Infinite Horizon Safety Controller Synthesis through Disjunctive Polyhedral Abstract Interpretation.

Hadi Ravanbakhsh and Sriram Sankaranarayanan
University of Colorado, Boulder, USA
firstname.lastname@colorado.edu

ABSTRACT

This paper presents a controller synthesis approach using disjunctive polyhedral abstract interpretation. Our approach synthesizes infinite time-horizon controllers for safety properties with discrete-time, linear plant model and a switching feedback controller that is suitable for time-triggered implementations. The core idea behind our approach is to perform an abstract interpretation over disjunctions of convex polyhedra to identify states that are potentially uncontrollable. Complementing this set yields the set of controllable states, starting from which, the safety property can be guaranteed by an appropriate controller feedback function. Since, a straightforward disjunctive domain is computationally inefficient, we present an abstract domain based on a state partitioning scheme that allows us to efficiently control the complexity of the intermediate representations. Next, we focus on the automatic generation of controller implementation from the abstract interpretation results. We show that a balanced tree approach can yield efficient controller code with guarantees on the worst-case execution time. Finally, we evaluate our approach on a suite of benchmarks, comparing different instantiations with related synthesis tools. The evaluation shows that our approach can successfully synthesize controller implementations for small to medium sized benchmarks.

Categories and Subject Descriptors

D.2.4 [**Software Program Verification**]: [Formal Methods]; I.2.2 [**Automatic Programming**]: [Program Synthesis]; I.2.8 [**Problem Solving, Control Methods and Search**]: [Control Theory]; J.7 [**Computers in Other Systems**]: [Command and Control, Process Control, Real Time]

Keywords

Switched Systems, Hybrid Systems, Controller Synthesis, Safety, Abstract Interpretation.

1. INTRODUCTION

This paper focuses on the correct-by-construction synthesis of controllers for safety properties of switched affine systems. Our approach assumes a discrete-time, state feedback controller for a discretized model of a hybrid plant. The controller is allowed to switch between the discrete modes of the plant as dictated by the transition relation of the plant. Additionally, our approach includes a disturbance input that models the uncertainty in sensing the state of the plant. Our approach synthesizes a controller in terms of a control invariant set (viability kernel) and a switching function that guarantees that once the system is inside the control invariant, it remains so, no matter what inputs are provided by the disturbance.

The key contribution of our approach is an abstract interpretation based framework for computing the disturbance invariant set that characterizes all those states from which the disturbance can force the plant to potentially violate the safety property, regardless of what control inputs are provided. Complementing this set, yields the required control invariant. Computing disturbance invariants allows us to use existing numerical abstract domains that are based on over-approximating least fixed points [7, 9, 17, 25, 31].We show that a simple polyhedral domain often fails to compute useful disturbance invariants, whereas a fully disjunctive domain is prohibitively expensive for all but the smallest systems. Therefore, we present a disjunctive domain structured by a partitioning of the state-space into finitely many polyhedra. Our disjunctive domain then maintains a single disjunct in each partition. We employ the standard abstract interpretation framework using widening to enforce termination to a post-fixed point. The resulting disturbance invariant is used to compute a controller implementation using a balanced decision diagram synthesis algorithm that uses the properties of the partitioning scheme to compute a decision tree of small depth. This tree encodes the real-time decisions made by the controller to choose a control action. Finally, our approach is implemented inside a prototype tool that reads in the description of a switched affine system and a safety specification, to output a controller implementation in a target environment.

Our approach is evaluated on a series of benchmarks taken from the related work with upto 6 state variables, and tens of discrete modes. We note that our approach using precondition-guided partitioning outperforms a fixed point computation using the standard disjunctive polyhedral domain [4] implemented in Parma Polyhedral Library [5], and a simpler grid-based partitioning approach. Our approach compares favorably to the related PESSOA tool for synthesizing hybrid controllers [24].

Organization The rest of the paper is organized as follows. Section 2 presents the plant model, and preliminary notions. Subsequently, section 3 presents the disjunctive polyhedral domain for computing disturbance invariants. Next, we present the partitioned disjunctive domain and the precondition-guided partitioning scheme in Section 4. Section 5 presents the derivation of the controller implementation from the abstract interpretation results. The prototype

ESWEEK'14 October 12 - 17 2014, New Delhi, India

http://dx.doi.org/10.1145/2656045.2656060.

implementation of our ideas and its evaluation over benchmarks are presented in Section 6.

