A Complete Abstract Interpretation Framework for Coverability Properties of WSTS[⋆]

Pierre Ganty[1,⋆⋆], Jean-François Raskin[1], and Laurent Van Begin[2]

[1] Département d'Informatique, Université Libre de Bruxelles
{pganty, jraskin}@ulb.ac.be
[2] LIAFA, Université Paris 7
lvbegin@liafa.jussieu.fr

Abstract. We present an abstract interpretation based approach to solve the coverability problem of well-structured transition systems. Our approach distinguishes from other attempts in that (1) we solve this problem for the whole class of well-structured transition systems using a forward algorithm. So, our algorithm has to deal with possibly infinite downward closed sets. (2) Whereas other approaches have a non generic representation for downward closed sets of states, which turns out to be hard to devise in practice, we introduce a generic representation requiring no additional effort of implementation.

1 Introduction

Model-checking is nowadays widely accepted as a powerful technique for the automatic verification of reactive systems that have natural finite state abstractions. However, many reactive systems are only naturally modeled as infinite-state systems. This is why a large research effort was done in the recent years to allow the direct application of model-checking techniques to infinite-state models. This research line has shown successes for several interesting classes of infinite-state systems, for example: timed automata [1], hybrid automata [2], FIFO channel systems [3, 4], extended Petri nets [5, 6], broadcast protocols [7], etc.

General decidability results hold for a large class of infinite-state systems called the *well-structured transition systems*, WSTS for short. WSTS are transition systems whose sets of states are well-quasi ordered and whose transition relations enjoy a monotonicity property with respect to the well-quasi order. Examples of WSTS are Petri nets [8], monotonic extensions of Petri nets (Petri nets with transfer arcs [9], Petri nets with reset arcs [10], and Petri nets with non-blocking arcs [11]), broadcast protocols [12], lossy channel systems [3]. For all those classes of infinite-state systems, we know that an interesting and large class of *safety properties* are decidable by reduction to the *coverability problem*. The coverability problem is defined as follows: "given a WSTS for the well-quasi

[⋆] Supported by the FRFC project "Centre Fédéré en Vérification" funded by the Belgian National Science Foundation (FNRS) under grant nr 2.4530.02.

[⋆⋆] Pierre Ganty is supported by the FNRS under a FRIA grant.

E.A. Emerson and K.S. Namjoshi (Eds.): VMCAI 2006, LNCS 3855, pp. 49–64, 2006.

order $\succeq$, and two states c_1 and c_2, does there exist a state c_3 which is reachable from c_1 and such that $c_3 \succeq c_2$?" (in that context, we say that c_3 covers c_2).

Broadly speaking, there are two ways to solve the coverability problem for WSTS. The first way to solve the coverability problem is to explore *backwardly* the transition system by iterating the *pre* operator[1] starting from the set of states that are greater or equal to c_2. This simple procedure is effective when very mild assumptions are met. In fact, for any well-quasi ordered set $(X, \succeq)$, the following nice property holds: every $\succeq$-upward closed[2] set can be *finitely* represented using its finite set of minimal elements[3]. This generic representation of $\succeq$-upward closed set is adequate as union and inclusion are effective. The only further property that is needed for the procedure to be effective is that given a finite set of minimal elements M defining an $\succeq$-upward closed set U, it must be possible to compute the finite set of minimal elements M' representing $pre(U)$. Higman's lemma [13] on well-quasi orders ensure the termination of this procedure.

The second way is to explore forwardly the transition system from the initial state c_1. Here, the situation is more complicated. A saturation method that iterates the *post* operator[4] from c_0 can not lead to an algorithm as the reachability problem is undecidable for WSTS. Recently, we have shown that the coverability problem can be decided in a forward way by constructing two sequences of abstractions of the reachable states of the system, one from below and one from above [14]. The sequence of abstractions from below allows us to detect positive instances of the coverability problem and it is simply the bounded iteration of *post* from the initial state. The abstraction from above is the iteration of an overapproximation of *post* over downward closed set of states that becomes more and more precise. This sequence allows us to decide negative instances of the problem. This schema of algorithm is general but to be applicable to a given class of WSTS, the user has to provide a, so called, *adequate domain of limits.* This set is in fact a (usually infinite) set of abstract values that allows to represent any downward closed set. The situation is less satisfactory than for upward closed set where there exists, as we have seen above, a simple and generic way to represent upward closed set by sets of minimal elements. Such a generic way of representing downward closed sets was missing and this problem is solved here.

The contributions of this paper are as follows. First, we show that for any well-quasi ordered set, there exists a generic and effective representation of downward closed sets. To the best of our knowledge, this is the first time that such a generic representation is proposed. An attempt in that direction was taken in [15] but the result is a theory for designing symbolic representation of downward closed sets

[1] A function that returns all the states that have a one-step successor in a given set of states.

[2] A set S is upward (resp. downward) closed if for any c such that $c \succeq s$ (resp. $c \preceq s$) for some $s \in S$ we have $c \in S$.

[3] Or a finite set of its minimal elements if $\succeq$ is not a partial order.

[4] A function that returns all the one-step successors states of a given set of states.

and not a generic symbolic representation of such sets. As a consequence, their theory has to be instantiated for the particular class of WSTS that is targeted and this is not a trivial task. Second, as downward closed sets are abstractions for sets of reachable states in the forward algorithm, we formalize our generic representation of downward closed set as a generic abstract domain. This allow us to rephrase in a simpler way the forward algorithm, first proposed in [14], in the context of abstract interpretation. Third, we show how to automatically refine the abstract domain in order to obtain, in an efficient way, overapproximations that are guaranteed to be sufficiently precise to decide the coverability problem.

Our paper is organized as follows. Section 2 presents some preliminaries. Section 3 introduces the generic representation of downward closed sets. In Section 4 we will be concerned with the abstract interpretation of WSTS. Section 5 is devoted to the refinement of the abstract domain. Section 6 shows on an example how these techniques work. A version of the paper containing all proofs is available at [16].

2 Preliminaries

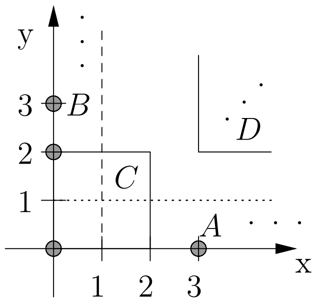
2.1 Well-Quasi Ordered Sets

A *preorder* $\succeq$ is a binary relation over a set X which is reflexive, and transitive. The preorder $\succeq$ is a *well-quasi order* (wqo for short) if there is no infinite sequence $x_0, x_1, \ldots$, such that $x_i \not\succeq x_j$ for all $i > j \geq 0$. A set $M \subseteq X$ is said to be *canonical* if for any distinct $x, y \in M$ we have $x \not\succeq y$. We say that $M \subseteq S$ is a *minor set* of $S \subseteq X$, if for all $x \in S$ there exists $y \in M$ such that $x \succeq y$, and M is canonical.

Lemma 1 (From [17]). *Let $(X, \succeq)$ be a well-quasi ordered set (wqo-set for short). For any set $S \subseteq X$, S has at least one finite minor set M.*

We use min to denote a function which, given a set $S \subseteq X$, returns a minor set of S. Let $(X, \succeq)$ be a wqo-set, we call $x{\downarrow} = \{x' \in X \mid x \succeq x'\}$ and $x{\uparrow} = \{x' \in X \mid x' \succeq x\}$ the $\succeq$*-downward closure* and $\succeq$*-upward closure* of $x \in X$, respectively. This definition is naturally extended to sets in X. We define a set $S \subseteq X$ to be a $\succeq$-downward closed set ($\succeq$*-dc-set* for short), respectively $\succeq$-upward closed set ($\succeq$*-uc-set* for short), iff $S{\downarrow} = S$, respectively $S{\uparrow} = S$. Examples of such sets are given in Fig. 1. For any wqo-set $(X, \succeq)$, we define $DCS(X)$ ($UCS(X)$) to be the set of all $\succeq$-dc-sets ($\succeq$-uc-sets) in X. For any $x \in X$ we define the $\succeq$-equivalence class of x, denoted $[x]$, to be the set $x{\uparrow} \cap x{\downarrow}$, i.e. the set of elements that are $\succeq$-equivalent to x. For A and B subsets of X, we say that $A \equiv B$ if $A{\uparrow} = B{\uparrow}$. Observe that $A \equiv B$ iff for all $a \in A$ there is a $b \in B$ such that $a \succeq b$, and vice versa. We now recall a well-known lemma on $\succeq$-uc-sets and $\succeq$-dc-sets.

