Deadlock-free Discrete Controller Synthesis for Infinite State Systems

Nicolas Berthier, Hervé Marchand

Deadlock-free Discrete Controller Synthesis for Infinite State Systems

Nicolas Berthier and Hervé Marchand*

Abstract— **We elaborate on our former work for the safety control of infinite reactive synchronous systems modeled by arithmetic symbolic transition systems. By using abstract interpretation techniques involving disjunctive polyhedral over-approximations, we provide effective symbolic algorithms allowing to solve the deadlock-free safety control problem while overcoming previous limitations regarding the non-convexity of the set of states violating the invariant to enforce.**

I. Introduction

The control theory of discrete event systems (Ramadge and Wonham [1]; Cassandras and Lafortune [2]) allows the use of constructive methods ensuring, *a priori* and by means of control, required properties on a system's behavior.

When modeling realistic systems, it is useful to manipulate state/event variables in which the states (as well as the events) can be seen as instantiations of vectors of variables. In this case, usual control techniques entail instantiating the variables during state space exploration and analysis. This use of variable may lead to undecidability results whenever the domain of some variables is infinite.

We consider here reactive synchronous systems, *i.e.,* data-flow systems reacting to inputs sent by the environment, and producing outputs resulting from internal transformations. In this framework, part of the inputs is uncontrollable (it may correspond to measures from sensors), whereas the other part of inputs is (*e.g.,* user commands to actuators, that make the system evolve from a configuration to some other one). In this setting, we aim at synthesizing controllers restricting the possible values of the controllable inputs so as to ensure some properties.

A. Contributions

More precisely, we assume that the systems are modeled by Arithmetic Symbolic Transition Systems (ASTSs), that are transition systems with variables (Boolean, integer, real) encoding both the inputs of the system and its internal behavior. In [3], we provided algorithms allowing to compute a controller ensuring safety properties with variables in infinite domains using abstract interpretation techniques over-approximating the state space that has to be forbidden by control; we thereby performed the computations on an abstract (infinite) domain. In this paper, we extend this previous work by providing effective symbolic algorithms allowing to solve the deadlock-free safety control problem while overcoming previous limitations regarding the non-convexity of the set of states violating the invariant to enforce. To these ends, we use disjunctive polyhedral over-approximations.

This work was supported by the French ANR project Ctrl-Green (ANR-11-INFR 012 11), funded by ANR INFRA and MINALOGIC.

* INRIA Rennes - Bretagne Atlantique, Rennes, France

B. Related Works on the Control of Infinite Systems

The control of infinite systems has been subject to several studies. One can think about the control of Timed Automata [4], or Vector of Discrete Events Systems [5]. Kumar and Garg [6] extend their previous work [7] to consider infinite systems. They prove that, in that case, the state avoidance control problem is undecidable. They also show that the problem can be solved in the case of Petri Nets, when the set of forbidden states is upward-closed. The controller synthesis of *infinite state systems* modeled by Petri Nets has also been considered by Holloway et al. [8]. Regarding models more closely related to our ASTSs, one can find the work of Le Gall et al. [9] that control symbolic transition systems with variables as well as Kalyon et al. [10], in an asynchronous framework and with finite alphabets.

The control of finite synchronous programs handling only Boolean variables has been subject to several studies [11; 12] and there exists a tool SIGALI implementing this theory.

We review further works related to our abstract interpretation setting in the Conclusion.

C. Outline

The rest of the paper is organized as follows. In Section II, we first fix notations, recall the model of ASTSs we developed in [3], define both the basic and deadlock-free safety control problems, and give the solutions for the basic problem. We then detail our contributions for the deadlock-free safety control problem for logico-numerical ASTSs in Section III. We quickly describe our implementation and conclude in Sections IV and V.

II. Control of Symbolic Transition Systems

We recall (and slightly reformulate) in this section the model of Arithmetic Symbolic Transition Systems, the associated invariance control problem, as well as the algorithmic solution developed in [3].

A. Arithmetic Symbolic Transition Systems

The model of Arithmetic Symbolic Transition Systems (ASTS) is a transition system with (internal or input) variables whose domain can be infinite, and composed of a finite set of symbolic transitions. Each transition is guarded on the system variables, and has an update function indicating the variable changes when a transition is fired. This model allows the representation of infinite systems whenever the variables take their values in an infinite domain, while it has a finite structure and offers a compact way to specify systems handling data.

1) Notations: Let $V = \langle v_1 \colon \mathcal{D}_{v_1}, \ldots, v_n \colon \mathcal{D}_{v_n} \rangle$ be a tuple of variables v defined on (infinite) domains $\mathcal{D}_v$. We note $\mathcal{D}_V = \prod_{i \in [1,n]} \mathcal{D}_{v_i}$ the (infinite) domain of V.

Given two vectors V_b and V_x of variables respectively defined over finite and infinite domains, $\mathcal{P}_{V_b \cup V_x}$ denotes the set of all *predicates* (also referred to as *characteristic functions*) defined over variables in V_b and a set of *interpreted variables* that proxy arithmetic conditions expressed on variables of V_x.

Remark 1*: Predicates involving a finite set of interpreted variables can be used to describe infinite spaces. For instance, given the set of variables $V = \langle b \colon \mathbb{B}, x \colon \mathbb{Z} \rangle$ (where $\mathbb{B}$ and $\mathbb{Z}$ respectively stand for the Booleans and Integers domains), the predicate $(b \wedge v_{x \leqslant 42}) \in \mathcal{P}_V$ using the interpreted variable $v_{x \leqslant 42}$ can be used to represent the infinite set $\{(b, x) \in \mathbb{B} \times \mathbb{Z} \mid b \wedge x \leqslant 42\}$.*

In the remainder of this paper, we develop symbolic algorithms exploiting the usual equivalence between a predicate defined over a vector of variables $P \in \mathcal{P}_V$ and its associated set of solutions $\{v \in \mathcal{D}_V \mid P(v)\}$.

By abuse of notation, depending on the context, we will often use P to denote the predicate and its associated set of solutions, and shall use logic notations to denote operations on such sets. Similarly in example predicates, we shall use the notation "$x \leqslant 42$" in place of the interpreted variable "$v_{x \leqslant 42}$".

2) The Model of ASTSs:

Definition 1 (Arithmetic Symbolic Transition System)*: An* Arithmetic Symbolic Transition System *is a tuple $S = \langle X, I, T, A, \Theta_0 \rangle$ where:*

- *$X = \langle x_1 \colon \mathcal{D}_{x_1}, \ldots, x_n \colon \mathcal{D}_{x_n} \rangle$ is a vector of state variables encoding the memory necessary for describing the system's behavior;*
- *$I = \langle i_1 \colon \mathcal{D}_{i_1}, \ldots, i_m \colon \mathcal{D}_{i_m} \rangle$ is a vector of input variables;*
- *T is of the form $(x_i' := T^{x_i})_{x_i \in X}$, such that, for each $x_i \in X$, the right-hand side T^{x_i} of the assignment $x_i' := T^{x_i}$ is an expression on $X \cup I$. T is called the transition function of S, and encodes the evolution of the state variable x_i. It characterizes the dynamic of the system between the current state and the next state when receiving an input vector.*
- *$A \in \mathcal{P}_{X \cup I}$ is a predicate encoding an assertion on the possible values of the inputs depending on the current state;*
- *$\Theta_0 \in \mathcal{P}_X$ is a predicate encoding the set of initial states.*

For technical reasons, we shall assume that A and expressions of T are expressed in a theory that is closed under quantifier elimination as for example the Presburger arithmetic.