Related Work The problem of synthesizing a disturbance invariant reduces to finding a winning set for a disturbance player who seeks to ensure that the system reaches an unsafe set in finitely many steps, as opposed to a control player who tries to ensure that the system remains within the safe set. This problem has been well studied for finite state systems using ideas from logic and automata theory [15]. For continuous systems, game-based formulations using differential games and Hamilton-Jacobi PDEs have been studied for controller synthesis [21]. These PDEs can be solved numerically for small but complex systems using level set methods [26].

The fixed point computation for control invariants (viability kernels) has been studied by many authors. Asarin et al. used a control precondition iteration to compute control invariants for timed automata [2]. Subsequently, they present extensions to affine hybrid systems using flowpipe computations [1]. Approximation in the latter work is addressed by using a domain of "griddy" polyhedra which are disjoint union of finitely many boxes. In contrast to griddy polyhedra, our domain allows proper polyhedral subsets of a grid cells rather than including/excluding entire cells. Wong-Toi uses the idea of over-approximating disturbance invariants rather than under-approximating control invariants. This is carried out for linear hybrid automata models with piecewise constant dynamics [37]. Vidal et al. use quantifier elimination over reals to compute control preconditions. They show termination if the system is linear in controllable canonical form [36]. A general drawback of the iterative approaches discussed thus far is that the complexity of the representation can be quite high, and termination of the iteration is not guaranteed in the general case.

Our approach is very closely related to the earlier work of LeGall et al. using abstract interpretation with widening in the supervisory control framework [20]. Their plant model is infinite state, consisting of controllable and uncontrollable events. Their approach seeks to build a supervisor that enables the controllable events to ensure that a safety property defined on the language of events is maintained. Notable differences exist in the problem domains: continuous disturbances in our approach versus uncontrolled events in theirs, a time triggered controller in our setup versus an event triggered approach. Beyond these, our work focuses building a disjunctive polyhedral domain suitable for computing disturbance invariants by trading off the complexity of the representation with computational complexity. Finally, we present a detailed evaluation of our approach on a suite of small and medium sized examples.

Many methods have used disjunctions of convex polyhedra for finite horizon control. The explicit MPC approach pioneered by Morari et al. is available as part of the popular Multi-Parametric Toolbox [18]. Maidens et al. use flowpipe construction to over approximate the (finite time) uncontrollable sets, and thus under approximate the finite time control invariant [22]. In comparison, our approach obtains infinite horizon controllers.

As mentioned earlier, the Pessoa approach by Tabuada et al. uses finite state abstractions that are approximately bisimilar to the original system [24]. On one hand, Pessoa handles a larger class of properties and systems, but without disturbance inputs. The gridding of the state-space produces large finite state systems over which winning positions for various players are computed. Sets of states are represented using binary decision diagrams. The advantage of using an abstract interpretation setup, as in this paper, is to separate the process of fixed point computation from the representations used for sets of states. Furthermore, widening/narrowing operators are useful for enforcing the termination of the fixed point.

A constraint-based approach to correct-by-construction synthesis of controllers for hybrid systems was provided by Taly et al. [34]. Their approach assumes a closed loop model with parameters representing various gains and switching surfaces of the controller, and reduces the problem to that of solving a system of non-linear constraints. As such, their approach is event triggered, whereas ours is time triggered. Furthermore, their approach is limited by the ability to solve non-linear constraints, while ours is restricted to linear systems and uses a state-space partition. More generally, the constraint-based approach for infinite state games has been explored in the context of program synthesis by Beyene et al. [6].

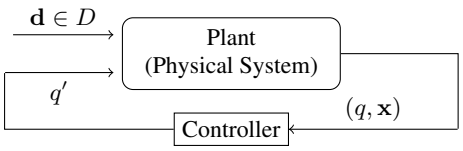


Figure 1: The closed loop model of the plant and the controller.

