Deadlock-free Discrete Controller Synthesis for Infinite State Systems
Nicolas Berthier, Hervé Marchand

To cite this version:
Nicolas Berthier, Hervé Marchand. Deadlock-free Discrete Controller Synthesis for Infinite State Systems. 54th IEEE Conference on Decision and Control, Dec 2015, Osaka, Japan. hal-01200976

HAL Id: hal-01200976
https://inria.hal.science/hal-01200976
Submitted on 5 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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 Lafontue [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 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 \( V = \langle v_1 : D_{v_1}, \ldots, v_n : D_{v_n} \rangle \) be a tuple of variables \( v \) defined on (infinite) domains \( D_v \). We note \( D_V = \prod_{i \in [1,n]} D_{v_i} \), 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 \( V = \{ b : \mathbb{B}, x : \mathbb{Z} \} \) (where \( \mathbb{B} \) and \( \mathbb{Z} \) respectively stand for the Booleans and Integers domains), the predicate \( (b \land v_x < 42) \in \mathcal{P}_V \) using the interpreted variable \( v_x \leq 42 \) can be used to represent the infinite set \( \{ (b, x) \in \mathbb{B} \times \mathbb{Z} | b \land x \leq 42 \} \).

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 \( \{ v \in D_V | P(v) \} \).

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 \( X = \mathbb{B}^k \cup \mathbb{R}^k \cup \mathbb{Z}^k \) where \( k + k' + k'' = n \) (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) \( \Theta_S \in \mathcal{P}_X \):

\[
\Theta_S = \text{lfp} \left( \lambda X. \Theta_0 \lor \text{post}(X) \right),
\]

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

\[
\text{post}(X) \triangleq \text{assign}_T (X \land A)
\]

where \( \text{assign}_T : \mathcal{P}_{X \cup I} \to \mathcal{P}_X \) 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, \( \text{assign}_T(t) = \text{false} \) (an empty set of states) if \( t = \text{false} \) (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 \( \Phi \) is an invariant of \( S \) (or \( S \) satisfies \( \Phi \)) whenever its orbit \( \Theta_S \) has an empty intersection with the complement of the state space induced by \( \Phi \), i.e., \( \neg \Phi \land \Theta_S = \text{false} \) (or equivalently, \( \Theta_S \subseteq \Phi \)). In other words, every state reachable by \( S \) satisfies \( \Phi \).

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 if \( \forall x \in \Theta_S, \exists i \in D_{\Phi}, A(x, i) \).

\[
\begin{align*}
\text{Fig. 1. Example Mealy automaton with an Integer variable.} \\
X = \langle \xi : \{ F, G \}, x : \mathbb{Z}, o : \mathbb{B} \rangle, \\
I = \langle a : \mathbb{B}, i : \mathbb{Z} \rangle,
\end{align*}
\]

![Example Mealy automaton with an Integer variable.](image-url)
B. Safety Control Problems for ASTSs

Assume given a system $S$ and a predicate $\Phi$ on $S$. Our aim is to restrict the behavior of $S$ by means of control in order to fulfill $\Phi$. 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 $\Phi$ is an invariant of the controlled system.

**Definition 4 (Basic Safety Control Problem):** Given an ASTS $S = \langle X, U \cup C, T, A, \Theta_0 \rangle$ 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:

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

The second point in the definition above states that $K_{\Phi}$ 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 $K_{\Phi}$ 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 $\Phi$, solving the deadlock-free safety control problem is to compute a controller $K_{\Phi}$ such that:

- $K_{\Phi}$ solves the basic safety control problem;
- $S' = \langle X, U \cup C, T, K_{\Phi}, \Theta_0 \rangle$ 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 $S = \langle X, U \cup C, T, A, \Theta_0 \rangle$ 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 $\Phi$ as $\mathfrak{F} \equiv \neg \Phi$. Obtaining a controller ensuring the unreachability of $\mathfrak{F}$ essentially boils down to compute the set of ultimately forbidden states $I_{\mathfrak{F}}$.

The set of forbidden states can then be computed as $\mathfrak{F} = \Theta_0 \cup \mathfrak{F}$.

The second point in the definition above states that $K_{\Phi}$ is to compute a controller $K_{\Phi}$ at least as restrictive as $A$, hence does not allow behaviors that are forbidden in the original system;

Let $\Theta_0 \cup \mathfrak{F} \neq \emptyset$, 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. $K_{\Phi}$ can be obtained as $K_{\Phi} = T^{-1}(\neg I_{\mathfrak{F}}) \land A$, 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
solve the basic controller synthesis problem for logico-numerical ASTs.