ASTSs can conveniently be represented as parallel compositions of Mealy automata with numerical variables and explicit locations or in its symbolic form.

Example 1*:* Consider the following example ASTS where

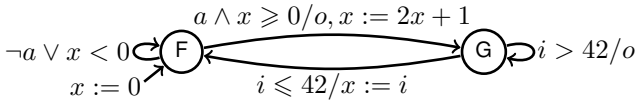

Fig. 1. Example Mealy automaton with an Integer variable.

$X = \langle \xi \colon \{\mathsf{F}, \mathsf{G}\}, x \colon \mathbb{Z}, o \colon \mathbb{B} \rangle$, $I = \langle a \colon \mathbb{B}, i \colon \mathbb{Z} \rangle$,

$$T = \begin{cases} \xi' := & \mathsf{G} & \text{if } (\xi = \mathsf{F} \wedge a \wedge x \geqslant 0), \\ & \mathsf{F} & \text{if } (\xi = \mathsf{G} \wedge i \leqslant 42), \xi \text{ otherwise} \\ x' := & 2x + 1 & \text{if } (\xi = \mathsf{F} \wedge a \wedge x \geqslant 0), \\ & i & \text{if } (\xi = \mathsf{G} \wedge i \leqslant 42), x \text{ otherwise} \\ o' := & (\xi = \mathsf{F} \wedge a \wedge x \geqslant 0) \vee (\xi = \mathsf{G} \wedge i > 42) \end{cases}$$

$A(\langle \xi, x, o, a, i \rangle) = (\xi = \mathsf{G} \Rightarrow 3x + 2i \leqslant 41)$,
$\Theta_0(\langle \xi, x, o \rangle) = (\xi = \mathsf{F} \wedge x = 0)$.
(The state variable o is left uninitialized for illustrative purposes: it is actually an *output* of the system.) The corresponding Mealy automaton with explicit locations (leaving A aside) can be represented as in Figure 1.

Remark 2*: We qualify as* logico-numerical *an ASTS whose state and input variables are Boolean variables ($\mathbb{B}$) or numerical variables (typically, $\mathbb{R}$ or $\mathbb{Z}$), i.e., such that $X = \mathbb{B}^k \cup \mathbb{R}^{k'} \cup \mathbb{Z}^{k''}$ with $k + k' + k'' = n$ (and similarly for the input variables). ASTSs with only Boolean state variables are called* finite*.*

3) Invariant for an ASTS: The characteristic function of the *reachable* state space of an ASTS $S = \langle X, I, T, A, \Theta_0 \rangle$ (also known as its *orbit*) can be defined as the least fixpoint (lfp) $\mathcal{O}_S \in \mathcal{P}_X$:

$$\mathcal{O}_S \stackrel{\text{def}}{=} \text{lfp}\,(\lambda \mathcal{X}. \Theta_0 \vee \text{post}(\mathcal{X}))\,,$$

where $\text{post} \colon \mathcal{P}_X \to \mathcal{P}_X$ computes the set of successors of a given set of states. post itself can be defined as:

$$\text{post}(\mathcal{X}) \stackrel{\text{def}}{=} \mathsf{assign}_T(\mathcal{X} \wedge A)$$

where $\mathsf{assign}_T \colon \mathcal{P}_{X \cup I} \to \mathcal{P}_X$ takes a set of transitions (a predicate on $X \cup I$), and computes all successor states according to the assignments in T.

Remark 3*: By construction of ASTSs' transition functions, $\mathsf{assign}_T(t) = \mathit{false}$ (an empty set of states) iff $t = \mathit{false}$ (an empty set of transitions).*

Definition 2 (Invariant for an ASTS)*: Given an ASTS $S = \langle X, I, T, A, \Theta_0 \rangle$ and a predicate $\Phi \in \mathcal{P}_X$, we say that Φ is an* invariant *of S (or S satisfies Φ — noted $S \models \Phi$) whenever its orbit $\mathcal{O}_S$ has an empty intersection with the complement of the state space induced by Φ, i.e., $\neg\Phi \wedge \mathcal{O}_S = \mathit{false}$ (or equivalently, $\mathcal{O}_S \subseteq \Phi$). In other words, every state reachable by S satisfies Φ.*

4) Deadlock-free ASTS: Further, one may want to ensure that an ASTS be "reactive", meaning that there does not exist a deadlocking state in its orbit. As a consequence of remark 3, deadlock states are necessarily the result of an unsatisfiable assertion:

Definition 3 (Deadlock-free ASTS)*: An ASTS $S = \langle X, I, T, A, \Theta_0 \rangle$ is* deadlock-free *iff $\forall x \in \mathcal{O}_S, \exists i \in \mathcal{D}_I, A(x, i)$.*

B. Safety Control Problems for ASTSs

Assume given a system S and a predicate Φ on S. Our aim is to restrict the behavior of S by means of control in order to fulfill Φ. We distinguish between the uncontrollable input variables U which are defined by the environment, and the controllable input variables C which are defined/restricted by the controller of the system. For a technical reason explained in Remark 4 below, we assume that the controllable variables are Booleans. Note that the partitioning of the input variables in S induces a "partitioning" of the input alphabet of S. A controller is then given by a predicate $K_\Phi \in \mathcal{P}_{X\cup U\cup C}$ that constrains the set of admissible (Boolean) controllable inputs so that Φ is an invariant of the controlled system.

Definition 4 (Basic Safety Control Problem)*: Given an ASTS* $S = \langle X, U \uplus C, T, A, \Theta_0\rangle$ *and a predicate* $\Phi \in \mathcal{P}_X$*, solving the* basic safety control problem *is to compute a predicate* $K_\Phi \in \mathcal{P}_X$ *such that:*

- $S' = \langle X, U \uplus C, T, K_\Phi, \Theta_0\rangle \models \Phi$*; and*
- $K_\Phi \Rightarrow A$.

The second point in the definition above states that K_Φ is at least as restrictive as A, hence does not allow behaviors that are forbidden in the original system; *i.e.,* in terms of sets of transitions, $K_\Phi \subseteq A$. The new assertion K_Φ is called a *controller* for the desired safety objective.

A solution to the basic safety control problem does not ensure that the resulting system is deadlock-free. We then refine this problem to the following new one:

Definition 5 (Deadlock-free Safety Control Problem)*: Given an ASTS* S *and a predicate* Φ*, solving the* deadlock-free safety control problem *is to compute a controller* K_Φ *such that:*

- K_Φ *solves the basic safety control problem;*
- $S' = \langle X, U \uplus C, T, K_\Phi, \Theta_0\rangle$ *is deadlock-free.*

Let us now summarize the algorithmic principles of the solution we proposed for the basic safety control problem for both finite and logico-numerical ASTSs in [3], as our main contribution consists in an extension of this previous work for the deadlock-free safety control problem, associated with new structures and techniques to compute the controller.