Disjunctive domains have been well studied in static analysis for verifying properties of numerical programs Broadly, the dynamic partitioning work by Bertrand Jeannet defines a disjunctive domain that identifies useful predicates and defines disjuncts based on these predicates [19]. As such their approach applies to control loops written in synchronous languages, wherein the central purpose of the disjunctive analysis is to recover partial control flow information that is encoded in the program's state variables. This is not the case in our work, wherein disjunctions arise from differences in the control strategies used by the disturbance player at different places. Nevertheless, our approach can be thought of as a "static" partitioning scheme. A few disjunctive domains used in program analysis include *trace partitioning* schemes that annotate polyhedra with fragments of the path taken through the control-flow graph [23], disjunctive polyhedral domain [4], disjunctive domains on so-called "elaborations" of the program's control flow [30] and using affinity metrics to determine if two polyhedra can be joined using a convex-hull or maintained as separate disjuncts [28].

2. PRELIMINARIES

In this section, we define the plant and controller models, recall the standard notions of *control* and *disturbance* invariants, expressing them as fixed points that can be approximated using abstract interpretation theory [8, 27].

2.1 System Model

We consider a closed-loop system with two parts, namely a *plant* and a *controller*. The plant and the controller form a feedback system as shown in Figure 1. The plant is *hybrid* with a continuous state $\mathbf{x} \in \mathbb{R}^n$, and a discrete mode $q \in Q$. Let $\mathbf{x}$ denote the state variables as a $n \times 1$ column vector, and x_i represent the i^{th} component of $\mathbf{x}$. The plant admits a drift input $\mathbf{d} \in D$ and a control input from the controller. The controller is assumed to be a discrete system that provides a new discrete input to the plant every Δ time units (see Figure 1). The plant responds to the control input by switching to the mode commanded by the controller. Formally, we model the plant as a switched affine (discrete-time) system:

DEFINITION 1. *A* discrete-time switched affine system *is a tuple* $\Psi : (Q, X, D, E, A, B)$ *described by: (a)* continuous states $X \subseteq \mathbb{R}^n$*, where* n *is the number of continuous state variables; (b) a finite set of* discrete modes Q*; (c)* disturbance inputs $D \subseteq \mathbb{R}^n$*, where* D *is a convex polyhedron in* $\mathbb{R}^n$*; (d) set of* edges $E \subseteq Q \times Q$*; and (e)* continuous transition matrices $A_q \in \mathbb{R}^{n \times n}$ *and* $B_q \in \mathbb{R}^{n \times 1}$ *for each* $q \in Q$.

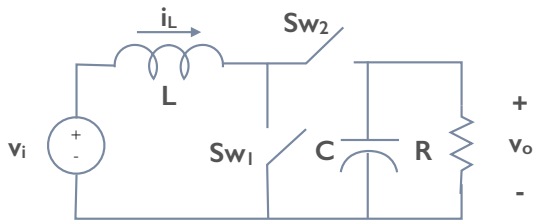


Figure 2: Circuit of a DC-DC converter

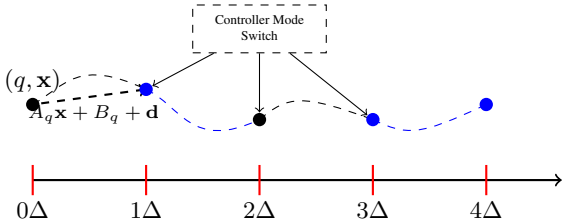


Figure 3: Illustration of the closed loop semantics.

We now specify the controller as a *memoryless* state-feedback law modeled by a controller switching function:

DEFINITION 2 (CONTROLLER SWITCHING FUNCTION). *Given a plant model Ψ, a switching function* $\mathsf{switch} : Q \times X \to Q$ *satisfies the constraint* $(\forall q \in Q)(\forall \mathbf{x} \in X)\ (q, \mathsf{switch}(q, \mathbf{x})) \in E$.

The controller switching function defines a *state-feedback law* that uses the current state $(q, \mathbf{x})$ of the plant to command the next mode $\mathsf{switch}(q, \mathbf{x})$ of the plant, respecting the edge relation E.