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 $\langle D, \sqsubseteq \rangle$ and $\langle \Lambda, \sqsubseteq \rangle$ be partial orders, and let $\alpha: D \to \Lambda$ and $\gamma: \Lambda \to D$ be functions. Then we say that $D \xleftarrow[\alpha]{\gamma} \Lambda$ is a Galois connection iff $\forall d \in D, \forall \ell \in \Lambda, d \sqsubseteq \gamma(\ell) \Leftrightarrow \alpha(d) \sqsubseteq \ell$. In abstract interpretation settings, $D$ is the (possibly infinite) concrete domain, and exactly represents sets of states; $\Lambda$ 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 $f^\alpha: \Lambda \to \Lambda$ the function “evaluating” a program statement $f: D \to D$ on abstract values (i.e., such that $\forall \ell \in \Lambda, f^\alpha(\ell) \sqsubseteq \alpha \circ f \circ \gamma(\ell)$).

Also, given an abstract value $\ell \in \Lambda$ and a function $f^\gamma: \Lambda \to \Lambda$ such that $\forall \ell \in \Lambda, f^\gamma(\ell) \sqsubseteq \alpha \circ f \circ \gamma(\ell)$, the widening operator $\nabla: \Lambda \times \Lambda \to \Lambda$ is such that $\text{fp}(\lambda \ell. \ell \sqcup f^\gamma(\ell)) \sqsubseteq \text{fp}(\lambda \ell. \ell \sqcup f^\alpha(\ell))$. 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 $\alpha$ and $\gamma$ such that $\mathcal{P}_{\mathcal{X} \cup \mathcal{U} \cup \mathcal{C}} \xleftarrow[\gamma]{\alpha} \Lambda$.

Elements of $\Lambda$ 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 $I^\delta$ for $S$ : 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 $T^{-1}$ the abstract version of the pre-image function $T^{-1}$ for $S$, we define $\text{pre}^\delta_\alpha: \Lambda \to \Lambda$ as

$$\text{pre}^\delta_\alpha(\ell) \equiv \text{fp}(\lambda \ell. \ell \sqcup \text{pre}^\delta_\alpha(\ell))$$

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

We have then $I^\delta = \gamma \circ \text{coreach}^\delta_\alpha \circ \alpha(\delta)$, with $\text{coreach}^\delta_\alpha: \Lambda \to \Lambda$ defined as:

$$\text{coreach}^\delta_\alpha(\ell) \equiv \text{fp}(\lambda \ell. \ell \sqcup \text{pre}^\delta_\alpha(\ell))$$

As $\text{coreach}^\delta_\alpha$ may not converge in finite time, we actually use the following over-approximating function to get $I^\delta$:

$$\text{coreach}^\delta_\alpha(\ell) \equiv \text{fp}(\lambda \ell. \ell \sqcup \text{pre}^\delta_\alpha(\ell))$$

From $I^\delta$, one can compute a controller $K_\Phi$, 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 $\forall^\delta_{\mathcal{C}, \ell}$ used in the definition of $\text{pre}^\delta_\alpha$ 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 $\delta$: “Split” Algorithm

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

To alleviate this problem, and to compute a more precise over-approximation of $\delta$, we proposed to “split” $\delta$ into a conjunction of $n$ convex polygonal clauses $\{\delta_i\}_{i \in [1,n]} = \text{conv}(\delta)$, compute every set $I^\delta_i$ independently as explained above, and then obtain the controller avoiding the states $\bigvee_{i \in [1,n]} I^\delta_i$.

Remark 5: A convex predicate $b \in \mathcal{P}_{\mathcal{V}_b \cup \mathcal{V}_c}$ is such that it can be rewritten as a conjunction $g \land f$, where $g \in \mathcal{P}_{\mathcal{V}_b}$ is a predicate on the Boolean variables $V_b$ and $f \in \mathcal{P}_{\mathcal{V}_c}$ is a conjunction of interpreted variables (that proxy arithmetic
The potential deadlocking states induced by $\gamma$ in this case can be performed by reasoning on the concrete approximating algorithm does not lead to deadlocks for $I$ than those also in $O_S$ problem and $K_I$ to $\exists x \in \Gamma \exists y \in \Theta_0$ then any potentially deadlocking state induced by $I$ free safety control problem for $S$ and set $X = \{x: \mathbb{R}, y: \mathbb{R}\}$, $U = \{i: \mathbb{R}, j: \mathbb{R}\}$, $C = \{c: \mathbb{R}\}$.

