_{i+1} which overapproximates sets of markings and predicate transformers more precisely than A_i.

We showed in Lem. 3 that if partition A refines A' (i.e. $A \prec A'$) then A represents sets of markings (and hence function over sets of markings) more precisely than A'. Note also that the partition A where each class is a singleton (i.e. the bottom partition) we have $\gamma_A \circ \alpha_A(S) = S$ for any set of markings S. Thus the loss of precision stems from the classes of the partition which are not singleton.

With these intuitions in mind we will refine an abstraction A_i into A_{i+1} by splitting classes of A_i which are not singleton. A first idea to refine abstraction A_i is to split a randomly chosen non singleton class of A_i. This approach is complete since it will eventually end up with the bottom partition which yields to a conclusive analysis with certainty. However, we adopt a different strategy which consists in computing for A_{i+1} the coarsest partition refining A_i and which able to represent precisely a given set of markings.

Now we present the algorithm $\mathsf{refinement}$ that given a set of markings M computes the coarsest partition A which is able to represent M precisely. The algorithm starts from the bottom partition then the algorithm chooses non-deterministically two candidate classes and merge them in a unique class. If this new partition still represents M precisely, we iterate the procedure. Otherwise the algorithm tries choosing different candidates. The algorithm is presented in Alg. 1.

Let $A = \{C_i\}_{i\in[1..k']}$ be a partition of $[1..k]$, we define $A_{C_i} = \{C_i\} \cup \{\{s\} \mid s \in [1..k] \wedge s \notin C_i\}$. We first prove the following lemma.

Lemma 9. *Let $A = \{C_i\}_{i\in[1..k']}$ be a partition of $[1..k]$, $M \subseteq \mathbb{N}^k$, we have:*

$$\gamma_A \circ \alpha_A(M) \subseteq M \Leftrightarrow \bigwedge_{C_i\in A} \gamma_{A_{C_i}} \circ \alpha_{A_{C_i}}(M) \subseteq M \enspace .$$

Algorithm 1: refinement

Input: $M \subseteq \mathbb{N}^k$
Output: a partition A of $[1..k]$ such that $\gamma_A \circ \alpha_A(M) \subseteq M$
Let A be $\{\{1\},\{2\},\ldots,\{k\}\}$;
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 $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq M$;
    $A \leftarrow (A \setminus \{C_i, C_j\}) \cup \{C_i \cup C_j\}$;

The following two lemmas and the corollary state the correctness and the optimality of Alg. 1.

Lemma 10. *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 $\gamma_A \circ \alpha_A(M) = M$ and so $\gamma_A \circ \alpha_A(M) \subseteq M$ which is an invariant maintained by the iteration following Lem. 9. □

Lemma 11. *Given $M \subseteq \mathbb{N}^k$ and 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 $A \preceq A'$, $\exists C_i, C_j \in A \, \exists C' \in A' \colon (C_i \neq C_j) \wedge C_i \cup C_j \subseteq C'$. We conclude from Lem. 9 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, we have by monotonicity that $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) \subseteq \gamma_{A_{C'}} \circ \alpha_{A_{A'}}(M) = M$. Since $M \subseteq \gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M)$ by Galois insertion, we conclude that $\gamma_{A_{C_i \cup C_j}} \circ \alpha_{A_{C_i \cup C_j}}(M) = M$.

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

Putting together Lem. 4 and 11 we get:

Corollary 1. *Given $M \subseteq \mathbb{N}^k$, the partition A returned by* $\mathsf{refinement}(M)$ *is the coarsest partition such that $\gamma_A \circ \alpha_A(M) = M$.*