Lemma 2 (From [17]). *Let $(X, \succeq)$ a wqo-set and an infinite sequence of $\succeq$-uc-set $U_0 U_1 \ldots$ such that $\forall i \geq 0\colon U_i \subseteq U_{i+1}$. There exists $j \geq 0\colon \forall j' \geq j\colon U_j = U_{j'}$. Symmetrically, given an infinite sequence of $\succeq$-dc-sets $D_0 D_1 \ldots$ such that $\forall i \geq 0\colon D_i \supseteq D_{i+1}$, there exists $j \geq 0\colon \forall j' \geq j\colon D_j = D_{j'}$.*



The wqo $\succeq$ is defined as follows $(a_1, a_2) \succeq (b_1, b_2)$ if and only if $a_1 \geq b_1$ and $a_2 \geq b_2$. The $\succeq$-dc-sets A and B are infinite $\succeq$-dc-set : $A = \{(x, y) \in \mathbb{N}^2 \mid y \leq 1\}$, $B = \{(x, y) \in \mathbb{N}^2 \mid x \leq 1\}$. On the contrary, the $\succeq$-dc-set $C = \{(x, y) \in \mathbb{N}^2 \mid x \leq 2 \wedge y \leq 2\}$ is finite. The $\succeq$-uc-set D is given by $\{(x, y) \in \mathbb{N}^2 \mid x \geq 3 \wedge y \geq 2\}$. Note that D has exactly one minor set since $\succeq$ is a partial order.

Fig. 1. $\succeq$-dc-sets and $\succeq$-uc-sets in $\mathbb{N}^2$

We now introduce a lemma stating several facts about sets and their closure. These facts are merely of technical interest and will be used subsequently.

Lemma 3.

1. *For any $S, S' \subseteq X$, $S\!\downarrow \cap S'\!\uparrow \neq \emptyset \Leftrightarrow S\!\downarrow \cap S' \neq \emptyset \Leftrightarrow S \cap S'\!\uparrow \neq \emptyset$.*
2. *For any $S, S' \subseteq X$, $S\!\uparrow \subseteq S'\!\uparrow \Leftrightarrow \forall s \in S\ \exists s' \in S' \colon s \succeq s'$.*
3. *$\forall s \in X, S \in UCS(X) \colon s \in S \Leftrightarrow \exists s' \in \mathsf{min}(S) \colon s \succeq s'$.*

Lemma 3.2 and 3.3 suggest an effective representation of $\succeq$-uc-sets: every $\succeq$-uc-set U can be finitely represented by $\mathsf{min}(U)$. For decidable well-quasi order $\succeq$, this readily gives us an effective procedure to check inclusion between two $\succeq$-uc-sets, to check membership and to compute union [18].

Notations. Sometimes we write s instead of the set $\{s\}$. Unless otherwise stated the transitive and reflexive closure f^* of a function f such that its domain and co-domain coincide is given by $\bigcup_{i \geq 0} f^i$ where f^0 is the identity and $f^{i+1} = f^i \circ f$. Finally, let us recall the following property on sets that we will use without mention in our proofs: $A \subseteq B$ iff $A \cap (X \setminus B) = \emptyset$.

2.2 Well-Structured Transitions Systems

In this paper we follow [19] in the definition of well-structured transition systems.

Definition 1. *A* well-structured transition system *(*WSTS*) $\mathcal{S}$ is a tuple $(X, \delta, \succeq)$ where X is a (possibly) infinite set of states, $\delta \subseteq X \times X$ is a transition relation between states — we use the notation $x \rightarrow x'$ if $(x, x') \in \delta$ —, and $\succeq \subseteq X \times X$ is a preorder between states such that the two following conditions hold: (i) $\succeq$ is a wqo; and (ii) $\forall x_1, x_2, x_3\ \exists x_4 \colon (x_3 \succeq x_1 \wedge x_1 \rightarrow x_2) \Rightarrow (x_3 \rightarrow^* x_4 \wedge x_4 \succeq x_2)$, where $\rightarrow^*$ is the reflexive and transitive closure of the transition relation (upward compatibility)*[5]*. Moreover, we define an* initialized WSTS *(*IWSTS*) to be a pair $(\mathcal{S}, x_0)$ where $\mathcal{S} = (X, \delta, \succeq)$ is a* WSTS *and $x_0 \in X$ is the* initial state. *We adhere to the convention that if $\mathcal{S}_0$ is an* IWSTS *then $\mathcal{S}$ is its* WSTS.

[5] Upward compatibility is more general than the compatibility used in [17].

Let $\mathcal{S} = (X, \delta, \succeq)$ be a WSTS and $T \subseteq X$, $post[\mathcal{S}](T) \overset{\text{def}}{=} \{x' \mid \exists x \in T\colon x \rightarrow x'\}$. Analogously, we define $pre[\mathcal{S}](T)$ as $\{x \mid \exists x' \in T\colon x \rightarrow x'\}$. We define $minpre[\mathcal{S}](T) \overset{\text{def}}{=} \mathsf{min}((pre[\mathcal{S}](T\uparrow))\uparrow)$. To shorten notation, we write *pre*,*minpre* and *post* if the WSTS is clear from the context. The following definition follows [17].

Definition 2. *An* effective WSTS *is a* WSTS $\mathcal{S} = (X, \delta, \succeq)$ *where both* $\succeq$ *and* $\rightarrow$ *are decidable and for all* $x \in X$: $minpre[\mathcal{S}](x)$ *is computable.*

2.3 The Coverability Problem

The verification of safety properties on IWSTS reduces to the so called coverability problem.

Problem 1. The *coverability problem* for IWSTS is defined as follows: "*Given an* IWSTS $((X, \delta, \succeq), x_0)$ *and* $\mathsf{bad} \in UCS(X)$, $post^*(x_0) \cap \mathsf{bad} = \emptyset$*?* "

In general, bad is an upward closed set of states where errors occur.

Two solutions to the coverability problem can be found in the literature. The first one (see [17, 19]) is a backward approach based on the following two lemmas:

Lemma 4 (From [19]). *Given a* WSTS $\mathcal{S} = (X, \delta, \succeq)$ *and* $U \in UCS(X)$*, (a)* $pre^*(U) \in UCS(X)$*, and (b)* $minpre^*(\mathsf{min}(U))\uparrow = pre^*(U)$.

Lemma 4, together with Lemma 1 and 2, show how to (symbolically) compute the (possibly) infinite set $pre^*(U)$ using the minor sets of $\succeq$-uc-sets. Once $pre^*(U)$ is computed, or rather a finite representation using one of its minor set, one can decide the coverability problem by testing if the initial state is in $pre^*(U)$ by using Lemma 3.3.