EXAMPLE 1 (DC-DC CONVERTER). *Figure 2 shows the circuit diagram for an ideal DC-DC converter adapted from Senesky et al. [32]. There are two control switches sw_1, sw_2. The controller switches them periodically to ensure that the state variables i_L and v_o, modeling the inductor current and output voltages are within some permissible values. The plant model has two discrete modes $Q = \{q_1, q_2\}$, wherein $q_1 = [sw_1$ is on, sw_2 is off$]$ and $q_2 = [sw_1$ is off, sw_2 is on$]$. The edge set $E : Q \times Q$ denotes that the controller can switch between these two states with no restrictions by sensing the values of (i_L, v_0) periodically.*

The disturbance $D : [-0.01, 0.01] \times [-0.01, 0.01]$ models the sensor errors for (i_L, v_0). Assuming a $50KHz$ controller, we discretize the dynamics for the modes q_1, q_2 with a time step $\Delta = 0.02ms$. Using $v_{in} = 1.5V$, $R = 6\Omega$, $L = 150\mu H$ and $C = 110\mu F$, the transitions matrices are

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0.22 \end{bmatrix}}_{A_{q_1}} \underbrace{\begin{bmatrix} 10 \\ 0 \end{bmatrix}}_{B_{q_1}} \underbrace{\begin{bmatrix} 0.095 & -0.4 \\ 0.55 & 0.004 \end{bmatrix}}_{A_{q_2}} \underbrace{\begin{bmatrix} 0.83 \\ 1.36 \end{bmatrix}}_{B_{q_2}} \quad (1)$$

Closed-Loop Semantics: Given a mode q, state $\mathbf{x}$ and disturbance input d, we denote the continuous state update in a step by $\mathsf{next}(q, \mathbf{x}, \mathbf{d}) : A_q\mathbf{x} + B_q + \mathbf{d}$. A *state* of the plant is a tuple $(q, \mathbf{x})$ wherein $q \in Q$ and $\mathbf{x} \in X$. An execution of the closed loop is an infinite sequence of plant states $(q_0, \mathbf{x}_0) \xrightarrow{\mathbf{d}_1} (q_1, \mathbf{x}_1) \xrightarrow{\mathbf{d}_2} (q_2, \mathbf{x}_2) \cdots$, such that (A) For each $i \in \mathbb{N}$, $q_i \in Q$ and $\mathbf{x}_i \in X$; (B) Forall $i \in \mathbb{N}$, there is a drift input $\mathbf{d}_{i+1} \in D$, such that $\mathbf{x}_{i+1} = \mathsf{next}(q_i, \mathbf{x}_i, \mathbf{d}_{i+1})$; and (C) For each $i \in \mathbb{N}$, $q_{i+1} = \mathsf{switch}(q_i, \mathbf{x}_i)$.

Figure 3 illustrates our closed loop semantics. The controller executes every Δ time units, which is also the discretization time step for the plant model. For simplicity, the time to compute the switching function is assumed negligible. We now consider the controller synthesis problem.

DEFINITION 3 (CONTROLLER SYNTHESIS PROBLEM). *Given Ψ, a plant model and a* safe set $S \subseteq Q \times X$, *the controller synthesis problem computes a subset $P \subseteq S$, called the* control invariant, *and a switching function* $\mathsf{switch} : Q \times X \to Q$, *such that, for each run $(q_0, \mathbf{x}_0) \xrightarrow{\mathbf{d}_1} (q_1, \mathbf{x}_1) \xrightarrow{\mathbf{d}_2} \cdots \xrightarrow{\mathbf{d}_j} (q_j, \mathbf{x}_j) \cdots$, if $(q_0, \mathbf{x}_0) \in P$ then $(q_i, \mathbf{x}_i) \in P$ for all $i \in \mathbb{N}$ and for all $\mathbf{d}_1, \mathbf{d}_2, \cdots \in D$.*

EXAMPLE 2 (SAFETY SPECIFICATION). *Consider again the plant model of the DC-DC controller from Example 1. The controller tries to maintain $i_L \in [0, 2.5]mA$ and $v_o \in [2.97, 3.63]V$ (around $3.3V$ with 10% error), independent of the control mode. The set S is therefore, $Q \times ([0, 2.5] \times [2.97, 3.63])$.*

This problem is solved by constructing a *control invariant set*, w.r.t to a plant Ψ and a safety specification S.