C. Solution for Finite ASTSs

Within a finite setting in which $X = \mathcal{B}^n$ and $I = \mathcal{B}^m$, the solution of the basic safety control problem is well known and relies on some fixpoint computation of predicates [11]. When dealing with numerical variables, the principle of the algorithm is actually the same (with the difference that the fixpoint computation may never terminate). We thus just recall in the sequel the principle of the controller computation and emphasize the part(s) that may not be actually computed when dealing with logico-numerical ASTSs.

Consider given an ASTS $S = \langle X, U \uplus C, T, A, \Theta_0\rangle$ and an objective invariant $\Phi \in \mathcal{P}_X$.

We first define the (characteristic function of) the set of *forbidden* states that do not satisfy Φ as $\mathfrak{F} \stackrel{\text{def}}{=} \neg\Phi$. Obtaining a controller ensuring the unreachability of $\mathfrak{F}$ essentially boils down to compute the set of *ultimately forbidden* states $I_\mathfrak{F}$, that can uncontrollably lead, in any number of transitions, into a state in $\mathfrak{F}$. We illustrate this principle in Figure 2.

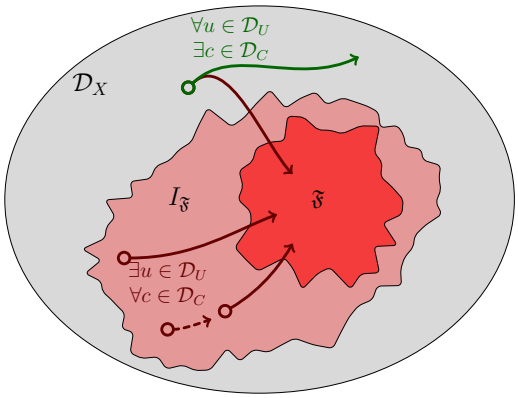

Fig. 2. Exact computation of the forbidden states $I_\mathfrak{F}$.

The definition of $\text{pre}_u \colon \mathcal{P}_X \to \mathcal{P}_X$, computing the set of states that can uncontrollably lead in one transition to a state belonging to a given set of states, is as follows:

$$\text{pre}_u(\mathcal{B}) \stackrel{\text{def}}{=} \exists_U \forall_C \left(T^{-1}(\mathcal{B}) \wedge A\right)$$

where $T^{-1} \colon \mathcal{P}_X \to \mathcal{P}_{X\cup U\cup C}$ denotes the "open" pre-image function (also giving inputs) of the ASTS S.

The set of forbidden states can then be computed as $I_\mathfrak{F} = \text{coreach}_u(\mathfrak{F})$, with $\text{coreach}_u \colon \mathcal{P}_X \to \mathcal{P}_X$ defined as the least fixpoint:

$$\text{coreach}_u(B) \stackrel{\text{def}}{=} \text{lfp}(\lambda\mathcal{B}.B \vee \text{pre}_u(\mathcal{B}))$$

The limit of the fixpoint $\text{coreach}_u(\mathfrak{F})$ actually exists as the function coreach_u is monotonic, but may not be computable when dealing with numerical variables.

If $\Theta_0 \wedge I_\mathfrak{F} \neq \varnothing$, then the synthesis fails [11]; *i.e.,* the set of leavers given by C does not provide sufficient means to ensure the unreachability of $\mathfrak{F}$. Otherwise, assuming that $I_\mathfrak{F}$ can actually be computed, one can obtain from it a new predicate $K_\Phi \in \mathcal{P}_{X\cup U\cup C}$, expressing the constraints over the state and input variables that must be satisfied for $\mathfrak{F}$ to be unreachable. K_Φ can be obtained as $K_\Phi = T^{-1}(\neg I_\mathfrak{F}) \wedge A$, and solves the safety control problem by intersecting A with all transitions *not* leading to $I_\mathfrak{F}$.

D. Over-approximating Solution for Logico-numerical ASTSs

As seen in the previous section, the actual computation of the controller, which is based on a fixpoint equation to compute $I_\mathfrak{F}$, is generally not possible for undecidability (or complexity) reasons. We use abstract interpretation techniques (see *e.g.,* [13]; [14]; [15]) to overcome the undecidability problem, and compute an over-approximation $I'_\mathfrak{F}$ of the fixpoint $I_\mathfrak{F}$. This over-approximation ensures that the forbidden states $\mathfrak{F}$ are not reachable in the controlled system. Thus, the synthesized controller remains correct, yet may not be maximally permissive *w.r.t.* the invariant. We illustrate this principle in Figure 3.

We shall now briefly present the abstract interpretation techniques and show how one can apply them to effectively

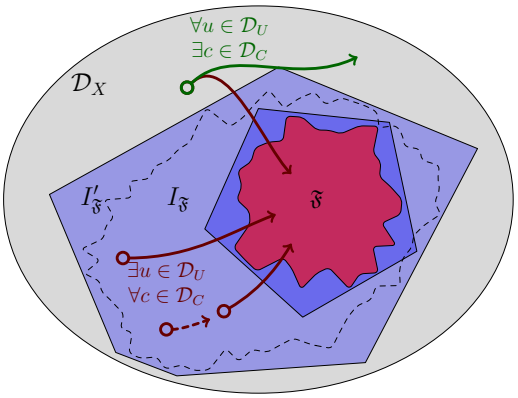

Fig. 3. Over-approximation $I'_{\mathfrak{F}}$ of the possibly non-computable set of forbidden states $I_{\mathfrak{F}}$.

solve the basic controller synthesis problem for logico-numerical ASTSs.

1) Overview of Abstract Interpretation Techniques: Abstract interpretation techniques rely on Galois connections to approximate the solution of fixpoints of the form $\text{fp}(\lambda x.f(x))$, for a monotonic function f.

Let $\langle D, \subseteq \rangle$ and $\langle \Lambda, \sqsubseteq \rangle$ be partial orders, and let $\alpha\colon D \to \Lambda$ and $\gamma\colon \Lambda \to D$ be functions. Then we say that $D \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Lambda$ is a *Galois connection* iff $\forall d \in D, \forall \ell \in \Lambda, d \subseteq \gamma(\ell) \Leftrightarrow \alpha(d) \sqsubseteq \ell$.

In abstract interpretation settings, D is the (possibly infinite) *concrete domain*, and exactly represents sets of states; Λ is the (possibly infinite) *abstract domain*, over-approximating these sets with a finite representation. $\sqcup$ (resp. $\sqcap$) denotes the *join* (resp. *meet*) operation, over-approximating in the abstract domain the union (resp. intersection) of sets. Note that $\langle \Lambda, \sqsubseteq \rangle$ associated with the above operations forms a complete lattice of which we denote by $\top$ the top and $\bot$ the bottom elements.

As the goal is to compute an over-approximation of (co-)reachable state space(s), we denote by $f^\sharp\colon \Lambda \to \Lambda$ the function "evaluating" a program statement $f\colon D \to D$ on abstract values (*i.e.,* such that $\forall \ell \in \Lambda, f^\sharp(\ell) \sqsupseteq \alpha \circ f \circ \gamma(\ell)$).