6 The Algorithm

The algorithm we propose is given in Alg. 2. Given a Petri net N and a $\leqslant$-dc-set S, the Algorithm builds abstractions $\hat{N}$ with smaller dimensionality than N (lines 8), analyses them (lines 5-13), and refines them (line 15) 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 $\alpha_i(S)$ based any algorithm proposed in [2, 15, 14]. Besides an answer those algorithms returns 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, algorithm 5 concludes that $post^*_N(m_0) \subseteq S$ (line 6). Otherwise, Algorithm 2 tries to decide if $\{m_0\} \not\subseteq S$ checking the inclusion given by 2 (line 9-13). The fixpoint of (2) is computable ([4]) 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 $gfp\lambda X.\, \alpha_i(S) \cap \bigcap_{t\in T} \alpha_i \circ \widetilde{pre}_N[t] \circ \gamma_i(X)$ whose concretization is an overapproximation of $gfp\lambda X.\, S \cap \widetilde{pre}_N(X)$. 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 to obtain a smaller fixpoint, and point (ii) is motivated by the fact that at each step i we have $gfp\lambda X.\, \alpha_i(S) \cap \mathcal{R}_i \cap \bigcap_{t\in T} \alpha_i \circ \widetilde{pre}_N[t] \circ \gamma_i(X) \subseteq \alpha_i(Z_i) \subseteq \alpha_i(S)$. That allows us consider $\alpha_i(Z_i)$ instead of $\alpha_i(S)$ without changing the fixpoint, leading to a more efficient computation of it (see [4] for more details). Those optimisations are safe in the sense that the fixpoint we evaluate at line 9 does not contain $\alpha_i(m_0)$ implies that $post^*_N(m_0) \not\subseteq S$, 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 9 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 model-checker will return the answer OK at iteration $i+1$. Moreover, if $post^*_N(m_0) \not\subseteq S$ then $\{m_0\} \not\subseteq \widetilde{pre}^*_N(S)$, hence $\{m_0\} \not\subseteq \widetilde{pre}^*_N(S) \cap \mathcal{R}$, and the algorithm will return KO at step $i+1$ because we have $Z_{i+1} = \widetilde{pre}^*_N(S) \cap \mathcal{R}$, hence $\{\widehat{m_0}\} \not\subseteq \alpha_{i+1}(Z_{i+1})$ 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 terminate

Proposition 4 (Soundness). *If Algorithm 2 says "OK" then we have*

$$lfp\lambda X.\, m_0 \cup post(X) \subseteq S \ .$$

Algorithm 2: Algorithm for the coverability problem, assume $\{m_0\} \subseteq S$

Data: A Petri net $N = (P, T, F, m_0)$, a $\leqslant$-dc-set S

1 $Z_0 = S$
2 $A_0 = \mathsf{refinement}(Z_0)$
3 **for** $i = 0, 1, 2, 3, \ldots$ **do**
4 $\quad$ $\boxed{\textbf{Abstract:}}$ Given A_i, compute $\hat{N}$ given by def. 8.
5 $\quad$ $\boxed{\textbf{Verify:}}$ $(answer, \mathcal{R}_i) = \mathsf{Checker}(\widehat{m_0}, post_{\hat{N}}, \alpha_i(S)$
6 $\quad$ **if** $answer == OK$ **then**
7 $\quad\quad$ **return** *OK*
8 $\quad$ **else**
9 $\quad\quad$ Let $\mathcal{S}_i = gfp\lambda X.\, \alpha_i(Z_i) \cap \mathcal{R}_i \cap \bigcap_{t \in T} \alpha_i \circ \widetilde{pre}_N[t] \circ \gamma_i(X)$
10 $\quad\quad$ **if** $\widehat{m_0} \not\subseteq \mathcal{S}_i$ **then**
11 $\quad\quad\quad$ **return** *KO*
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 $\quad$ Let $Z_{i+1} = Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(\gamma_i(\mathcal{S}_i))$
15 $\quad$ $\boxed{\textbf{Refine:}}$ Let $A_{i+1} = A_i \curlywedge \mathsf{refinement}(Z_{i+1})$
16 **end**

Proof. If Algorithm says "OK" then

$$
\begin{array}{ll}
\mathcal{R}_i \subseteq \alpha_i(S) & \\
\Rightarrow \gamma_i(\mathcal{R}_i) \subseteq \gamma_i \circ \alpha_i(S) & \gamma_i \text{ is monotonic} \\
\Rightarrow \gamma_i(\mathcal{R}_i) \subseteq S & \text{Line 2,15 and Lem. 10} \\
\Rightarrow \gamma_i(post^*_{\hat{N}}(\widehat{m_0})) \subseteq S & \text{def of } \mathcal{R}_i \text{ in Alg. 2} \\
\Rightarrow post^*_N(m_0) \subseteq S & \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}}, \text{ Prop. 2} \quad \square
\end{array}
$$

We need intermediary results (Prop. 5 and Lem. 12) to establish the completeness of Algorithm 2. The following result is about greatest fixpoints.