Proposition 1: The potential deadlocking states induced by $X$, noted $\text{pre}_d(X)$, are the predecessor states of $X$ that would become potential deadlocks if all states in $X$ were forbidden.

The potential deadlocking states induced by $X$ can also be defined as $\text{pre}_d(X) = \bigvee_{y \in C} (T^{-1}(X) \wedge A)$.

Proposition 2: The exact algorithm solves the deadlock-free safety control problem for $S$ and $\Phi$.

Proof. Let $\bar{\Phi} = \overline{\Phi}$. As coreach$_u$ is monotonically increasing, then any potentially deadlocking state induced by $I_S$ is in $T^{-1}\text{coreach}_u(\bar{\Phi})$; i.e., $\text{pre}_d(I_S) \subseteq I_S$.

Let $\Phi = T^{-1}(\overline{\Phi}) \wedge A$ solving the basic safety control problem and $S'$ constructed as in Definition 5. The result follows by observing that $O_S$ has not deadlocking states ($S$ is deadlock-free) and $O_{S'} \subseteq O_S$, and thus no states other than those also in $I_S$ are forbidden by $\Phi$ in $O_{S'}$.

Proposition 3: The potential deadlocking states induced by $X$, noted $\text{pre}_d(X)$, are the predecessor states of $X$ that would become potential deadlocks if all states in $X$ were forbidden.

The potential deadlocking states induced by $X$ can also be defined as $\text{pre}_d(X) = \bigvee_{y \in C} (T^{-1}(X) \wedge A)$.

Proposition 2: The exact algorithm solves the deadlock-free safety control problem for $S$ and $\Phi$.

Proof. Let $\bar{\Phi} = \overline{\Phi}$. As coreach$_u$ is monotonically increasing, then any potentially deadlocking state induced by $I_S$ is in $T^{-1}\text{coreach}_u(\bar{\Phi})$; i.e., $\text{pre}_d(I_S) \subseteq I_S$.

Let $\Phi = T^{-1}(\overline{\Phi}) \wedge A$ solving the basic safety control problem and $S'$ constructed as in Definition 5. The result follows by observing that $O_S$ has not deadlocking states ($S$ is deadlock-free) and $O_{S'} \subseteq O_S$, and thus no states other than those also in $I_S$ are forbidden by $\Phi$ in $O_{S'}$.

Proposition 3: The potential deadlocking states induced by $X$, noted $\text{pre}_d(X)$, are the predecessor states of $X$ that would become potential deadlocks if all states in $X$ were forbidden.

The potential deadlocking states induced by $X$ can also be defined as $\text{pre}_d(X) = \bigvee_{y \in C} (T^{-1}(X) \wedge A)$.

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 deadlock states. We then expose our solution for deadlock-free control in these cases by assuming that the original system does not induce deadlock 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 $S = (X, U \cup C, T, A, \Theta_0)$, and any set of states $\mathcal{X} \in \mathcal{P}_X$ and objective invariant $\mathcal{I} \in \mathcal{P}_X$.

Definition 6: The potential deadlock states induced by $\mathcal{X}$, noted $\text{pre}_d(\mathcal{X})$, are the predecessor states of $\mathcal{X}$ that would become potential deadlocks if all states in $\mathcal{X}$ were forbidden.

The potential deadlocking states induced by $\mathcal{X}$ can also be defined as $\text{pre}_d(\mathcal{X}) = \bigvee_{\mathcal{Y} \in C} (T^{-1}(X) \wedge A)$.

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

Proof. We can rewrite the above statement as $\bigvee_{\mathcal{Y} \in C} (T^{-1}(X) \wedge A) \Rightarrow \bigvee_{\mathcal{Y} \in C} (T^{-1}(X) \wedge A)$, that follows directly since $\forall \mathcal{P} \in \mathcal{P}_X U \mathcal{P}_U \mathcal{P}_C, (\bigvee_{\mathcal{Y} \in C} \mathcal{P} \Rightarrow \bigvee_{\mathcal{Y} \in C} \mathcal{P})$.

Proposition 2: The exact algorithm solves the deadlock-free safety control problem for $S$ and $\Phi$.

Proof. Let $\bar{\Phi} = \overline{\Phi}$. As coreach$_u$ is monotonically increasing, then any potentially deadlocking state induced by $I_S$ belongs to $T_S = \text{coreach}_u(\bar{\Phi})$; i.e., $\text{pre}_d(I_S) \subseteq I_S$.