DEFINITION 4 (CONTROL INVARIANT SET). *A set $C \subseteq Q \times X$ is a control invariant set for $\langle \Psi, S \rangle$ iff (a) $C \subseteq S$, and (b) starting from every state $(q, \mathbf{x}) \in C$, there is a possible next mode q', such that the next state produced by the joint action of the plant and the mode switch belongs to C:*

$$(\forall_{(q,\mathbf{x})\in C})(\exists_{q'\in Q})(\forall_{\mathbf{d}\in D}) \left(\begin{array}{c} (q, q') \in E \wedge \\ (q', \mathsf{next}(q, \mathbf{x}, \mathbf{d})) \in C \end{array} \right). \quad (2)$$

Skolemizing Eq. (2), we immediately conclude that there exists a controller switching function switch such that for all initial states $(q_0, \mathbf{x}_0) \in C$ and for all runs of the closed loop system: $(q_0, \mathbf{x}_0) \xrightarrow{\mathbf{d}_1} \cdots \xrightarrow{\mathbf{d}_n} (q_n, \mathbf{x}_n)$, the state $(q_n, \mathbf{x}_n) \in C, \forall n \in \mathbb{N}$.

Rather than control invariant, our approach here will focus, instead, on computing *disturbance invariants*, which are sets of states from which no control strategy can prevent the disturbance from ensuring that an unsafe state is reached. The advantage of disturbance invariant computation is that it allows us to perform over-approximations, and thus, reuse a large body of work on numerical domains for abstract interpretations.

Let Ψ be a system with a desired safe set S. Let $U = X \setminus S$ represent the state space complement of S. Since S refers to the safe set, the set U is referred to as the *unsafe* set.

DEFINITION 5 (DISTURBANCE PRECONDITION). *The* disturbance pre-condition *of a set P denoted by* $\mathsf{dpre}(P)$ *is defined as:*

$$\left\{ (q, \mathbf{x}) | \ (\forall_{q'\in Q})\ (\exists_{\mathbf{d}\in D})\ \left(\begin{array}{c} (q, q') \in E \Rightarrow \\ (q', \mathsf{next}(q, \mathbf{x}, \mathbf{d})) \in P \end{array} \right) \right\}.$$

The disturbance precondition of $\mathsf{dpre}(P)$ represents those states $(q, \mathbf{x})$ from which the disturbance has a strategy to ensure that the plant remains in the set P for the next step, *no matter which action is taken by the controller*.

DEFINITION 6. *A set $W \subseteq Q \times X$ is a disturbance invariant iff (1) $U \subseteq W$; and (2)* $\mathsf{dpre}(W) \subseteq W$.

THEOREM 1. *W is a disturbance invariant of a system Ψ iff $X \setminus W$ is a control invariant. There exists a disturbance invariant set W^* such that for every disturbance invariant W, $W^* \subseteq W$.*

All proofs are provided in an extended version available upon request. As a direct result, we can compute a disturbance invariant W, and complement it to obtain a control invariant.

Concrete	Abstract	Soundness requirement
$W_1 \subseteq W_2$	$a_1 \sqsubseteq a_2$	$(a_1 \sqsubseteq a_2) \Rightarrow \gamma(a_1) \subseteq \gamma(a_2)$
$W_1 \cup W_2$	$a_1 \sqcup a_2$	$(\gamma(a_1) \cup \gamma(a_2)) \subseteq \gamma(a_1 \sqcup a_2)$
$W_1 \cap W_2$	$a_1 \sqcap a_2$	$(\gamma(a_1) \cap \gamma(a_2)) \subseteq \gamma(a_1 \sqcap a_2)$
$\mathsf{dpre}(W_1)$	$\widetilde{\mathsf{dpre}}(a_1)$	$\mathsf{dpre}(\gamma(a_1)) \subseteq \gamma(\widetilde{\mathsf{dpre}}(a_1))$

Table 1: Basic Abstract Domain Operations corresponding concrete operations over sets.

2.2 Disturbance Invariants as Fixed Points

We now present a fixed point characterization of disturbance invariants. Consider the complete lattice of sets of states $P \subseteq X$ ordered by set inclusion $\subseteq$. First, we define an operator $\mathcal{G} : 2^X \rightarrow 2^X$, on the lattice, whose fixed points correspond to disturbance invariants: $\mathcal{G}(P) : U \cup (\mathsf{dpre}(P))$. A set W is a post-fixed point of $\mathcal{G}$ iff $\mathcal{G}(W) \subseteq W$

LEMMA 2.1. *The operator $\mathcal{G}$ is a monotone operator on the lattice of sets: $C_1 \subseteq C_2 \Rightarrow \mathcal{G}(C_1) \subseteq \mathcal{G}(C_2)$.*

A set W is a disturbance invariant iff it is a post-fixed point of $\mathcal{G}$: $\mathcal{G}(W) \subseteq W$. Finally, the minimal disturbance invariant W^* exists, and is the least fixed point of the monotone operator $\mathcal{G}$.