The second approach is a forward approach based on the notion of *covering set* [20, 12]. The covering set $Cover(\mathcal{S}_0)$ of an IWSTS $\mathcal{S}_0 = (\mathcal{S}, x_0)$ is given by $Cover(\mathcal{S}_0) \overset{\text{def}}{=} post^*(x_0)\downarrow$. The following lemma shows the usefulness of covering sets to solve the coverability problem:

Lemma 5. *Given an* IWSTS $\mathcal{S}_0 = ((X, \delta, \succeq), x_0)$ *and* $\mathsf{bad} \in UCS(X)$, $Cover(\mathcal{S}_0) \cap \mathsf{bad} = \emptyset$ *if and only if* $post^*(x_0) \cap \mathsf{bad} = \emptyset$.

As already mentioned in the introduction, there are two difficulties to overcome when trying to design a forward algorithm for the coverability problem:

1. Currently, there are no generic way to effectively represent and manipulate $\succeq$-dc-sets (as the one shown above for $\succeq$-uc-sets). So, for every wqo-set $(X, \succeq)$ one has to design a symbolic representation for the sets in $DCS(X)$.
2. The set $Cover(\mathcal{S}_0)$ is in general not effectively constructible, see [10] for details. As a consequence, all the algorithms based on its construction (except the well-known Karp-Miller algorithm on Petri nets) may fail to terminate.

To overcome those two difficulties:

1. In [15], the authors propose a methodology to design a symbolic representation of dc-sets. However the design of such a symbolic data-structure is far from being trivial.
2. The authors of this paper proposed, in [14], an algorithmic schema called *expand, enlarge and check* which can be instantiated for any class of WSTS as long as a symbolic representation of dc-sets is provided (called there an adequate set of limits).

In this paper, we provide, in our opinion, a much more satisfactory answer to those two difficulties by providing, in the form of a generic abstract domain and a generic abstract analysis, a completely generic algorithm to solve the coverability problem for WSTS.

3 A Generic Abstract Domain

In this section, we present a parametrized abstract domain that allows us to represent any $\succeq$-dc-set in a wqo-set $(X, \succeq)$. The parameter D is a finite subset of X and it defines the precision of the abstract domain. We also show that this parametrized abstract domain enjoys the following properties: (i) our parametrized abstract domain defines a complete lattice, (ii) we define an abstraction and a concretisation function that is shown to be a Galois insertion, (iii) any $\succeq$-dc-set can be exactly represented by our parametrized abstract domain provided an adequate value for the parameter D is used, and (iv) each $\succeq$-dc-set has a finite representation.

Recall that the powerset lattice $PL(A)$ associated to a set A is the complete lattice having the powerset of A as carrier, and union and intersection as least upper bound and greatest lower bound, respectively. In our setting the concrete lattice is the powerset lattice $PL(X)$ of the set of states X.

Fix a finite set $D \subseteq X$ which is called the *finite domain*, the abstract lattice $DPL(D)$ has $DCS(D)$ as a carrier, $\sqcup_D$ as the *least upper bound* operator, $\sqcap_D$ as the *greatest lower bound* operator, and D and $\emptyset$ are the $\sqsubseteq_D$-maximal and $\sqsubseteq_D$-minimal element, respectively. We define the relation $\sqsubseteq_D$ over $DCS(D) \times DCS(D)$ such that for all $P_1, P_2 \in DCS(D)$: $P_1 \sqsubseteq_D P_2$ if and only if $P_1 \subseteq P_2$, $P_1 \sqcup_D P_2 \stackrel{\text{def}}{=} P_1 \cup P_2$, $P_1 \sqcap_D P_2 \stackrel{\text{def}}{=} P_1 \cap P_2$. Notice that $DPL(D)$ is complete because the union and the intersection operations are closed in $DPL(D)$. Given an abstract lattice $DPL(D)$, the abstraction and concretisation mappings are given as follows:

$$\begin{aligned}
\forall E \in PL(X)\colon &\qquad \alpha[D](E) \stackrel{\text{def}}{=} E{\downarrow} \cap D \\
\forall P \in DPL(D)\colon &\qquad \gamma[D](P) \stackrel{\text{def}}{=} \{x \in X \mid x{\downarrow} \cap D \subseteq P\} \ .
\end{aligned}$$

The set between brackets defines the *parameter* of the function and the set between parentheses is its *argument*. For simplicity of notation, we also write $\gamma(P)$, $\alpha(E)$, $\sqsubseteq$, $\sqcup$ and $\sqcap$ if the parameter is clear from the context.

We next show through an example that the finite domain D actually parametrizes the precision of the abstract domain with respect to the concrete domain.

Example 1. Let us consider the $\succeq$-dc-sets of Fig. 1 and consider the following finite domain $D = \{(0,0),(3,0),(0,2),(0,3)\}$ depicted by the grey dots. Applying α on the $\succeq$-dc-sets A, B and C give, respectively, the (abstract) sets $\alpha(A) = \{(0,0),(3,0)\}, \alpha(B) = \{(0,0),(0,2),(0,3)\}, \alpha(C) = \{(0,0),(0,2)\}$. A and C are exactly represented, i.e. $\gamma(\alpha(A)) = A$ and $\gamma(\alpha(C)) = C$, but B is not: $\gamma(\alpha(B)) = \{(x,y) \in \mathbb{N}^2 \mid x \leq 2\}$. But, if we add $(2,0)$ to D then B becomes representable.

This generic abstract domain is a generalization of the ideas exposed in [21] for finite states systems.

Fix a finite domain D, the *concrete* $PL(X)$ and *abstract* $DPL(D)$ domains and the *abstraction* $\alpha\colon PL(X) \mapsto DPL(D)$ and *concretisation* $\gamma\colon DPL(D) \mapsto PL(X)$ maps form a *Galois insertion*, denoted by $PL(X) \underset{\gamma}{\overset{\alpha}{\rightleftharpoons}} DPL(D)$.

Proposition 1. *For every finite domain D, $PL(X) \underset{\gamma}{\overset{\alpha}{\rightleftharpoons}} DPL(D)$.*

Proof. Fix a finite domain D. It follows immediately from the definitions that α is monotonic (i.e., $C \subseteq C'$ implies $\alpha(C) \sqsubseteq \alpha(C')$) and γ as well. Indeed, $\gamma(P_1) \subseteq \gamma(P_2) \Leftrightarrow \{c \mid c\!\downarrow \cap D \subseteq P_1\} \subseteq \{c \mid c\!\downarrow \cap D \subseteq P_2\} \Leftrightarrow P_1 \subseteq P_2 \Leftrightarrow P_1 \sqsubseteq P_2$. So, it suffices to prove (a) and (b) below:

(a) $C \subseteq (\gamma \circ \alpha)(C)$ for every $C \in PL(X)$.

$$\begin{aligned}(\gamma \circ \alpha)(C) &= \{c \in X \mid c\!\downarrow \cap D \subseteq C\!\downarrow \cap D\}\\ &\supseteq \{c \in C \mid c\!\downarrow \cap D \subseteq C\!\downarrow \cap D\}\\ &= C\end{aligned}$$

(b) $(\alpha \circ \gamma)(P) = P$ for every $P \in DPL(D)$.

$$\begin{aligned}(\alpha \circ \gamma)(P) &= \{c \mid c\!\downarrow \cap D \subseteq P\}\!\downarrow \cap D &&\\ &= \{c \mid c\!\downarrow \cap D \subseteq P\} \cap D && \gamma(P)\!\downarrow = \gamma(P)\\ &= \{c \in D \mid c\!\downarrow \cap D \subseteq P\} &&\\ &= P && P \subseteq D \text{ and } P \in DCS(D) \qquad \square\end{aligned}$$

We now prove some properties on the precision of our abstract domain. The next lemma states that any $\succeq$-dc-set of X can be represented exactly using a finite domain D and a set $P \in DCS(D)$.

Lemma 6 (Completeness of the abstract domain). *For each $E \in DCS(X)$ there exists a finite domain D such that $(\gamma \circ \alpha)(E) = E$.*

Proof. Given E, we define the finite domain D to be $D = \mathsf{min}(X \setminus E)$. We prove $(\gamma \circ \alpha)(E) = E$.