Proposition 5. *Let $\widehat{R} \subseteq R \subseteq \mathbb{N}^k$ such that $R \subseteq \widetilde{pre}(R)$ and $\widehat{R} \subseteq \widetilde{pre}(\widehat{R})$.*

$$gfp\lambda X.\, (S \cap \widehat{R} \cap \widetilde{pre}(X)) = \widehat{R} \cap gfp\lambda X.\, (S \cap R \cap \widetilde{pre}(X)) \ .$$

Lemma 12. *Consider Algorithm 2 with input a Petri net $N = (P, T, F, m_0)$, and $\leqslant$-dc-set S. If $post^*_N(m_0) \subseteq S$ then for any value of i we have $post^*_N(m_0) \subseteq Z_i$.*

Proof. The proof is by induction on i.
base case. Trivial since line 1 defines Z_0 to be S.

inductive case. Line 9 shows that $\gamma_i(\mathcal{S}_i)$ overapproximates $\mathit{gfp}\lambda X.\, Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(X)$, hence that $Z_{i+1} \supseteq \mathit{gfp}\lambda X.\, Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(X)$ by line 14.

$$
\begin{array}{lr}
post_N^*(m_0) \subseteq Z_i & \text{hyp} \\
\Rightarrow post_N^*(m_0) \subseteq \mathit{gfp} Z_i \cap \widetilde{pre}_N(X) & \text{def of } post_N^*(m_0)\text{, [12, Thm. 8]} \\
\Rightarrow post_N^*(m_0) \subseteq \gamma_i(\mathcal{R}_i) \cap \mathit{gfp} Z_i \cap \widetilde{pre}_N(X) & post_N^*(m_0) \subseteq \gamma_i(\mathcal{R}_i) \\
\Leftrightarrow post_N^*(m_0) \subseteq \mathit{gfp} Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(X) & \text{Prop. 5, def of } \mathcal{R}_i \\
\Rightarrow post_N^*(m_0) \subseteq Z_{i+1} & \text{by above} \quad \square
\end{array}
$$

Proposition 6 (Completeness). *If Algorithm 2 says "KO" then we have*

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

Proof. If Algorithm says "KO" then

$$
\begin{array}{lr}
\widehat{m_0} \nsubseteq \mathcal{S}_i & \\
\Leftrightarrow \alpha(m_0) \nsubseteq \mathcal{S}_i & \text{def of } \widehat{m_0} \\
\Leftrightarrow m_0 \nsubseteq \gamma_i(\mathcal{S}_i) & \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \\
\Leftrightarrow m_0 \nsubseteq \gamma_i(\mathit{gfp}\lambda X.\, \alpha_i(Z_i) \cap \mathcal{R}_i \cap \bigcap_{t \in T} \alpha_i \circ \widetilde{pre}_N[t] \circ \gamma_i(X)) & \text{def of } \mathcal{S}_i \\
\Rightarrow m_0 \nsubseteq \gamma_i(\mathit{gfp}\lambda X.\, \alpha_i(Z_i) \cap \alpha_i \circ \gamma_i(\mathcal{R}_i) \cap \bigcap_{t \in T} \alpha_i \circ \widetilde{pre}_N[t] \circ \gamma_i(X)) & \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \\
\Rightarrow m_0 \nsubseteq \gamma_i(\mathit{gfp}\lambda X.\, \alpha_i(Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(\gamma_i(X)))) & \alpha(\cap \cdot) \subseteq \cap \alpha(\cdot) \\
\Rightarrow m_0 \nsubseteq \mathit{gfp}\lambda X.\, Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(X) & \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \\
\Leftrightarrow \mathit{lfp}\lambda X.\, m_0 \cup post(X) \nsubseteq Z_i \cap \gamma_i(\mathcal{R}_i) & \text{[12, Thm. 4]} \\
\Rightarrow \mathit{lfp}\lambda X.\, m_0 \cup post(X) \nsubseteq Z_i & post_N^*(m_0) \subseteq \gamma_i(\mathcal{R}_i) \\
\Rightarrow \mathit{lfp}\lambda X.\, m_0 \cup post(X) \nsubseteq S & \text{Lem. 12} \quad \square
\end{array}
$$