Also, given an abstract value $\ell \in \Lambda$ and a function $f^\sharp\colon \Lambda \to \Lambda$ such that $\forall \ell \in \Lambda,\ \ell \sqsubseteq f^\sharp(\ell)$, the *widening operator* $\nabla\colon \Lambda \times \Lambda \to \Lambda$ is such that $\text{lfp}(\lambda l.\ell \sqcup f^\sharp(l)) \sqsubseteq \text{lfp}(\lambda l.\ell \nabla f^\sharp(l))$.This operator guarantees the convergence of the fixpoint computation in a finite number of steps.

Typical numerical abstract domains represent state spaces defined on numerical variables only (*i.e.,* $\mathbb{R}^n$); these domains are usually represented as *convex* over-approximations like boxes [16] or convex polyhedra [17].

We already gave details in [3] about composed logico-numerical abstract domains, that allow to represent and manipulate over-approximations of logico-numerical spaces. We pointed out then that the *power domain* mapping finite components of spaces onto convex numerical domains was appropriate for the purpose of synthesis.

So, in the remainder of this article and unless specified otherwise, we shall assume given a power abstract domain $\Lambda = \mathbb{B}^n \to \mathcal{N}$ based on a convex numerical domain $\mathcal{N}$, along with associated functions α and γ such that $\mathcal{P}_{X \cup U \cup C} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Lambda$.

Elements of Λ shall be represented as total functions $c_1 \mapsto f_1, \ldots, c_n \mapsto f_n$ from satisfiable conditions on Boolean variables to conjunctions of arithmetic constraints (possibly shortened as a single formula when its domain is a singleton).

2) Computing $I'_{\mathfrak{F}} \supseteq I_{\mathfrak{F}}$: We extend the definitions given in Section II-C to over-approximate the set of states that may uncontrollably lead to a state violating the desired invariant.

Given $T^{-1\sharp}$ the abstract version of the pre-image function T^{-1} for S, we define $\text{pre}^\sharp_u\colon \Lambda \to \Lambda$ as

$$\text{pre}^\sharp_u(\ell) \overset{\text{def}}{=} \exists^\sharp_U \forall^\sharp_C \left(T^{-1\sharp}(\ell) \sqcap^\sharp_\gamma A\right)$$

where $\exists^\sharp_U \ell$ (resp. $\forall^\sharp_C \ell$) denotes the existential (resp. universal) quantification of every variable belonging to U (resp. C), in the abstract value $\ell \in \Lambda$, and the $\sqcap^\sharp_\gamma\colon \Lambda \times \mathcal{P}_{X \cup I} \to \Lambda$ operator over-approximates the intersection between an abstract value and a concrete set of transitions specified by its characteristic function.

We have then $I'_{\mathfrak{F}} = \gamma \circ \text{coreach}^\sharp_u \circ \alpha(\mathfrak{F})$, with $\text{coreach}^\sharp_u\colon \Lambda \to \Lambda$ defined as:

$$\text{coreach}^\sharp_u(l) \overset{\text{def}}{=} \text{lfp}\left(\lambda \ell.l \sqcup \text{pre}^\sharp_u(\ell)\right)$$

As $\text{coreach}^\sharp_u$ may not converge in finite time, we actually use the following over-approximating function to get $I'_{\mathfrak{F}}$:

$$\text{coreach}^\nabla_u(l) \overset{\text{def}}{=} \text{lfp}\left(\lambda \ell.l \nabla \left(\ell \sqcup \text{pre}^\sharp_u(\ell)\right)\right)$$

From $I'_{\mathfrak{F}}$, one can compute a controller K_Φ, using the same technique as in the Boolean case.

Remark 4: Note that to be tractable and have an actual meaning when dealing with over-approximations, the universal quantification $\forall^\sharp$ used in the definition of $\text{pre}^\sharp_u$ above requires the eliminated dimensions to be finite. A direct consequence of this remark is that all solutions developed here require the controllable variables to be finite (i.e., Booleans in our case).

E. Overcoming the non-Convexity of $\mathfrak{F}$: "Split" Algorithm

In [3], we pointed out that our original solution could led to abstractions of the set $\mathfrak{F}$ that are too coarse to permit an actual computation of a controller. Indeed, the computation of $I'_{\mathfrak{F}}$ strongly relies on a representation of the numerical constraints by means of convex polyhedra. However, sometimes the over-approximation $\alpha(\mathfrak{F})$ of $\mathfrak{F}$ is too coarse to allow for an effective synthesis (*e.g.,* $\alpha\left(x \leqslant 0 \vee 10 \leqslant x\right) = \top$ with usual numerical abstract domains).

To alleviate this problem, and to compute a more precise over-approximation of $\mathfrak{F}$, we proposed to "split" $\mathfrak{F}$ into a disjunction of n convex clauses $\{\mathfrak{F}_i\}_{i \in [1,n]} = \text{convex}(\mathfrak{F})$, compute every set $I'_{\mathfrak{F}_i}$ independently as explained above, and then obtain the controller avoiding the states $\bigvee_{i \in [1,n]} \mathfrak{F}_i$ from $\bigvee_{i \in [1,n]} I'_{\mathfrak{F}_i}$.

Remark 5: A convex predicate $b \in \mathcal{P}_{V_b \cup V_x}$ is such that it can be rewritten as a conjunction $g \wedge f$ where $g \in \mathcal{P}_{V_b}$ is a predicate on the Boolean variables V_b and $f \in \mathcal{P}_{V_x}$ is a conjunction of interpreted variables (that proxy arithmetic

conditions). The function $\mathsf{convex}\colon \mathcal{P}_{X\cup U\cup C} \to \wp(\mathcal{P}_{X\cup U\cup C})$ *above splits a given predicate into a finite set of convex ones such that* $\forall \mathcal{B} \in \mathcal{P}_{X\cup U\cup C}, \mathcal{B} = \bigvee \mathsf{convex}(\mathcal{B})$.

As we turn to the deadlock-free safety control problem in the next Section, we shall see that this "split" algorithm leads to the appearance of deadlocking states. We will then present a new solution overcoming this limitation.

III. Contributions for Deadlock-free Control

We first handle the case where the original ASTS is already deadlock-free, and show that the exact and over-approximation algorithms do not introduce deadlocks by themselves. We then turn to the "split" algorithm and detail an example execution showing the appearance of deadlocking states. We then expose our solution for deadlock-free control in these cases by assuming that the original system does not induce deadlocking states, and relax this hypothesis at last.

A. Deadlock-free Control for Deadlock-free ASTSs

Let us first show that the exact computation in the finite case, as well as the over-approximating algorithm, do not introduce new deadlocks *per se*; *i.e.,* the controlled system has deadlocks iff the original system has deadlocks or the "split" algorithm is used.

In the remainder of this Section, we consider any deadlock-free ASTS $S = \langle X, U \uplus C, T, A, \Theta_0 \rangle$, and any set of states $\mathcal{X} \in \mathcal{P}_X$ and objective invariant $\Phi \in \mathcal{P}_X$.