Let us show that $(\gamma \circ \alpha)(E) \subseteq E$. For that, suppose by contradiction that there exists $p \in (\gamma \circ \alpha)(E) \wedge p \notin E$.

$$
\begin{aligned}
& p \notin E \\
& \Leftrightarrow p{\downarrow} \not\subseteq E && E \in \mathit{DCS}(X) \\
& \Leftrightarrow p{\downarrow} \cap (X \setminus E) \neq \emptyset \\
& \Leftrightarrow p{\downarrow} \cap \mathsf{min}(X \setminus E) \neq \emptyset && \text{Lem. 3.1} \\
& \Leftrightarrow \exists p' \colon p' \in p{\downarrow} \wedge p' \in \mathsf{min}(X \setminus E) \\
& \Rightarrow \exists p' \colon p' \in p{\downarrow} \wedge p' \in \mathsf{min}(X \setminus E) \wedge \exists p'' \in [p'] \colon p'' \in D && \text{def. of } D \\
& \Leftrightarrow \exists p'' \colon p'' \in p{\downarrow} \wedge p'' \in D \wedge p'' \notin E && p, p' \succeq p''; p'{\downarrow} \cap E = \emptyset
\end{aligned}
\tag{1}
$$

$$
\begin{aligned}
& p \in (\gamma \circ \alpha)(E) \\
& \Leftrightarrow p{\downarrow} \cap D \subseteq \alpha(E) && \text{def. of } \gamma \\
& \Leftrightarrow p{\downarrow} \cap D \subseteq E{\downarrow} \cap D && \text{def. of } \alpha \\
& \Leftrightarrow p{\downarrow} \cap D \subseteq E \cap D && E \in \mathit{DCS}(X) \\
& \Leftrightarrow p{\downarrow} \cap D \subseteq E \\
& \Leftrightarrow \forall p' \colon p' \in p{\downarrow} \wedge p' \in D \Rightarrow p' \in E \\
& \Leftrightarrow \neg\neg \left(\forall p' \colon p' \in p{\downarrow} \wedge p' \in D \Rightarrow p' \in E\right) \\
& \Leftrightarrow \neg \left(\exists p' \colon p' \in p{\downarrow} \wedge p' \in D \wedge p' \notin E\right)
\end{aligned}
\tag{2}
$$

From (1) and (2) follows a contradiction.

$E \subseteq (\gamma \circ \alpha)(E)$ is immediate by property of Galois insertion. So, we have proved that $(\gamma \circ \alpha)(E) = E$. □

Remark 1. While previous lemma states that any $\succeq$-dc-set can be represented using an adequate finite domain D, there is usually no finite domain D which is able to represent all the $\succeq$-dc-sets. It should be pointed out that $\succeq$-dc-sets can be easily represented through their ($\succeq$-uc-set) complement, i.e. by using a finite set of minimal elements of their complement. However with this approach the manipulation of $\succeq$-dc-sets is not obvious. In particular, there is no generic way to compute the *post* operation applied on a $\succeq$-dc-set by manipulating its complement. Also, as $\mathit{Cover}(\mathcal{S}_0)$ is not constructible, it is, in some sense, useless to try to represent exactly the $\succeq$-dc-sets encountered during the forward exploration. On the other hand, we will see in Section 4 that our abstract domain allow us to define an effective and generic abstract post operator.

Hereunder, Proposition 3 shows that the more elements you put into the finite domain D, the more $\succeq$-dc-sets the abstract domain is able to represent exactly. Proposition 2, which is used in many proofs, provides an equivalent definition for $\gamma[D](P)$.

Proposition 2. *Fix a finite domain D, for every $P \in \mathit{DPL}(D)$ we have $\gamma(P) = X \setminus (D \setminus P){\uparrow}$.*

Proposition 3. *Fix two finite domains D and D' such that $D \subset D'$. For every $P \in \mathit{DPL}(D)$, there exists a $P' \in \mathit{DPL}(D')$ such that $\gamma[D](P) = \gamma[D'](P')$.*

Effectiveness. It is worth pointing that since we impose finiteness of D then $\sqcup_D$, $\sqcap_D$ are effective and $\sqsubseteq_D$ is decidable. So, given a finite domain D, the complete lattice $DPL(D)$ represents an effective way to manipulate (infinite) $\succeq$-dc-sets. Even if D is finite, it can be very large and so the abstract domain may be computationally expensive to manipulate. Compact data structures like Binary Decision Diagrams [22] and Sharing Trees [23, 18] may be necessary to use in practice.

In Sect. 5 we need to decide the intersection emptiness between an $\succeq$-uc-set and a $\succeq$-dc-set. In input of this problem we are given an effective representation of these two sets. Then we solve the problem using the result of Lemma 3.1 together with the following proposition.

Proposition 4. *Fix a finite domain D, for all $P \in DPL(D)$ there exists an effective procedure to answer the membership test, i.e. "given $c \in X$, does c belong to $\gamma(P)$?".*

4 Abstract Interpretation

In this section, we define the forward abstract interpretation of a WSTS using an abstract domain parametrized by D as defined in the previous section.

Let $\mathcal{S}$ be a WSTS and D be a finite domain, $post^{\#}[\mathcal{S}, D]\colon DPL(D) \mapsto DPL(D)$ is the function defined as follows: $post^{\#}[\mathcal{S}, D] \stackrel{\text{def}}{=} \lambda P.(\alpha[D] \circ post[\mathcal{S}] \circ \gamma[D])(P)$. The function $post^{\#}[\mathcal{S}, D]^*\colon DPL(D) \mapsto DPL(D)$ is defined as follows: $post^{\#}[\mathcal{S}, D]^* \stackrel{\text{def}}{=} \lambda P. \sqcup_{i\geq 0}\, post^{\#}[\mathcal{S}, D]^i(P)$. We shorten $post^{\#}[\mathcal{S}, D]$ to $post^{\#}$ and $post^{\#}[\mathcal{S}, D]^*$ to $(post^{\#})^*$ if the WSTS and the finite domain are clear from the context.

The following lemma establishes the soundness of our abstract interpretation of WSTS which follows by property of Galois connection:

Lemma 7. *Given a WSTS $(X, \delta, \succeq)$ with $I \subseteq X$ and a finite domain D, (i) $post(I) \subseteq (\gamma \circ post^{\#} \circ \alpha)(I)$ and (ii) $post^*(I) \subseteq (\gamma \circ (post^{\#})^* \circ \alpha)(I)$.*

The next proposition shows that we can improve the precision of the analysis by improving the precision of the abstract domain.

Proposition 5 ($post^{\#}$ Monotonicity). *Given a WSTS $\mathcal{S}= (X, \delta, \succeq)$, two finite domains D, D' with $D \subseteq D'$, and two sets $C, C' \subseteq X$ with $C \subseteq C'$, we have, (1) $(\gamma[D'] \circ post^{\#}[\mathcal{S}, D'] \circ \alpha[D'])(C) \subseteq (\gamma[D] \circ post^{\#}[\mathcal{S}, D] \circ \alpha[D])(C')$; and (2) $(\gamma[D'] \circ post^{\#}[\mathcal{S}, D']^* \circ \alpha[D'])(C) \subseteq (\gamma[D] \circ post^{\#}[\mathcal{S}, D]^* \circ \alpha[D])(C')$.*

Let us now show that if we fix a finite domain D, then $post^{\#}$ is computable for any effective WSTS but first we need the following lemma:

Lemma 8. *Given a WSTS $\mathcal{S}{=}(X, \delta, \succeq)$, $\forall x, x' {\in} X\colon x {\in} pre(x'{\uparrow}) \Leftrightarrow x' {\in} post(x){\downarrow}$.*

Proof. $x \in pre(x'{\uparrow}) \Leftrightarrow \exists x''\colon x'' \succeq x' \wedge x \rightarrow x'' \Leftrightarrow x' \in post(x){\downarrow}$. □

We have the following characterization of $post^{\#}$.