Proposition 7 (Termination). *Algorithm 2 terminates.*

Proof. It is clear by line 14 that we have

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

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

$$
\begin{array}{lr}
Z_{i+1} = Z_i \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(\gamma_i(\mathcal{S}_i)) & \text{def of } Z_{i+1} \\
\Rightarrow Z_{i+1} \subseteq S \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(\gamma_i(\mathcal{S}_i)) & Z_i \subseteq Z_0 = S \\
\Rightarrow Z_{i+1} \subseteq S \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(Z_i) & \widetilde{pre}_N \text{ is monotonic,} \gamma_i(\mathcal{S}_i) \subseteq Z_i \\
\Leftrightarrow Z_{i+1} \subseteq S \cap \gamma_i(\mathcal{R}_i) \cap \widetilde{pre}_N(Z_{i+1}) & Z_{i+1} = Z_i \\
\Leftrightarrow Z_j \subseteq S \cap \gamma_{j-1}(\mathcal{R}_{j-1}) \cap \widetilde{pre}_N(Z_j) & \text{let } j = i+1 \\
\Rightarrow Z_j \subseteq S \wedge Z_j \subseteq \widetilde{pre}_N(Z_j) & \text{glb} \\
\Leftrightarrow Z_j \subseteq S \wedge post_N(Z_j) \subseteq Z_j & \text{(Gc) p. 4} \\
\Rightarrow \alpha_j(Z_j) \subseteq \alpha_j(S) \wedge \alpha_j \circ post_N(Z_j) \subseteq \alpha_j(Z_j) & \alpha_j \text{ is monotonic} \\
\Rightarrow \alpha_j(Z_j) \subseteq \alpha_j(S) \wedge \alpha_j \circ post_N(\gamma_j \circ \alpha_j(Z_j)) \subseteq \alpha_j(Z_j) & \text{line 15, Lem. 10} \\
\Rightarrow \alpha_j(Z_j) \subseteq \alpha_j(S) \wedge post_{\hat{N}}(\alpha_j(Z_j)) \subseteq \alpha_j(Z_j) & \text{Prop. 2}
\end{array}
$$

Then, either $\widehat{m_0} \subseteq \alpha_j(Z_j)$ and so [12, Thm. 4] shows that

$$
\begin{array}{ll}
lfp\lambda X.\, \widehat{m_0} \cup post_{\hat{N}}(X) \subseteq \alpha_j(S) & \\
\Rightarrow \gamma_j(lfp\lambda X.\, \widehat{m_0} \cup post_{\hat{N}}(X)) \subseteq \gamma_j \circ \alpha_j(S) & \\
\Rightarrow \gamma_j(lfp\lambda X.\, \widehat{m_0} \cup post_{\hat{N}}(X)) \subseteq S & \text{by line 2,15 and Lem. 3}
\end{array}
$$

and line 6 shows that the algorithm terminates. Or we have,

$$
\begin{array}{ll}
\widehat{m_0} \nsubseteq \alpha_j(Z_j) & \\
\Rightarrow \widehat{m_0} \nsubseteq S_j & S_j \subseteq \alpha_j(Z_j) \text{ by line 9}
\end{array}
$$

and line 10 shows that the algorithm 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 $\langle \subseteq, DCS(\mathbb{N}^k) \rangle$ is a wqo. We conclude from A2, Lem. 5, $\leqslant$-dc-set are closed to $\widetilde{pre}$ and $\cap$ that for any value of i in Alg. 2 we have $Z_i \in DCS(\mathbb{N}^k)$. However $\leqslant$ defines a wqo and following [14, Lem. 2] there is no infinite strictly decreasing sequence of $\langle \subseteq, DCS(\mathbb{N}^k) \rangle$, hence a contradiction. □

7 Experimental results

We implemented Alg. 2 in C using the symbolic data structure of [17] to represent and manipulate sets of markings. We used, for the model-checker referenced at line 5, the algorithm of [15].

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. *Parameterized systems* describe systems where we have a parameterized number of resources: ME [5, Fig. 1], MultiME (Fig. 1 of Sect. 2), CSM [18, Fig. 76, page 154], FMS [19], the mesh 2x2 of [18, 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. For instance, to prove that we never reach a marking

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.