Definition 6*: The* potential deadlocking states induced *by* $\mathcal{X}$*, noted* $\mathrm{pre}_d(\mathcal{X})$*, are the predecessor states of* $\mathcal{X}$ *that would become potential deadlocks if all states in* $\mathcal{X}$ *was forbidden. The potential deadlocking states induced by* $\mathcal{X}$ *can also be defined as* $\mathrm{pre}_d(\mathcal{X}) \stackrel{def}{=} \forall_U \forall_C \left(T^{-1}(\mathcal{X}) \wedge A\right)$.

Proposition 1*:* The potential deadlocking states induced by $\mathcal{X}$ are included in the set of uncontrollable predecessors of $\mathcal{X}$; *i.e.,* $\mathrm{pre}_d(\mathcal{X}) \subseteq \mathrm{pre}_u(\mathcal{X})$.

Proof. We can rewrite the above statement as $\forall_U \forall_C \left(T^{-1}(\mathcal{X}) \wedge A\right) \Rightarrow \exists_U \forall_C \left(T^{-1}(\mathcal{X}) \wedge A\right)$, that follows directly since $\forall P \in \mathcal{P}_{X\cup U\cup C}, (\forall_U \forall_C P \Rightarrow \exists_U \forall_C P)$. □

Proposition 2*:* The exact algorithm solves the deadlock-free safety control problem for S and Φ.

Proof. Let $\mathfrak{F} = \neg\Phi$. As $\mathrm{coreach}_u$ is monotonically increasing, then any potentially deadlocking state induced by $I_{\mathfrak{F}}$ belongs to $I_{\mathfrak{F}} = \mathrm{coreach}_u(\mathfrak{F})$; *i.e.,* $\mathrm{pre}_d(I_{\mathfrak{F}}) \subseteq I_{\mathfrak{F}}$.
Let $K_\Phi = T^{-1}(\neg I_{\mathfrak{F}}) \wedge A$ solving the basic safety control problem and S' constructed as in Definition 5. The result follows by observing that $\mathcal{O}_S$ has not deadlocking states (S is deadlock-free) and $\mathcal{O}_{S'} \subseteq \mathcal{O}_S$, and thus no states other than those also in $I_{\mathfrak{F}}$ are forbidden by K_Φ in $\mathcal{O}_{S'}$. □

Similar arguments can be used to conclude that the over-approximating algorithm does not lead to deadlocks for deadlock-free logico-numerical ASTSs; the developments in this case can be performed by reasoning on the concrete over-approximations $\gamma \circ \alpha(\mathcal{X})$ and $I'_{\mathfrak{F}}$.

B. Illustrating the Deadlock Appearance Problem

We now show that the "split" algorithm presented in Section II-E, although giving a controller solving the basic safety control problem, may lead to deadlocks in the resulting system. Let us illustrate this deadlock appearance problem by using an example:

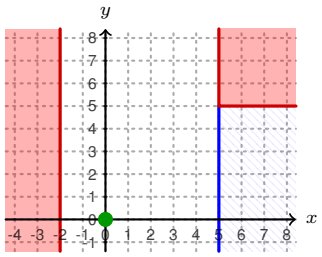

Fig. 4. Clipped Graphical Representation of $\mathfrak{F}$ (in plain) for the Example 2; the hashed zone depicts potentially deadlocking states that should be avoided to solve the deadlock-free control problem.

Example 2 *(S_{xy}):* Consider the following controllable ASTS S_{xy} where: $X = \langle x\colon \mathbb{R}, y\colon \mathbb{R}\rangle$, $U = \langle i\colon \mathbb{R}, j\colon \mathbb{R}\rangle$, $C = \langle c\colon \mathbb{B}\rangle$,

$$T = \begin{cases} x' := & x + i & \text{if } c, x \text{ otherwise} \\ y' := & y + j & \text{if } \neg c, y \text{ otherwise} \end{cases}$$

$A(\langle x, y, i, j, c\rangle) = \mathit{true}$, and $\Theta_0(\langle x, y\rangle) = (x = 0 \wedge y = 0)$. At each step, the only lever given to the controller by this ASTS is the choice of the variable to update (either x or y).

Consider now the invariant objective for this ASTS: $\Phi(\langle x, y\rangle) = ((x < 5 \vee y < 5) \wedge (x > -2))$. When expressed in disjunctive normal form, $\mathfrak{F}$ ($= \neg\Phi$) can be stated as the predicate $((x \geqslant 5 \wedge y \geqslant 5) \vee (x \leqslant -2))$; we illustrate in Figure 4 the corresponding zones of $\mathbb{R}^2$. Obviously, $\alpha(\mathfrak{F}) = \top$ due to the non-convexity of $\mathfrak{F}$. Hence we need to consider using the "split" algorithm to handle this example.

For each convex clause $\{\mathfrak{F}_i\}_{i\in\{1,2\}} = \mathsf{convex}(\mathfrak{F})$ (*e.g.,* $= \{x \geqslant 5 \wedge y \geqslant 5, x \leqslant -2\}$), we obtain $I'_{\mathfrak{F}_i} = \gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\mathfrak{F}_i) = \mathfrak{F}_i$ by using the numerical abstract domain of convex polyhedra. One can then compute $I'_{\mathfrak{F}} = I'_{\mathfrak{F}_1} \vee I'_{\mathfrak{F}_2}$; the initial state does not belong to this set of states, hence a correct controller can be obtained by computing:

$$K_\Phi = T^{-1}\left(\neg I'_{\mathfrak{F}}\right) \wedge A = (\neg c \wedge x > -2 \wedge x < 5) \vee (c \wedge x + i > -2 \wedge (y < 5 \vee x + i < 5)).$$

This controller constrains the value of c to prevent S_{xy} from entering exactly the states represented by the zones highlighted in red in Figure 4, and can be used to build the system $S'_{xy} \models \Phi$.

Assume now that from its initial state, S'_{xy} is given an input vector such that $i = 6 \wedge c$. Substituting the values for every known variable in K_Φ, we obtain the constraint *true*, meaning that this input does not violate the assertion imposed by the system (the controller K_Φ); thus, S'_{xy} can evolve within the state where $x = 6 \wedge y = 0$. Assume that while in this latter state, S'_{xy} receives an input vector such that $i = -8 \wedge j = 5$. In such a case, $K_\Phi = (\neg c \wedge 6 > -2 \wedge 6 < 5) \vee (c \wedge 6 - 8 > -2 \wedge (0 < 5 \vee 6 - 8 < 5))$ is unsatisfiable whatever the value of the controllable variable c, and we have reached a deadlocking state.

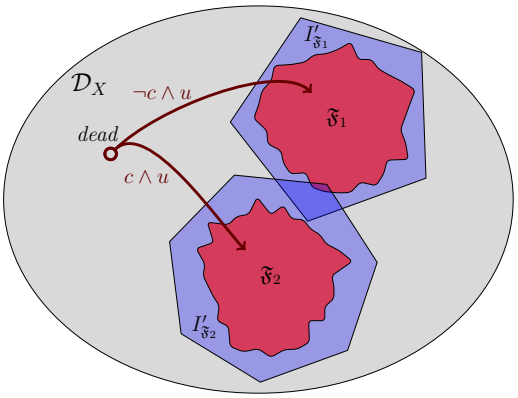

Fig. 5. Deadlocking state appearance when independently computing predecessors of convex over-approximations.