Proposition 6. *Fix a finite domain D, and an effective WSTS $\mathcal{S} = (X, \delta, \succeq)$. For every $x \in D$ and $P \in DPL(D)$:*

$$x \in post^{\#}(P) \Leftrightarrow (x \in D \wedge \neg(pre(x{\uparrow}) \subseteq (D \setminus P){\uparrow})) \quad .$$

The sets $\mathit{pre}(x\!\uparrow)\!\uparrow$ and $(D \setminus P)\!\uparrow$ are $\succeq$-uc-sets which have as finite minor set $\mathit{minpre}(x)$ and $(D \setminus P)$ respectively. Lemma 3.2 shows that if both $\mathit{minpre}(x)$ and $(D \setminus P)$ are finite sets and $\succeq$ is decidable then we have an *effective* procedure to decide if $\mathit{pre}(x\!\uparrow)\!\uparrow \subseteq (D \setminus P)\!\uparrow$ which is equivalent to $\mathit{pre}(x\!\uparrow) \subseteq (D \setminus P)\!\uparrow$. Furthermore, since the complete lattice $\mathit{DPL}(D)$ is finite, it follows that:

Corollary 1. *For any effective* IWSTS $\mathcal{S}_0 = (\mathcal{S}, x_0)$*, and any finite domain* D*,* $((\mathit{post}^{\#}[\mathcal{S}, D])^* \circ \alpha)(x_0)$ *can be effectively computed.*

5 Domain Refinements

In this section, we show that the abstract interpretation that we have defined previously can be made sufficiently precise to decide the coverability problem of (effective) IWSTS. We present two ways of achieving completeness of the abstract interpretation. Both are based on abstract domain refinement. The first (and naïve) way is through enumeration of finite domains. The enumerating algorithm shows that completeness is achievable by systematically enlarging the finite domain D. The second algorithm, which is more sophisticated, enlarges the finite domain D using abstract counter-examples.

5.1 Enumerate Finite Domains

In Sect. 3, we showed that any $\succeq$-dc-set can be represented using a well chosen domain (Lemma 6). In particular, the covering set can be represented using a finite domain D.

Hereunder, Theorem 1 asserts that the abstract interpretation of an IWSTS $\mathcal{S}_0$ using a finite domain D that allows to represent exactly the covering set of $\mathcal{S}_0$ leads to the construction of that set.

Theorem 1. *Given* $\mathit{Cover}(\mathcal{S}_0)$*, the covering set of an* IWSTS $\mathcal{S}_0$*, and some finite domain* D *such that there is* $\Theta \in \mathit{DPL}(D)$: $\gamma(\Theta) = \mathit{Cover}(\mathcal{S}_0)$*. For any* $P \in \mathit{DPL}(D)$ *such that* $P \sqsubseteq \Theta$ *we have* $(\gamma \circ (\mathit{post}^{\#})^*)(P) \subseteq \mathit{Cover}(\mathcal{S}_0)$*.*

Proof.

$$\begin{array}{ll}
\gamma(\Theta) = \mathit{Cover}(\mathcal{S}_0) & \text{by hypothesis} \\
\Rightarrow (\mathit{post} \circ \gamma)(\Theta) = \mathit{post}(\mathit{Cover}(\mathcal{S}_0)) & \text{monotonicity of } \mathit{post} \\
\Rightarrow (\mathit{post} \circ \gamma)(\Theta) \subseteq \mathit{Cover}(\mathcal{S}_0) & \mathit{post}(\mathit{Cover}(\mathcal{S}_0)) \subseteq \mathit{Cover}(\mathcal{S}_0) \\
\Rightarrow (\alpha \circ \mathit{post} \circ \gamma)(\Theta) \sqsubseteq \alpha(\mathit{Cover}(\mathcal{S}_0)) & \text{by monotonicity of } \alpha \\
\Leftrightarrow \mathit{post}^{\#}(\Theta) \sqsubseteq \alpha(\mathit{Cover}(\mathcal{S}_0)) & \text{def. of } \mathit{post}^{\#} \\
\Leftrightarrow \mathit{post}^{\#}(\Theta) \sqsubseteq \Theta & \gamma(\Theta) = \mathit{Cover}(\mathcal{S}_0), \Theta = (\alpha \circ \gamma)(\Theta) \quad (3)
\end{array}$$

Since $\mathit{post}^{\#}$ is a monotone function on a complete lattice, (3) shows that for any $P \sqsubseteq \Theta$ we have

$$\begin{array}{ll}
((\mathit{post}^{\#})^*)(P) \sqsubseteq \Theta & \\
\Rightarrow (\gamma \circ (\mathit{post}^{\#})^*)(P) \subseteq \gamma(\Theta) & \text{monotonicity of } \gamma \\
\Leftrightarrow (\gamma \circ (\mathit{post}^{\#})^*)(P) \subseteq \mathit{Cover}(\mathcal{S}_0) & \text{by hypothesis}
\end{array}$$

□

Thanks to this proposition and the results of [14] Algorithm 1 decides the coverability problem for an effective IWSTS $\mathcal{S}_0 = (\mathcal{S}, x_0)$ and a $\succeq$-uc-set bad. The main idea underlying the algorithm is to iteratively analyze an underapproximation of the reachable states (line 1) followed by an overapproximation (line 2). Positive instances of the coverability problem are decided by underapproximations and negative instances are decided by overapproximations. By enumeration of finite domains D_i and Theorem 1, it is ensured that our abstract interpretation will eventually become precise enough for the negative instances. For this algorithm to be effective, we only need the (mild) additional assumption that elements of X are enumerable.

Algorithm 1. Enumeration

```
   Input: An IWSTS $\mathcal{S}_0 = ((X, \delta, \succeq), x_0)$ and a set $\mathsf{bad} \in UCS(X)$
   for $D_i = D_0, D_1, \ldots$ an enumeration of the finite subsets of $X$ do
1      if $\exists x_0, \ldots, x_k \in D_i \colon x_0 \to \ldots \to x_k \wedge x_k \in \mathsf{bad}$ then
           return REACHABLE
2      else if $(\gamma[D_i] \circ (post^{\#}[\mathcal{S}, D_i])^* \circ \alpha[D_i])(x_0) \cap \mathsf{bad} = \emptyset$ then
           return UNREACHABLE
   end
```

In the next subsection, we show that this assumption can be dropped and propose a more sophisticated way to obtain a finite domain D which is precise enough to solve the coverability problem. Our refinement technique is based on the analysis of the states leading to bad.

5.2 Eliminate Overapproximations Leading to bad

Let us first consider the following lemma that is a first step towards completeness.

Lemma 9. *Given a* WSTS $(X, \delta, \succeq)$ *and a set* $\mathsf{bad} \in UCS(X)$ *fix a finite domain* D *and a set* $P' \in DPL(D)$ *such that* $post^{\#}(P') \sqsubseteq P'$ *and* $\mathsf{min}(pre^*(\mathsf{bad})) \cap \gamma(P') \subseteq D$. *For every* $P \in DPL(D)$ *such that* $P \sqsubseteq P'$ *we obtain* $\gamma(P) \cap pre^*(\mathsf{bad}) = \emptyset \Rightarrow \gamma(post^{\#}(P)) \cap pre^*(\mathsf{bad}) = \emptyset$.