Therefore, we may use *Kleene iteration* to attempt to compute the least fixed point W^* over sets of states, starting from the empty set $W_0 : \emptyset$. $W_0 : \emptyset$, $W_1 : \mathcal{G}(W_0)$, $W_2 : \mathcal{G}(W_1)$, $\cdots, W_n = \mathcal{G}(W_{n-1})$. The process terminates if $W_n \subseteq W_{n-1}$ for some $n \geq 1$, and upon termination, we obtain $W_n = W^*$, the least fixed point of the operator $\mathcal{G}$, and by Lemma 2.1, the minimal disturbance invariant. The set $C^* = X \setminus W^*$ yields the required control invariant region from which a controller can be obtained by keep the system in the safe region regardless of the actions of the disturbance.

Direct Kleene iteration is not possible for all but the simplest of systems: (a) We require a representation for arbitrary sets of states W_i, and the ability to compute the operator $\mathcal{G}$ on this representation. Representing these sets in the computer can be prohibitively expensive or outright impossible. (b) The Kleene iteration often does not converge to a result in finitely many steps. This means that the iteration can go on forever.

We resort to abstract interpretation to solve both these problems and enable us to compute a disturbance invariant set W that is an over-approximation of the smallest set W^*. The set $\overline{W} : X \setminus W$ is the required control invariant.

2.3 Abstract Interpretation

The framework of abstract interpretation was proposed by Cousot and Cousot [8,27]. It is especially effective in providing conservative approximations of fixed points defined over sets of states. Common numerical abstract domains include intervals [7], octagons [25] , convex polyhedra [9, 17], template polyhedra [31] and ellipsoids [29]. We provide a brief introduction, glossing over many details discussed elsewhere [8, 27].

Abstract Domains: Rather than allowing arbitrary sets $W \subseteq Q \times X$, we will restrict ourselves to a smaller class of sets that can be represented and manipulated computationally. Examples of possible representations for subsets of $\mathbb{R}^n$ include *convex polyhedra*, *ellipsoids*, *intervals*, *octagons* and *template polyhedra*. Formally, we fix an abstract lattice A ordered by inclusion $\sqsubseteq$. This lattice is linked to the concrete lattice $\langle 2^{Q \times X}, \subseteq \rangle$ by means of (a) *Abstraction* function $\alpha : 2^{Q \times X} \rightarrow A$ maps every subset $W \subseteq Q \times X$ to its *representative* element in A, and (b) *Concretization* function $\gamma : A \rightarrow 2^{Q \times X}$ maps every element $a \in A$ back to its concretization. The pair $\langle \alpha, \gamma \rangle$ satisfies the *Galois connection* property: $(\forall\, S \in 2^{Q \times X})\ (\forall\, a \in A)\ (\alpha(S) \sqsubseteq a)$ iff $S \subseteq \gamma(a)$.

Corresponding to the set operations of $S_1 \cup S_2$ (union), $S_1 \cap S_2$ (intersection), $\mathsf{dpre}(S_1)$ (disturbance precondition), and $S_1 \subseteq S_2$ (inclusion checking) that are required to compute the monotone operator $\mathcal{G}$ over the concrete lattice $\langle 2^{Q \times X}, \subseteq \rangle$, we define the abstract versions, as depicted in Table 1.

Thus, the concrete operator $\mathcal{G}(W) : U \cup (\mathsf{dpre}(W))$, can be replaced by an abstract operator: $\widehat{\mathcal{G}}(a) : \alpha(U) \sqcup \left[\widetilde{\mathsf{dpre}}(a)\right]$.