Fig. 4. Clipped Graphical Representation of $\mathcal{X}$ (in plain) for the Example 2; the hashed zone depicts potentially deadlocking states that should be avoided to solve the deadlock-free control problem.
Manually examining $S_m$ and $\Phi$, one can deduce that a
correcting solution to the deadlock-free safety control problem
should at least prevent the system from reaching all states
where $x \geq 5$. 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 $\mathcal{F}$ such that $\{\mathcal{F}_1, \mathcal{F}_2\} = \text{convex}(\mathcal{F})$ as represented in Figure 5. If computed using
$\bigvee_{i \in \{1, 2\}} I_i^{\mathcal{F}}$, with $I_i^{\mathcal{F}} = \gamma \circ \text{coreach}_u \circ \alpha(\mathcal{F}_i)$, since the
universal quantifications induced by the individual fixpoints
operate on over-approximations of subsets of forbidden states,
then the resulting set of sets to avoid would not comprise
dead. In other words, for each $I_i^{\mathcal{F}}$, 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 $\Lambda$.

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 $(\Lambda, \sqsubseteq, U, \sqcup, \sqcap, \top, \bot)$, its finite powerset extension is the
join-semilattice $(\varphi(\Lambda), \sqsubseteq, \sqcup, \emptyset, \bot)$ where:

- $\varphi(\Lambda)$ denotes the powerset of $\Lambda$, restricted to finite
subsets;

- $\in$ is the Hoare partial order, i.e., such that
$(L_1 \subseteq L_2) \iff (\forall \ell_1 \in L_1, \exists \ell_2 \in L_2, \ell_1 \subseteq \ell_2)$;

- $\sqcup$ 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 $\varphi(\Lambda)$ to be the maximal ones; i.e., every
$L \in \varphi(\Lambda)$ is such that $\forall (\ell_1, \ell_2) \in L^2, (\ell_1 \neq \ell_2 \Rightarrow \ell_1 \not\subseteq \ell_2)$.
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 $S = \langle X, U \cup C, T, A, \Theta_0 \rangle$ and a given base abstract
domain $\Lambda$ such that $P_{X \cup U \cup C} \xrightarrow{\alpha} \Lambda$, a powerset extension and some abstraction and concretization functions such that $P_{X \cup U \cup C} \xrightarrow{\alpha} \varphi(\Lambda)$, with $\alpha^\varphi(d) \equiv \bigoplus_{\delta \in \text{convex}(d)} \alpha(\delta)$ and
$\gamma^\varphi(L) \equiv \bigvee_{\ell \in L} \gamma(\ell)$.

Further, we shall denote elements of $\varphi(\Lambda)$ as sets of
abstract values surrounded by ‘ ’ and ’ ’, e.g., with the
disjunctive interval abstract domain,
$\alpha^\varphi(x < 0 \lor 10 \leq x) = \{[x < 0], [x \geq 10]\}$.

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 $\nabla^\varphi$.

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

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

Firstly, we base our solution on the $\text{pre}^\varphi: \varphi(\Lambda) \rightarrow \varphi(\Lambda)$
function, that “augments” a given disjunctive abstract value
with all states that would uncontrollably lead to any one
of the convex clauses belonging to $L$:

$$\text{pre}^\varphi_\delta(L) \equiv \exists^\varphi_\delta \forall^\varphi (T^{-1} \nabla^\varphi(L) \cap \varphi_\delta \Lambda)$$

where $\exists^\varphi_\delta: \varphi(\Lambda) \times P_{X \cup U \cup C} \rightarrow \varphi(\Lambda)$ is the $\exists^\varphi$ operator on
disjunctive abstract domains.

Then, to successively capture the over-approximation $I_\delta^{\mathcal{F}}$
of all states to avoid based on a disjunctive abstract value,
we compute $I_\delta^{\mathcal{F}} = \gamma^\varphi \circ \text{coreach}_\nabla^\varphi \circ \alpha^\varphi(\mathcal{F})$, with
\begin{align*}
\text{coreach}_\nabla^\varphi(L) \equiv \text{lp}(\lambda L. L\nabla^\varphi(L \oplus \text{pre}^\varphi_\delta(L))) \circ \nabla^\varphi \Lambda.
\end{align*}

Note that in the computation of $I_\delta^{\mathcal{F}}$ above, the predicate $\mathcal{F}$
is directly given to the abstraction function $\alpha^\varphi$ that builds a
disjunctive abstract value out of it by using convex. Thereby,
$\text{coreach}_\nabla^\varphi$ operates on a disjunctive over-approximation of
all states represented by $\mathcal{F}$, 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 $I_\delta^{\mathcal{F}}$, 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 $\mathcal{F}$ with all states $D = \forall i \neg A$ that are deadlocking due to the original assertion $A$. We then define $\mathcal{F} \equiv \neg \Phi \lor D$.