Proof.

$$\begin{array}{lr}
\gamma(P) \cap pre^*(\mathsf{bad}) = \emptyset & \\
\Leftrightarrow (\downarrow \circ post \circ \gamma)(P) \cap pre^*(\mathsf{bad}) = \emptyset & \text{Lem. 8 and } pre(pre^*(\mathsf{bad})) = pre^*(\mathsf{bad}) \\
\Rightarrow (\alpha \circ post \circ \gamma)(P) \cap pre^*(\mathsf{bad}) = \emptyset & (\alpha \circ post \circ \gamma)(P) \subseteq (\downarrow \circ post \circ \gamma)(P) \\
\Leftrightarrow post^{\#}(P) \cap pre^*(\mathsf{bad}) = \emptyset & \text{def. of } post^{\#} \\
\Leftrightarrow pre^*(\mathsf{bad}) \subseteq (X \setminus post^{\#}(P)) &
\end{array}$$

So we have established

$$\gamma(P) \cap pre^*(\mathsf{bad}) = \emptyset \Rightarrow pre^*(\mathsf{bad}) \subseteq (X \setminus post^{\#}(P)) \ . \tag{4}$$

Moreover, we conclude from $P \sqsubseteq P'$ that $\mathit{post}^{\#}(P) \sqsubseteq \mathit{post}^{\#}(P')$ (by monotonicity of $\mathit{post}^{\#}$), hence that $\mathit{post}^{\#}(P) \sqsubseteq P'$ ($\mathit{post}^{\#}(P') \sqsubseteq P'$) and finally that $\gamma(\mathit{post}^{\#}(P)) \subseteq \gamma(P')$ (by monotonicity of γ).

Now, let us consider $\gamma(\mathit{post}^{\#}(P))$:

$$\begin{array}{lr}
\gamma(\mathit{post}^{\#}(P)) & \\
= \{c \mid c{\downarrow} \cap D \subseteq \mathit{post}^{\#}(P)\} & \text{definition of } \gamma \\
= \{c \in \gamma(P') \mid c{\downarrow} \cap D \subseteq \mathit{post}^{\#}(P)\} & \gamma(\mathit{post}^{\#}(P)) \subseteq \gamma(P') \\
\subseteq \{c \in \gamma(P') \mid c{\downarrow} \cap \mathsf{min}(\mathit{pre}^{*}(\mathsf{bad})) \cap \gamma(P') \subseteq \mathit{post}^{\#}(P)\} & \text{def. of } D \\
= \{c \in \gamma(P') \mid c{\downarrow} \cap \mathsf{min}(\mathit{pre}^{*}(\mathsf{bad})) \subseteq \mathit{post}^{\#}(P)\} & c \in \gamma(P') \text{ implies } c{\downarrow} \subseteq \gamma(P') \\
= \{c \in \gamma(P') \mid c{\downarrow} \cap \mathsf{min}(\mathit{pre}^{*}(\mathsf{bad})) \cap (X \setminus \mathit{post}^{\#}(P)) = \emptyset\} & \\
\subseteq \{c \in \gamma(P') \mid c{\downarrow} \cap \mathsf{min}(\mathit{pre}^{*}(\mathsf{bad})) \cap \mathit{pre}^{*}(\mathsf{bad}) = \emptyset\} & \text{By (4)} \\
= \{c \in \gamma(P') \mid c{\downarrow} \cap \mathsf{min}(\mathit{pre}^{*}(\mathsf{bad})) = \emptyset\} & \mathsf{min}(A) \subseteq A \text{ if } A \in \mathit{UCS}(X) \\
= \{c \in \gamma(P') \mid \{c\} \cap \mathit{pre}^{*}(\mathsf{bad}) = \emptyset\} & \text{Lem. 3.1} \\
= \{c \in \gamma(P') \mid c \notin \mathit{pre}^{*}(\mathsf{bad})\} &
\end{array}$$

Hence, $\gamma(\mathit{post}^{\#}(P)) \cap \mathit{pre}^{*}(\mathsf{bad}) = \emptyset$. □

Using the previous lemma and induction we can establish the following theorem.

Theorem 2. *Given a* WSTS $(X, \delta, \succeq)$ *and a set* $\mathsf{bad} \in \mathit{UCS}(X)$ *fix a finite domain* D *and a set* $P' \in \mathit{DPL}(D)$ *such that* $\mathit{post}^{\#}(P') \sqsubseteq P'$ *and* $\mathsf{min}(\mathit{pre}^{*}(\mathsf{bad})) \cap \gamma(P') \subseteq D$*. For every* $I \subseteq X$ *such that* $\alpha(I) \sqsubseteq P'$*, we have* $I \cap \mathit{pre}^{*}(\mathsf{bad}) = \emptyset \Leftrightarrow (\gamma \circ (\mathit{post}^{\#})^{*} \circ \alpha)(I) \cap \mathsf{bad} = \emptyset$*.*

We are nearly in position to define our refinement-based algorithm. We first define the following operator parametrized by $\mathcal{O} \subseteq X$ which is applied to a finite subset of states $T \subseteq X$: $\mathit{minpre}[\mathcal{S}, \mathcal{O}](T) \stackrel{\text{def}}{=} \mathit{minpre}[\mathcal{S}](T) \cap \mathcal{O}$.We also write $\mathit{minpre}[\mathcal{O}](T)$ instead of $\mathit{minpre}[\mathcal{S}, \mathcal{O}](T)$ if the WSTS is clear from the context.

In the remainder of this section we adopt the following convention: a set A acting as the argument of *minpre* should be read as $\mathsf{min}(A)$. A direct consequence of the definition of *minpre* is the following, for any $\mathcal{O} \subseteq \mathcal{O}' \subseteq X$ and $A \subseteq X$ we have:

$$\mathit{minpre}[\mathcal{O}]^{*}(A) \subseteq \mathit{minpre}[\mathcal{O}']^{*}(A) \ . \tag{5}$$

The main ideas underlying our refinement-based algorithm (Algorithm 2) are as follows. In a first approximation, we consider a finite domain D_0 that contains a minor set of bad. With this set, we compute a first overapproximation of the reachable states of $\mathcal{S}_0$, noted $\mathcal{O}_0$. If this overapproximation is fine enough to prove that we are in presence of a negative instance of the problem then we conclude at line 2. If it is not the case, we compute R'_0 that represents all the states within $\mathcal{O}_0$ that can reach bad in one step. If this set contains x_0 then we conclude that bad is reachable. Otherwise, we refine the finite domain D_0 into D_1 to ensure at the next iteration that our overapproximation will be more precise (Prop. 5.2) and that $(\gamma[D_1] \circ \mathit{post}^{\#}[\mathcal{S}, D_1] \circ \alpha[D_1])(x_0))$ will not intersect with bad. So, we have excluded **all** spurious counter-examples of length one. We then proceed with this enlarged finite domain.

Since $\mathsf{min}(pre^*(\mathsf{bad}))$ is computable, Theorem 2 intuitively shows that our algorithm terminates. We formally establish the correctness of our technique as stated in the next lemmas which prove soundness, completeness, and termination of Algorithm 2.

Algorithm 2. Refinement loop

```
  Input: An IWSTS $\mathcal{S}_0$ and a set $\mathsf{bad} \in UCS(X)$
  Let $D_0 \supseteq (\mathsf{min}(\mathsf{bad}))$
  for $i = 0, 1, 2, \ldots$ do
1     Compute $R_i$ defined to be $((post^{\#}[\mathcal{S}, D_i])^* \circ \alpha[D_i])(x_0)$
      Let $\mathcal{O}_i$ denote $\gamma[D_i](R_i)$
2     if $\mathcal{O}_i \cap \mathsf{bad} = \emptyset$ then return UNREACHABLE
      else
3         Compute $R'_i$ defined to be $\mathsf{min}\left(\bigcup_{k=0}^{i+1} minpre[\mathcal{S}, \mathcal{O}_i]^k(\mathsf{bad})\right)$
4         if $\{x_0\} \cap R'_i{\uparrow} = \emptyset$ then
5             choose $D_{i+1} \supseteq D_i \cup R'_i$
          else return REACHABLE
      end
  end
```

Lemma 10 (Soundness). *If Algorithm 2 says "*REACHABLE*" then we have* $post^*(x_0) \cap \mathsf{bad} \neq \emptyset$.