LEMMA 2.2. *For all $a \in A$, $\mathcal{G}(\gamma(a)) \subseteq \gamma(\widehat{\mathcal{G}}(a))$.*

THEOREM 2. *Let $a \in A$ be a post-fixed point of the $\widehat{\mathcal{G}}$ operator: $\widehat{\mathcal{G}}(a) \sqsubseteq a$. It follows that $\gamma(a)$ is a disturbance invariant satisfying $\mathcal{G}(\gamma(a)) \subseteq \gamma(a)$.*

Abstract Kleene Iteration: The concrete Kleene iteration sequence is replaced by an abstract Kleene iteration sequence using the $\widehat{\mathcal{G}}$ operator on A: $(a_{n+1} = \widehat{\mathcal{G}}(a_n)$ starting from $a_0 : \alpha(\emptyset))$, until we reach an abstract post fixed point $a_{n+1} \sqsubseteq a_n$. Following theorem 2, we conclude that $\gamma(a_{n+1})$ is a disturbance invariant.

Termination: In general, unless the lattice A is finite or has the *ascending chain condition*, the abstract Kleene iteration is not guaranteed to terminate. Therefore, abstract interpretation uses a *widening* operator ∇ over A to *force* termination in finitely many steps. The widening operator $a_1 \nabla a_2$ for $a_1, a_2 \in A$ satisfies the following properties: (1) $a_1 \sqsubseteq a_1 \nabla a_2$ and $a_2 \sqsubseteq a_1 \nabla a_2$. (2) For any finite ascending sequence $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \sqsubseteq \cdots$, the *widened sequence* $b_n : b_{n-1} \nabla a_n$ with $b_0 : a_0$, terminates to yield $b_{n+1} \sqsubseteq b_n$. To enforce termination, the Kleene iteration is combined with widening: $a_{n+1} = a_n \nabla \widehat{\mathcal{G}}(a_n)$ with $a_0 : \alpha(\emptyset)$. Often, a delayed widening strategy is applied after finite number of steps of Kleene iteration is carried out without widening. Other strategies for the optimal application of widening, especially over convex polyhedra are discussed by Bagnara et al. [3].

3. POLYHEDRAL AND DISJUNCTIVE POLYHEDRAL DOMAINS

In this section, we discuss convex polyhedra as an abstract domain for computing disturbance invariants using abstract interpretation. We first consider the single polyhedron per discrete mode, and discuss the computation of domain operators, especially the disturbance precondition $\widetilde{\mathsf{dpre}}$. However, a single convex polyhedron is insufficient to represent disturbance invariants. Therefore, we make the case for a disjunctive domain that uses unions of finitely many convex polyhedra. However, the standard disjunctive domain is too expensive. This motivates us to consider specialized types of disjunctive domains.

Convex Polyhedra: The convex polyhedral domain was originally investigated by Cousot & Halbwachs and later by Halbwachs et al. [9, 17]. Mathematically, a convex polyhedron is given as a conjunction of linear inequalities. Let $\mathbf{x} : (x_1, \ldots, x_n)$ represent a set of variables. A *linear inequality* over variables $\mathbf{x}$ is an inequality of the form $\mathbf{a}^t\mathbf{x} \leq b$ for $\mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}$. A *linear assertion* ψ is a conjunction of finitely many linear inequalities: $\psi : \bigwedge_{j=1}^{m} \mathbf{a}_j^t\mathbf{x} \leq b_j$. A linear assertion is succinctly written in the matrix form as $A\mathbf{x} \leq \mathbf{b}$. Given linear assertion ψ, a *convex polyhedron* is a set of points that satisfy it: $[\![\psi]\!] : \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \models \psi\}$. Let $\text{POLY}(X)$ represent all convex polyhedra $P \subseteq X$.

DEFINITION 7 (CONVEX POLYHEDRA DOMAIN). *The convex polyhedral domain consists of all maps* $\eta : Q \to \text{POLY}(X)$ *mapping each mode* q *to a convex polyhedron* $\eta(q) \in \text{POLY}(X)$. *We denote* $\eta_1 \sqsubseteq \eta_2$ *iff* $(\forall q \in Q)\ \eta_1(q) \subseteq \eta_2(q)$.