Manually examining S_{xy} and Φ, one can deduce that a controller solving the deadlock-free safety control problem should at least prevent the system from reaching all states where $x \geqslant 5$. Figure 4 also pictures the set of deadlocking states that should be avoided by control.

C. A Solution Relying on Disjunctive Over-approximation

We now turn to the deadlock-free safety control for deadlock-free logico-numerical ASTSs.

The example above illustrates that solving this problem requires to capture potentially deadlocking states that may have appeared if the "split" algorithm was used. Consider for instance a set of forbidden states $\mathfrak{F}$ such that $\{\mathfrak{F}_1, \mathfrak{F}_2\} = \mathsf{convex}(\mathfrak{F})$ as represented in Figure 5. If computed using $\bigvee_{i\in\{1,2\}} I'_{\mathfrak{F}_i}$, with $I'_{\mathfrak{F}_i} = \gamma \circ \mathrm{coreach}^{\sharp}_u \circ \alpha(\mathfrak{F}_i)$, since the universal quantifications induced by the individual fixpoints operate on over-approximations of subsets of forbidden states, then the resulting set of states to avoid would not comprise $dead$. In other words, for each $I'_{\mathfrak{F}_i}$ taken individually, the state $dead$ is not detected as necessarily leading into forbidden states, yet the controller obtained from their disjunction leads to a deadlock in $dead$.

Unlike in the "split" algorithm, we now overcome this issue by allowing to compute the universal quantifications based on over-approximations of all forbidden states at once. To do so, we extend the solution detailed in Section II-D.2 to operate on disjunctive abstract domains, that can be built as refinements of other domains like Λ.

1) Disjunctive Refinement of Abstract Domains: The refinement of a base convex abstract domain towards the construction of disjunctive domains and associated operations has been investigated in the context of program analysis and model-checking; see for instance the works of Cousot and Cousot [18], Giacobazzi and Ranzato [19], Filé and Ranzato [20], and Bagnara et al. [21].

Definition 7 (Finite Powerset Extension)*: Given a base domain* $\langle \Lambda, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$*, its* finite powerset extension *is the join-semilattice* $\langle \wp(\Lambda), \Subset, \oplus, \emptyset \rangle$ *where:*

- $\wp(\Lambda)$ *denotes the powerset of* Λ*, restricted to finite subsets;*
- $\Subset$ *is the Hoare partial order; i.e., such that*

$$(\mathscr{L}_1 \Subset \mathscr{L}_2) \Leftrightarrow (\forall \ell_1 \in \mathscr{L}_1, \exists \ell_2 \in \mathscr{L}_2, \ell_1 \sqsubseteq \ell_2)\,;$$

- $\oplus$ *is the set union;*
- $\emptyset$ *is the least element, i.e., the empty set.*

Remark 6 (Non-redundancy & Maximality)*: Note that for the sake of implementation efficiency, we can restrict the elements of* $\wp(\Lambda)$ *to be the maximal ones; i.e., every* $\mathscr{L} \in \wp(\Lambda)$ *is such that* $\forall (\ell_1, \ell_2) \in \mathscr{L}^2, (\ell_1 \neq \ell_2 \Rightarrow \ell_1 \not\sqsubseteq \ell_2)$*. Bagnara et al. [21] present the necessary adjustments to the operations above to enforce this property.*

Based on the above definitions, we can associate to an ASTS $S = \langle X, U \uplus C, T, A, \Theta_0 \rangle$ and a given base abstract domain Λ such that $\mathcal{P}_{X\cup U\cup C} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Lambda$, a powerset extension and some abstraction and concretization functions such that $\mathcal{P}_{X\cup U\cup C} \underset{\alpha^{\wp}}{\overset{\gamma^{\wp}}{\leftrightarrows}} \wp(\Lambda)$, with $\alpha^{\wp}(d) \overset{\text{def}}{=} \bigoplus_{\delta\in\mathsf{convex}(d)} \alpha(\delta)$ and $\gamma^{\wp}(\mathscr{L}) \overset{\text{def}}{=} \bigvee_{\ell\in\mathscr{L}} \gamma(\ell)$.

Further, we shall denote elements of $\wp(\Lambda)$ as sets of abstract values surrounded by '$[\![$' and '$]\!]$'; *e.g.,* with the disjunctive interval abstract domain, $\alpha^{\wp}(x \leqslant 0 \vee 10 \leqslant x) = \{[\![x \leqslant 0]\!], [\![x \geqslant 10]\!]\}$.

Several widening operators for disjunctive domains have been proposed, by Bagnara et al. [21] for instance. Some of them have various intents such as reducing the number of disjuncts to improve performance at the expense of precision. In the sequel, we shall assume given such a widening operator, denoted $\nabla^{\wp}$.

Other operations on the base domain like $\exists^{\sharp}$ and $\forall^{\sharp}$ can be lifted to define their respective disjunctive extensions $\exists^{\wp}$ and $\forall^{\wp}$. Note that Remark 4 still holds for the operator $\forall^{\wp}$.

2) Capturing Deadlock States: Let us now present the new algorithm operating on disjunctive abstract domains.

Firstly, we base our solution on the $\mathrm{pre}^{\wp}_u \colon \wp(\Lambda) \to \wp(\Lambda)$ function, that "augments" a given disjunctive abstract value $\mathscr{L}$ with all states that would uncontrollably lead to *any one* of the convex clauses belonging to $\mathscr{L}$:

$$\mathrm{pre}^{\wp}_u(\mathscr{L}) \overset{\text{def}}{=} \exists^{\wp}_U \forall^{\wp}_C \left(T^{-1\wp}(\mathscr{L}) \sqcap^{\wp}_{\gamma} A\right)$$

where $\sqcap^{\wp}_{\gamma} \colon \wp(\Lambda) \times \mathcal{P}_{X\cup U} \to \wp(\Lambda)$ lifts the $\sqcap^{\sharp}_{\gamma}$ operator on disjunctive abstract domains.

Then, to successively capture the over-approximation $I'_{\mathfrak{F}}$ of all states to avoid based on a disjunctive abstract value, we compute $I'_{\mathfrak{F}} = \gamma^{\wp} \circ \mathrm{coreach}^{\nabla^{\wp}}_u \circ \alpha^{\wp}(\mathfrak{F})$, with

$$\mathrm{coreach}^{\nabla^{\wp}}_u(L) \overset{\text{def}}{=} \mathrm{lfp}\left(\lambda\mathscr{L}.L\nabla^{\wp}\left(\mathscr{L} \oplus \mathrm{pre}^{\wp}_u(\mathscr{L})\right)\right).$$

Note that in the computation of $I'_{\mathfrak{F}}$ above, the predicate $\mathfrak{F}$ is directly given to the abstraction function $\alpha^{\wp}$ that builds a disjunctive abstract value out of it by using convex. Thereby, $\mathrm{coreach}^{\nabla^{\wp}}_u$ operates on a disjunctive over-approximation of all states represented by $\mathfrak{F}$, which has to be opposed to the "split" algorithm we gave in [3].