Proof. Let c be the value of variable i when the algorithm says "REACHABLE". $minpre[\mathcal{O}_c]^*(\mathsf{bad}){\uparrow} \subseteq minpre[X]^*(\mathsf{bad}){\uparrow} = pre^*(\mathsf{bad})$, the inclusion follows from $\mathcal{O}_c \subseteq X$, (5) and $\uparrow$ is monotonic, and the equality follows from Lemma 4.b. Since $\{x_0\} \cap pre^*(\mathsf{bad}) \neq \emptyset$ iff $post^*(x_0) \cap \mathsf{bad} \neq \emptyset$, $minpre[\mathcal{O}_c]^c(\mathsf{bad}){\uparrow} \subseteq pre^*(\mathsf{bad})$ shows that $post^*(x_0) \cap \mathsf{bad} \neq \emptyset$, by $\{x_0\} \cap minpre[\mathcal{O}_c]^c(\mathsf{bad}){\uparrow} \neq \emptyset$ (line 4). □

Lemma 11 (Completeness). *If Algorithm 2 says "*UNREACHABLE*" then we have* $post^*(x_0) \cap \mathsf{bad} = \emptyset$.

Proof. Fix a finite domain D, by Lemma 7 we have that $post^*(x_0) \subseteq (\gamma \circ (post^{\#})^* \circ \alpha)(x_0)$. Let c be the value of variable i when the algorithm says "UNREACHABLE" at line 2. We conclude from $(\gamma[D_c] \circ (post^{\#}[\mathcal{S}, D_c])^* \circ \alpha[D_c])(x_0) \cap \mathsf{bad} = \emptyset$ that $post^*(x_0) \cap \mathsf{bad} = \emptyset$ which is the desired conclusion. □

Lemma 12 (Termination). *Given an effective* IWSTS $\mathcal{S}_0$ *and* $\mathsf{bad} \in UCS(X)$, *Algorithm 2 always terminates.*

Proof. It is routine to check that each domain D_i is finite. Hence, since $\sqcup_{D_i}$ is computable because the D_i's are finite and $post^{\#}[\mathcal{S}, D_i]$ is computable following Proposition 6 (notice that $\alpha[D_i](x_0)$ is computable since $\succeq$ is assumed to be decidable), the fixpoint computation of line 1 finishes after a finite amount of time.

Suppose, contrary to our claim, that the algorithm does not terminate. Since each line is evaluated in a finite amount of time, it follows that the algorithm executes the main loop infinitely many times. From line 5, we conclude that the algorithm considers an infinite sequence of finite domains $D_0 \subseteq D_1 \subseteq \cdots$ From Proposition 5.2, we know that $\mathcal{O}_0 \supseteq \mathcal{O}_1 \supseteq \cdots$ From Lemma 2, we conclude that there exists $i \geq 0$ such that $\mathcal{O}_i = \mathcal{O}_{i+1} = \cdots$

Let us consider the iteration i of the algorithm such that $\mathcal{O}_i = \mathcal{O}_{i+1} = \cdots$ We have the infinite sequence $R'_i\!\uparrow \subseteq R'_{i+1}\!\uparrow \subseteq \cdots$. From Lemma 2, we conclude that there exists $j \geq i$ such that $R'_j\!\uparrow = R'_{j+1}\!\uparrow = \cdots$. Hence, following line 5 of the algorithm, D contains $\mathsf{min}(\mathit{minpre}[\mathcal{O}_i]^*(\mathsf{bad}))$ (or rather D contains equivalent states to those of $\mathsf{min}(\mathit{minpre}[\mathcal{O}_i]^*(\mathsf{bad}))$) after the j^{th} iteration.

Let us now prove that **(a)** $\mathsf{min}(\mathit{minpre}[\mathcal{O}_{j+1}]^*(\mathsf{bad})) \equiv \mathsf{min}(\mathit{minpre}[X]^*(\mathsf{bad})) \cap \mathcal{O}_{j+1}$. Indeed, if it is not the case there exist $l \geq 0, c, c' \in X$ such that $c \in \mathit{minpre}[X]^l(\mathsf{bad})$, $c' \in \mathit{minpre}[X](c)$, $c \notin \mathcal{O}_{j+1}$ and $c' \in \mathcal{O}_{j+1}$. Hence, $\mathit{post}(c') \not\subseteq \mathcal{O}_{j+1}$ since $\mathit{post}(c') \cap c\!\uparrow \neq \emptyset$ and $\mathcal{O}_{j+1}$ is a $\succeq$-dc-set. But, $\forall \overline{c} \in \mathcal{O}_{j+1} \colon \mathit{post}(\overline{c}) \subseteq \mathcal{O}_{j+1}$. From this follows a contradiction.

Moreover, **(b)** $\mathsf{min}((\mathit{minpre}[X]^*(\mathsf{bad}))\!\uparrow) \equiv \mathsf{min}(\mathit{pre}^*(\mathsf{bad}))$ holds by Lemma 4.b and by definition of $\equiv$. We conclude, following line 5 of the algorithm, that D_{j+1} contains equivalent states to those of $\mathsf{min}(\mathit{pre}^*(\mathsf{bad})) \cap \mathcal{O}_{j+1}$.

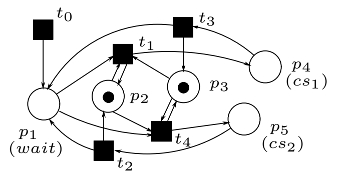
By applying Theorem 2, we have $\{x_0\} \cap \mathit{pre}^*(\mathsf{bad}) = \emptyset$ iff $\mathcal{O}_{j+1} \cap \mathsf{bad} = \emptyset$. We consider two cases: (i) $\{x_0\} \cap \mathit{pre}^*(\mathsf{bad}) = \emptyset$, then we have $\mathcal{O}_{j+1} \cap \mathsf{bad} = \emptyset$ and the algorithm terminates since the test of line 2 is evaluated to true; (ii) $\{x_0\} \cap \mathit{pre}^*(\mathsf{bad}) \neq \emptyset$, then $\mathcal{O}_{j+1} \cap \mathsf{bad} \neq \emptyset$. Following **(a)** and **(b)** at line 3 of the algorithm $R'_{j+1} \equiv \mathsf{min}(\mathit{pre}^*(\mathsf{bad})) \cap \mathcal{O}_{j+1}$. Since $\{x_0\} \cap \mathit{pre}^*(\mathsf{bad}) \neq \emptyset$, there exists, on account of Lemma 3.3, $x \in \mathsf{min}(\mathit{pre}^*(\mathsf{bad}))\colon x_0 \succeq x$. $x_0 \in \mathcal{O}_{j+1}$ and $\mathcal{O}_{j+1} \in \mathit{DCS}(X)$ shows that $x \in \mathcal{O}_{j+1}$. We conclude from $R'_{j+1} \equiv \mathsf{min}(\mathit{pre}^*(\mathsf{bad})) \cap \mathcal{O}_{j+1}$ that $[x] \cap R'_{j+1} \neq \emptyset$, hence that $\{x_0\} \cap R'_{j+1}\!\uparrow \neq \emptyset$, and finally that the test of line 4 is evaluated to false which yields the algorithm to terminate. □

Remark 2. Let us notice that the practical efficiency of Algorithm 2 depends on (i) the preciseness of the overapproximations $\mathcal{O}_i$ and (ii) the time (and space) needed to build those overapproximations. Point (i) is crucial since rough approximations will lead to the computation of $\mathsf{min}(\mathit{pre}^*(\mathsf{bad}))$, which is time and space consuming in practice [23]. Point (ii) is important because an inefficient computation of overapproximations leads to an inefficient algorithm. Hence, a trade-off between (i) and (ii) must be chosen. This problem exceeds the scope of this paper and will be addressed in future works.