	Example	**Var**	**Cvar**	**Ref**	**time**
Unbounded PN	ME	5	4	3	0.02
	multiME	12	5	3	2.69
	FMS	22	4	3	8.30
	CSM	14	9	4	11.78
	mesh2x2	32	9	4	340
	mesh3x2	52	9	4	3357
Bounded PN	**Example**	**Var**	**Cvar**	**Ref**	**time**
	lamport	11	9	4	8.50
	dekker	16	15	4	60.2
	peterson	14	12	5	21.5

with more than one token in the set of places $\{p_4, p_5\}$ in the MultiME (Fig. 1) our algorithm finds that the following partition $\{\{p_1, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}\}, \{p_2\}, \{p_3\}, \{p_4, p_5\}\}$ is sufficient to establish the mutual exclusion between p_4 and p_5. So our algorithm manipulates subsets of $\mathbb{N}^4$ instead of subsets of $\mathbb{N}^{12}$.

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 parameterized system and are abstracted by our algorithm. Hence, all the parameterized systems are analysed building an abstract Petri net with few places.

The other 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, (ii) and 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.

References

1. German, S.M., Sistla, A.P.: Reasoning about Systems with Many Processes. Journal of ACM **39**(3) (1992) 675–735 [1]

2. Karp, R.M., Miller, R.E.: Parallel program schemata. Journal of Comput. Syst. Sci. **3**(2) (1969) 147–195 [1, 5, 14]
3. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theoretical Computer Science **256**(1-2) (2001) 63–92 [1]
4. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.K.: General decidability theorems for infinite-state systems. In: Proc. of the 11th Annual IEEE Symp. on Logic in Computer Science (LICS), IEEE Computer Society (1996) 313–321 [1, 5, 15]
5. Delzanno, G., Raskin, J.F., Van Begin, L.: Attacking Symbolic State Explosion. In: Proc. of the 13th Intl. Conf. on Computer Aided Verification (CAV 2001). Volume 2102 of LNCS., Springer (2001) 298–310 [1, 18]
6. Van Begin, L.: Efficient Verification of Counting Abstractions for Parametric systems. PhD thesis, Université Libre de Bruxelles, Belgium (2003) [1]
7. Abdulla, P.A., Iyer, S.P., Nylén, A.: Unfoldings of unbounded petri nets. In: Proc. of the 12th Intl. Conf. on Computer Aided Verification (CAV). Volume 1855 of LNCS., Springer (2000) 495–507 [1]
8. Grahlmann, B.: The PEP Tool. In: Proceedings of the 9th Conference on Computer Aided Verification (CAV'97). Volume 1254 of LNCS., Springer (1997) 440–443 [1]
9. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Conf. Record of the Sixth Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, ACM Press (1979) 269–282 [2, 7, 8]
10. 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 [2]
11. Berthelot, G., Roucairol, G., Valk, R.: Reductions of nets and parallel prgrams. In: Advanced Course: Net Theory and Applications. Volume 84 of LNCS., Springer (1975) 277–290 [3]
12. Cousot, P.: Partial completeness of abstract fixpoint checking, invited paper. In: Proc. of the 4th Intl. Symp. on Abstraction, Reformulations and Approximation (SARA). Volume 1864 of LNAI., Springer-Verlag (2000) 1–25 [4, 5, 17, 18]
13. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. Amer. J. Math. **35** (1913) 413–422 [5]
14. Ganty, P., Raskin, J.F., Van Begin, L.: A complete abstract interpretation framework for coverability properties of WSTS. In: Proc. of Verification, Model Checking and Abstract Interpretation (VMCAI). Volume 3855 of LNCS., Springer-Verlag (2006) 49–64 [5, 14, 18]
15. Geeraerts, G., Raskin, J.F., Van Begin, L.: Expand, enlarge and check: new algorithms for the coverability problem of WSTS. In: Proc. of the 24th Intl. Conf. on Fondation of Software Technology and Theoretical Computer Science (FSTTCS). Volume 3328 of LNCS., Springer (2004) 287–298 [5, 14, 18]
16. Burris, S., Sankappanavar, H.P.: A Course in Universal Algebra. Springer-Verlag, New York (1981) [6, 9]
17. Ganty, P., Meuter, C., Delzanno, G., Kalyon, G., Raskin, J.F., Van Begin, L.: Symbolic data structure for sets of k-uples. Technical report (2007) [18]
18. Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. Series in Parallel Computing (1995) [18]
19. Ciardo, G., Miner, A.: Storage Alternatives for Large Structured State Space. In: Proc. of the 9th Intl. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (Tools). Volume 1245 of LNCS., Springer (1997) 44–57 [18]