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

Remark also that this technique allows to enforce deadlock-freeness of the resulting system even if $\Phi = \text{true}$, 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 Bdd Apron library, the disjunctive extension of abstract domains has been implemented on top of power domains (i.e., such that $\mathcal{P}_X \cup \mathcal{U}_C \iff \varphi(\mathbb{R}^n \to \mathcal{N})$, with operations like universal quantification involving translations to and from MTBDDs. This extension is actually equivalent to the more “natural” one: $\mathcal{P}_X \cup \mathcal{U}_C \iff \mathbb{R}^n \to \varphi(\mathcal{N})$.

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 trial for improving our solution.

REFERENCES


\(^1\)http://vlsi.colorado.edu/~fabio/CUDD/
\(^2\)http://pop-art.inrialpes.fr/~bjeannet/bjeannet-forge/bddapron/


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, \( \mathcal{L}_0 = \alpha^p (-\Phi) = \{ [x \geq 5 \land y \geq 5], [x \leq -2] \} \). During the first iteration in coreach\({\gamma}\),

\[
\mathcal{L}_0' = \text{pre}_{\gamma}^u (\mathcal{L}_0) = \exists^p_{(i,j)} \forall^p_{(c)} (T^{-1} \neg^{p} (\mathcal{L}_0) \cap \gamma_{u} A)
\]

\[
= \exists^p_{(i,j)} \forall^p_{(c)} \left\{ \begin{array}{l}
[c \mapsto y \geq 5 \land x + i \geq 5, \neg c \mapsto x \geq 5 \land j + y \geq 5,] \\
[c \mapsto x + i \leq -2, \neg c \mapsto x \leq -2] \end{array} \right\}
\]

\[
= \exists^p_{(i,j)} \left\{ \begin{array}{l}
[x \leq -2 \land y \geq 5 \land x + i \geq 5,] \\
[x \geq 5 \land x + i \leq -2 \land y + j \geq 5,] \\
[x \leq -2 \land x + i \leq -2,] \\
[x \geq 5 \land y \geq 5 \land x + i \geq 5 \land y + j \geq 5] \end{array} \right\}
\]

\[
= \{ [x \geq 5], [x \leq -2] \} .
\]

Next, \( \mathcal{L}_0'' = \mathcal{L}_0 \vee \mathcal{L}_0' = \{ [x \geq 5], [x \leq -2] \} \) since \( [x \geq 5 \land y \geq 5] \subseteq [x \geq 5] \) (see Remark 6). Assuming that \( \forall^p \) is the Egli-Milner enforcing widening as defined by Bagnara et al. [21], we obtain:

\[
\mathcal{L}_1 = \mathcal{L}_0 \forall^p \mathcal{L}_0'' = \{ [x \geq 5], [x \leq -2] \} .
\]

At the next iteration in coreach\({\gamma}\),

\[
\mathcal{L}_1' = \text{pre}_{\gamma}^u (\mathcal{L}_1) = \exists^p_{(i,j)} \forall^p_{(c)} (T^{-1} \neg^{p} (\mathcal{L}_1) \cap \gamma_{u} A)
\]

\[
= \exists^p_{(i,j)} \forall^p_{(c)} \left\{ \begin{array}{l}
[c \mapsto x + i \leq -2, \neg c \mapsto x \leq -2,] \\
[c \mapsto x + i \geq 5,\neg c \mapsto x \geq 5] \end{array} \right\}
\]

\[
= \exists^p_{(i,j)} \left\{ \begin{array}{l}
[x \leq -2 \land x + i \geq 5,] \\
[x \geq 5 \land x + i \leq -2,] \\
[x \leq -2 \land x + i \leq -2,] \\
[x \geq 5 \land x + i \geq 5] \end{array} \right\}
\]

\[
= \{ [x \geq 5], [x \leq -2] \} .
\]

Eventually, \( \mathcal{L}_1 \cup \mathcal{L}_1' = \mathcal{L}_1 \), so the fixpoint computation terminates and we have \( I^p_\Phi = \gamma^p (\mathcal{L}_1) = x \geq 5 \lor x \leq -2 \). The initial state does not belong to \( I^p_\Phi \), so we can obtain a controller by computing:

\[
K_\Phi = T^{-1} (-I^p_\Phi) \land A = (\neg c \land x \geq -2 \land x < 5) \lor (c \land x + i \geq -2 \land x + i < 5)
\]

As expected, \( K_\Phi \) forbids \( c \) to hold whenever \( x + i \leq -2 \lor x + i \geq 5 \).