To ensure termination we require, at line 5, that the finite domain is enlarged by, at least, the states of R'_i. The algorithm remains correct if we add more states.

6 Illustrations

We have produced a prototype that implements Algorithm 2. We describe in this section the execution of that prototype when applied on a toy example. The example of IWSTS $\mathcal{S}_0$ is represented through a Petri net (see [8] for details), depicted in Fig. 2, which models a very simple mutual exclusion protocol. We want to check for safety of the protocol, that is check that *there is never more than one process in the critical sections.* The markings that violates the property, denoted bad, are given by $\{\langle 0,0,0,1,1\rangle, \langle 0,0,0,0,2\rangle, \langle 0,0,0,2,0\rangle\}\!\uparrow$. It is worth



The processes (the tokens in place p_1) can access some critical section (place p_4 or p_5) provided they acquired some lock (the tokens in places p_2 and p_3). The initial marking is given by $\langle 0,1,1,0,0\rangle$. Transition t_0 spawns processes.

Fig. 2. A simple mutual exclusion protocol

pointing that we want to establish the safety for any number of processes taking part in the protocol (recall that t_0 spawns processes).

Execution of the prototype. We describe the execution of the prototype iteration by iteration. On account of remark 2, we do not take $\mathsf{min}(\mathsf{bad})$ as initial finite domain but its downward closure instead and we do not add the set R'_i to D_i at the i^{th} iteration but its downward closure instead. Taking the $\succeq$-downward closure of the sets allows us to efficiently prove the safeness of the protocol.

Initialisation. As mentioned before, the initial value of the finite domain, which is referred as D_0, is given by $\{\langle 0,0,0,1,1\rangle, \langle 0,0,0,0,2\rangle, \langle 0,0,0,2,0\rangle\}\!\downarrow$.

Iteration 1 (i=0). After the fixpoint computation of line 1, we have $R_0 = D_0$, and so $\mathcal{O}_0 = X$. Hence the test of line 2 fails and we compute $R'_0 = \mathsf{min}(\mathsf{bad}) \cup \{\langle 1,1,1,0,1\rangle, \langle 1,1,1,1,0\rangle\}$ which corresponds to $\mathsf{min}(\mathsf{bad} \cup \mathit{pre}(\mathsf{bad}))$. Because the test of line 4 fails, we execute line 5 and we set D_1 to $R'_0\!\downarrow$.

Iteration 2 (i=1). The fixpoint computation of line 1 ends up with $R_1 = D_1$, hence $\mathcal{O}_1 = X$. Again we perform a refinement step by (i) computing $R'_1 = \mathsf{min}(\mathsf{bad}) \cup \{\langle 0,1,1,0,1\rangle, \langle 0,1,1,1,0\rangle, \langle 2,2,1,0,0\rangle, \langle 2,1,2,0,0\rangle\}$ (which corresponds to $\mathsf{min}(\mathsf{bad} \cup \mathit{pre}(\mathsf{bad}) \cup \mathit{pre}^2(\mathsf{bad})))$ and (ii) adding tuples of $R'_1\!\downarrow$ with the ones of D_1 to obtain D_2.

Iteration 3 (i=2). The fixpoint computation of line 1 finishes with a set R_2 such that the test of line 2 ($\mathcal{O}_2 \cap \mathsf{bad} = \emptyset$) succeeds and the system is proved to be safe.

Indeed $\mathcal{O}_2 = \{(p_1,p_2,p_3,p_4,p_5) \in \mathbb{N}^5 \mid (\bigwedge_{i=2}^{5} p_i \leq 1) \wedge p_4 + p_5 \leq 1 \wedge p_3 + p_4 \leq 1 \wedge p_2 + p_5 \leq 1\}$ which is equal to $\mathit{Cover}(\mathcal{S}_0)$. Since $\mathit{Cover}(\mathcal{S}_0)$ is, in general, not computable ([10]), the equality does always not hold. Notice that $\mathit{pre}^*(\mathsf{bad})$ is computed in five iterations with the classical algorithm of [17]. Hence, the forward analysis allows to drastically cut the backward search. We hope this gain will appear also on many practical examples.

References

1. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science **126** (1994) 183–236
2. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings of LICS, IEEE Computer Society Press (1996) 278–292

3. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. Inf. Comput. **127** (1996) 91–101
4. Abdulla, P., Annichini, A., Bouajjani, A.: Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In: Proceedings of TACAS. Volume 1579 of LNCS., Springer (1999) 208–222
5. Delzanno, G., Raskin, J.F., Van Begin, L.: Towards the automated verification of multithreaded java programs. In: Proceedings of TACAS. Volume 2280 of LNCS., Springer (2002) 173–187
6. Bardin, S., Finkel, A., Leroux, J., Petrucci, L.: Fast: Fast acceleration of symbolic transition systems. In: Proceedings of CAV. Volume 2725 of LNCS., Springer (2003) 118–121
7. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: Proceedings of LICS, IEEE Computer Society Press (1999) 352–359
8. Reisig, W.: Petri Nets. An introduction. Springer (1986)
9. Ciardo, G.: Petri nets with marking-dependent arc multiplicity: properties and analysis. In: Proc. of ATPN. Volume 815 of LNCS., Springer (1994) 179–198
10. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: Proceedings of ICALP. Volume 1443 of LNCS., Springer (1998) 103–115
11. Raskin, J.F., Van Begin, L.: Petri nets with non-blocking arcs are difficult to analyse. In: Proceedings of INFINITY. Volume 96 of ENTCS., Elsevier (2003)
12. Emerson, E.A., Namjoshi, K.S.: On model checking for non-deterministic infinite-state systems. In: Proc. of LICS, IEEE Computer Society Press (1998) 70–80
13. Higman, G.: Ordering by divisibility in abstract algebras. Proc. London Math. Soc. (3) **2** (1952) 326–336
14. Geeraerts, G., Raskin, J.F., Van Begin, L.: Expand, Enlarge and Check: new algorithms for the coverability problem of WSTS. In: Proceedings of FSTTCS. Volume 3328 of LNCS., Springer (2004) 287–298
15. Abdulla, P., Deneux, J., Mahata, P., Nylen, A.: Forward reachability analysis of timed petri nets. In: Proceedings of Formats-FTRTFT. Volume 3253 of LNCS., Springer (2004) 343–362
16. Ganty, P., Raskin, J.F., Van Begin, L.: A complete abstract interpretation framework for coverability properties of WSTS. Technical Report 2005.57, Centre Fédéré en Vérification (CFV) (2005)
17. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.K.: General decidability theorems for infinite-state systems. In: Proceedings of LICS, IEEE Computer Society Press (1996) 313–321
18. Delzanno, G., Raskin, J.F., Begin, L.V.: Covering sharing trees: a compact data structure for parameterized verification. Software Tools for Technology Transfer (STTT) **5** (2004) 268–297
19. Finkel, A., Schnoebelen, Ph.: Well-structured transition systems everywhere! Theoretical Computer Science **256** (2001) 63–92
20. Finkel, A.: Reduction and covering of infinite reachability trees. Inf. Comput. **89** (1990) 144–179
21. Esparza, J., Ganty, P., Schwoon, S.: Locality-based abstractions. In: Proceedings of SAS. Volume 3672 of LNCS., Springer (2005) 118–134
22. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers **35** (1986) 677–691
23. Van Begin, L.: Efficient Verification of Counting Abstractions for Parametric Systems. PhD thesis, Université Libre de Bruxelles (2003)