The arguments used in Section III-A still hold for disjunctive over-approximations as long as every appearing deadlock state is captured in $I'_{\mathfrak{F}}$, then the above algorithm solves the

deadlock-free safety control problem on deadlock-free logico-numerical ASTSs.

We detail in appendix A an example execution of this algorithm for the problem of Example 2.

D. Enforcing Deadlock-Freeness

As is, the algorithm of Section II-C for finite ASTSs, and the above algorithm for logico-numerical ASTSs, solve the deadlock-free safety control problem as long as the original system is also deadlock-free.

Yet, if the given system is not deadlock-free, one needs to ensure that deadlocking states are made unreachable by control. Hence, in virtue of Remark 3, to enforce deadlock-freeness in every case, it is sufficient to augment the set $\mathfrak{F}$ with all states $\mathfrak{D} = \forall_I \neg A$ that are deadlocking due to the original assertion A. We then define $\mathfrak{F} \stackrel{\text{def}}{=} \neg\Phi \vee \mathfrak{D}$.

Note that in logico-numerical ASTSs involving linear constraints only, $\mathfrak{D}$ can be exactly computed by using the disjunctive power domain with convex polyhedra, according to the formula $\mathfrak{D} = \neg\gamma^{\wp}\left(\exists_I^{\wp}\alpha^{\wp}\left(A\right)\right)$.

Remark also that this technique allows to enforce deadlock-freeness of the resulting system even if $\Phi = \mathit{true}$, *i.e.,* no invariant is to be enforced.

IV. Implementation in ReaX

We have implemented the finite powerset extension of abstract domains and its associated operations, as well as all the algorithms above in the tool ReaX [3]. This tool makes use of various libraries for the representation of logico-numerical ASTSs and the manipulation of composed abstract domains. Among others, it uses the CUDD library[1] to represent (multi-terminal) binary decision diagrams (MTBDDs) [22], as well as BddApron[2] for the manipulation of power domains over convex numerical domains implemented in APRON [23].

For reasons related to the internal structure of the BddApron library, the disjunctive extension of abstract domains has been implemented on top of power domains (*i.e.,* such that $\mathcal{P}_{X\cup U\cup C} \leftrightarrows \wp\left(\mathbb{B}^n \to \mathcal{N}\right)$), with operations like universal quantification involving translations to and from MTBDDs. This extension is actually equivalent to the more "natural" one: $\mathcal{P}_{X\cup U\cup C} \leftrightarrows \mathbb{B}^n \to \wp\left(\mathcal{N}\right)$.

Our preliminary evaluation results on toy examples show that it is able to synthesize deadlock-avoiding controllers for them within a few milliseconds. Yet, the cardinality of disjunctive abstract values impacts performance results, and we still need to design more intricate examples and find case studies to assess the practicality of our solution.

V. Conclusions and Further Works

In this paper, we put forward a solution for the deadlock-free safety control problem for logico-numerical ASTSs. This solution extends our previous work on the subject [3], and relies on abstract interpretation techniques and disjunctive refinements of abstract domains to finitely represent and compute over-approximations of the set of forbidden states. It also permits to better handle cases where this set is non-convex. Our tool ReaX implements these techniques and opens the way to further experimentation.

For further works, we want to assess the practicality of our technique by conducting some performance evaluations, including on realistic use cases. We also plan to explore the use of abstract acceleration techniques [24] to overcome the loss of precision induced by the widening operations. We remark also that our algorithmic solution would certainly benefit from techniques designed to limit the number of disjuncts in abstract values. For instance, Sankaranarayanan et al. [25] proposed on the fly elaboration techniques to refine the control-flow graph of analyzed programs so as to limit the growth of the number of disjuncts. The use of other widening operators as those designed by Bagnara et al. [21] would certainly be worth investigating. At last, the partitioned polyhedral disjunctive domains designed by Ravanbakhsh and Sankaranarayanan [26] in the setting of infinite horizon safety controller synthesis, could also be an interesting trail for improving our solution.

[1] http://vlsi.colorado.edu/~fabio/CUDD/

[2] http://pop-art.inrialpes.fr/~bjeannet/bjeannet-forge/bddapron/

References

[1] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems*, vol. 77, no. 1, pp. 81–98, 1989.

[2] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer, 2007.

[3] N. Berthier and H. Marchand, "Discrete Controller Synthesis for Infinite State Systems with ReaX," in *12th Int. Workshop on Discrete Event Systems*, ser. WODES '14. IFAC, May 2014.

[4] F. Cassez, A. David, E. Fleury, K. Larsen, and D. Lime, "Efficient on-the-fly algorithms for the analysis of timed games," in *Conf. on Concurrency Theory (CONCUR)*, ser. LNCS, vol. 3653, Aug. 2005, pp. 66–80.

[5] Y. Li and W. Wohnam, "Control of vector discrete-event systems-part ii : controller synthesis," *IEEE Trans. Automatic Control*, vol. 39, no. 3, pp. 512–531, Mar. 1994.

[6] R. Kumar and V. Garg, "On computation of state avoidance control for infinite state systems in assignment program model," *IEEE Trans. on Automation Science and Engineering*, vol. 2, no. 2, pp. 87–91, 2005.

[7] R. Kumar, V. Garg, and S. Marcus, "Predicates and predicate transformers for supervisory control of discrete event dynamical systems," *IEEE Trans. Autom. Control*, vol. 38, no. 2, pp. 232–247, 1993.

[8] L. Holloway, B. Krogh, and A. Giua, "A survey of Petri net methods for controlled discrete event systems," *Discrete Event Dynamic Systems: Theory and Application*, vol. 7, pp. 151–190, 1997.

[9] T. Le Gall, B. Jeannet, and H. Marchand, "Supervisory control of infinite symbolic systems using abstract interpretation," in *Proceedings of the 44th IEEE Conference on Decision and Control*, December 2005, pp. 31–35.

[10] G. Kalyon, T. Le Gall, H. Marchand, and T. Massart, "Symbolic supervisory control of infinite transition systems under partial observation using abstract interpretation," *Discrete Event Dynamic Systems : Theory and Applications*, vol. 22, no. 2, pp. 121–161, 2011.

[11] H. Marchand and M. Samaan, "Incremental Design of a Power Transformer Station Controller Using a Controller Synthesis Methodology," *IEEE Trans. Softw. Eng.*, vol. 26, pp. 729–741, Aug. 2000.

[12] H. Marchand, P. Bournai, M. Le Borgne, and P. Le Guernic, "Synthesis of discrete-event controllers based on the signal environment," *Discrete Event Dynamic System: Theory and Applications*, vol. 10, no. 4, pp. 325–346, Oct. 2000.

[13] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proceedings of the 4th ACM Symposium on Principles of Programming Languages*, ser. POPL '77, 1977, pp. 238–252.

[14] N. Halbwachs, Y.-E. Proy, and P. Roumanoff, "Verification of Real-Time Systems using Linear Relation Analysis," *Form. Methods Syst. Des.*, vol. 11, pp. 157–185, Aug. 1997.

[15] B. Jeannet, "Dynamic Partitioning in Linear Relation Analysis: Application to the Verification of Reactive Systems," *Form. Methods Syst. Des.*, vol. 23, pp. 5–37, Jul. 2003.

[16] P. Cousot and R. Cousot, "Static determination of dynamic properties of programs," in *Proceedings of the Second International Symposium on Programming*. Dunod, Paris, France, 1976, pp. 106–130.

[17] P. Cousot and N. Halbwachs, "Automatic discovery of linear restraints among variables of a program," in *Proceedings of the 5th ACM Symposium on Principles of Programming Languages*, ser. POPL '78, 1978, pp. 84–96.

[18] P. Cousot and R. Cousot, "Systematic design of program analysis frameworks," in *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, ser. POPL '79, 1979, pp. 269–282.

[19] R. Giacobazzi and F. Ranzato, "Optimal domains for disjunctive abstract interpretation," *Science of Computer Programming*, vol. 32, no. 1, pp. 177–210, 1998.

[20] G. Filé and F. Ranzato, "The powerset operator on abstract interpretations," *Theoretical Computer Science*, vol. 222, no. 1, pp. 77–111, 1999.

[21] R. Bagnara, P. M. Hill, and E. Zaffanella, "Widening operators for powerset domains," *Int. J. Softw. Tools Technol. Transf.*, vol. 8, no. 4, pp. 449–466, Aug. 2006.

[22] J. Billon, "Perfect normal forms for discrete programs," Bull, Tech. Rep., 1987.

[23] B. Jeannet and A. Miné, "Apron: A library of numerical abstract domains for static analysis," in *Computer Aided Verification*. Springer, 2009, pp. 661–667.

[24] P. Schrammel and B. Jeannet, "Applying Abstract Acceleration to (Co-)Reachability Analysis of Reactive Programs," *J. Symb. Comput.*, vol. 47, no. 12, pp. 1512–1532, Dec. 2012.

[25] S. Sankaranarayanan, F. Ivančić, I. Shlyakhter, and A. Gupta, "Static analysis in disjunctive numerical domains," in *Static Analysis*. Springer, 2006, pp. 3–17.

[26] H. Ravanbakhsh and S. Sankaranarayanan, "Infinite horizon safety controller synthesis through disjunctive polyhedral abstract interpretation," in *Embedded Software (EMSOFT), 2014 International Conference on*. IEEE, 2014, pp. 1–10.

APPENDIX

A. Example Execution

Let us now execute the above algorithm on the problem of Example 2 with the disjunctive power domain over convex polyhedra. First, $\mathscr{L}_0 = \alpha^{\wp}(\neg\Phi) = \{[\![x \geqslant 5 \wedge y \geqslant 5]\!], [\![x \leqslant -2]\!]\}$. During the first iteration in $\text{coreach}_u^{\nabla^{\wp}}$,

$$\begin{aligned}
\mathscr{L}_0' &= \text{pre}_u^{\wp}(\mathscr{L}_0) = \exists^{\wp}_{\{i,j\}} \forall^{\wp}_{\{c\}} \left(T^{-1\wp}(\mathscr{L}_0) \sqcap^{\wp}_{\gamma} A\right) \\
&= \exists^{\wp}_{\{i,j\}} \forall^{\wp}_{\{c\}} \left\{ \begin{array}{l} [\![c \mapsto y \geqslant 5 \wedge x + i \geqslant 5, \\ \ \neg c \mapsto x \geqslant 5 \wedge y + j \geqslant 5]\!], \\ [\![c \mapsto x + i \leqslant -2, \neg c \mapsto x \leqslant -2]\!] \end{array} \right\} \\
&= \exists^{\wp}_{\{i,j\}} \left\{ \begin{array}{l} [\![x \leqslant -2 \wedge y \geqslant 5 \wedge x + i \geqslant 5]\!], \\ [\![x \geqslant 5 \wedge x + i \leqslant -2 \wedge y + j \geqslant 5]\!], \\ [\![x \leqslant -2 \wedge x + i \leqslant -2]\!], \\ [\![x \geqslant 5 \wedge y \geqslant 5 \wedge x + i \geqslant 5 \wedge y + j \geqslant 5]\!] \end{array} \right\} \\
&= \{[\![x \geqslant 5]\!], [\![x \leqslant -2]\!]\}.
\end{aligned}$$

Next, $\mathscr{L}_0'' = \mathscr{L}_0 \oplus \mathscr{L}_0' = \{[\![x \geqslant 5]\!], [\![x \leqslant -2]\!]\}$ since $[\![x \geqslant 5 \wedge y \geqslant 5]\!] \sqsubseteq [\![x \geqslant 5]\!]$ (see Remark 6). Assuming that $\nabla^{\wp}$ is the Egli-Milner enforcing widening as defined by Bagnara et al. [21], we obtain:

$$\mathscr{L}_1 = \mathscr{L}_0 \nabla^{\wp} \mathscr{L}_0'' = \{[\![x \geqslant 5]\!], [\![x \leqslant -2]\!]\}.$$

At the next iteration in $\text{coreach}_u^{\nabla^{\wp}}$,

$$\begin{aligned}
\mathscr{L}_1' &= \text{pre}_u^{\wp}(\mathscr{L}_1) = \exists^{\wp}_{\{i,j\}} \forall^{\wp}_{\{c\}} \left(T^{-1\wp}(\mathscr{L}_1) \sqcap^{\wp}_{\gamma} A\right) \\
&= \exists^{\wp}_{\{i,j\}} \forall^{\wp}_{\{c\}} \left\{ \begin{array}{l} [\![c \mapsto x + i \leqslant -2, \neg c \mapsto x \leqslant -2]\!], \\ [\![c \mapsto x + i \geqslant 5, \neg c \mapsto x \geqslant 5]\!] \end{array} \right\} \\
&= \exists^{\wp}_{\{i,j\}} \left\{ \begin{array}{l} [\![x \leqslant -2 \wedge x + i \geqslant 5]\!], \\ [\![x \geqslant 5 \wedge x + i \leqslant -2]\!], \\ [\![x \leqslant -2 \wedge x + i \leqslant -2]\!], \\ [\![x \geqslant 5 \wedge x + i \geqslant 5]\!] \end{array} \right\} \\
&= \{[\![x \geqslant 5]\!], [\![x \leqslant -2]\!]\}.
\end{aligned}$$

Eventually, $\mathscr{L}_1 \oplus \mathscr{L}_1' = \mathscr{L}_1$, so the fixpoint computation terminates and we have $I'_{\mathfrak{F}} = \gamma^{\wp}(\mathscr{L}_1) = x \geqslant 5 \vee x \leqslant 2$. The initial state does not belong to $I'_{\mathfrak{F}}$, so we can obtain a controller by computing:

$$\begin{aligned}
K_{\Phi} = \ T^{-1}\left(\neg I'_{\mathfrak{F}}\right) \wedge A = &(\neg c \wedge x > -2 \wedge x < 5) \vee \\
&(c \wedge x + i > -2 \wedge x + i < 5)
\end{aligned}$$

As expected, K_{Φ} forbids c to hold whenever $x + i \leqslant -2 \vee x + i \geqslant 5$.