Assume a function ConvexPolyhedralHull(P) that takes a set $P \subseteq X$, and returns a convex polyhedron over-approximation. Numerous strategies exist for such approximations, including approaches based on templates [25] and support functions [16]. Given any subset $W \subseteq Q \times X$, its abstraction α maps W to a η_W wherein

$$\alpha(W) :\ \lambda q.\ \text{ConvexPolyhedralHull}\left(\{\mathbf{x} \in X \mid (q, \mathbf{x}) \in W\}\right) .$$

The concretization of a map η is given by $\gamma(\eta) :\ \{(q, \mathbf{x}) \mid q \in Q,\ \text{and } \mathbf{x} \in \eta(q)\}$. Join ($\sqcup$), meet($\sqcap$) and widening($\nabla$) operators on maps extends mode-wise. More precisely, for each operator $\oplus$ we define: $(\eta_1 \oplus \eta_2) :\ \lambda q.\ (\eta_1(q) \oplus \eta_2(q))$. The problem of representing, computing intersections, convex hull, widenings and projection is well understood in the context of abstract interpretation [3, 17]. Efficient *sub-polyhedral* domains have been investigated including two variables per inequalities [33] and template polyhedra [31] to mention a few. We now derive the abstract $\widetilde{\mathsf{dpre}}(\eta)$ operator over the convex polyhedral domain.

Disturbance Precondition Operator: We are given a map η representing a subset of $Q \times X$ and wish to compute $\widetilde{\mathsf{dpre}}(\eta)$ so that for all η, $\mathsf{dpre}(\gamma(\eta)) \subseteq \gamma(\widetilde{\mathsf{dpre}}(\eta))$. Recall the definition of the disturbance precondition of a set P (Def. 5), $\mathsf{dpre}(P)$:

$$\left\{(q, \mathbf{x}) \mid (\forall_{q' \in Q})\ (\exists_{\mathbf{d} \in D})\ \left(\begin{array}{c} (q, q') \in E\ \Rightarrow \\ (q', \mathsf{next}(q, \mathbf{x}, \mathbf{d})) \in P \end{array} \right) \right\} .$$

Let $\eta : Q \mapsto \text{POLY}(X)$ be a map. Our goal is to compute a map $\hat{\eta} : Q \mapsto \text{POLY}(X)$ that represents $\widetilde{\mathsf{dpre}}(\eta)$.

Let $\psi : P\mathbf{x} \leq \mathbf{r}$ be a linear assertion and $\mathbf{x}' := A_q\mathbf{x} + B_q + \mathbf{d}$ be the linear transformation associated with mode q with disturbance $\mathbf{d} \in D$. We define the *pre-image* of ψ w.r.t q as the linear assertion:

$$\mathsf{pre}(q, \psi, \mathbf{d}) :\ P(A_q\mathbf{x} + B_q + \mathbf{d}) \leq \mathbf{r} \ \equiv\ PA_q\mathbf{x} \leq \mathbf{r} - PB_q - P\mathbf{d} .$$

Let us consider the convex polyhedron $\hat{\eta}(q)$ for a fixed $q \in Q$. Let $\{(q, q_1), \ldots, (q, q_k)\} \subseteq E$ be the set of outgoing edges of q.

$$\hat{\eta}(q) :\ \bigcap_{(q,q_j) \in E} (\exists\ \mathbf{d} \in D)\ \mathsf{pre}(q, \eta(q_j), \mathbf{d}) . \tag{3}$$

LEMMA 3.1. *The disturbance precondition* $\widetilde{\mathsf{dpre}}(\eta)$ *for* $\eta : Q \to \text{POLY}(X)$ *is a map* $\hat{\eta} : Q \to \text{POLY}(X)$ *according to* (3). *Furthermore,* $\widetilde{\mathsf{dpre}}$ *satisfies the soundness condition:*

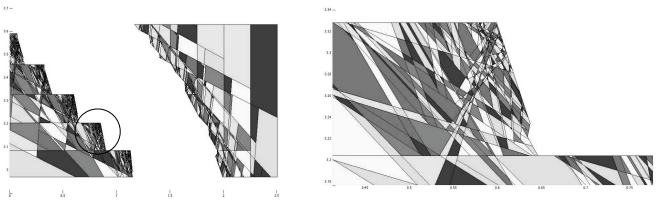
$$(\forall \eta : Q \to \text{POLY}(X))\ \mathsf{dpre}(\gamma(\eta)) \subseteq \gamma(\widetilde{\mathsf{dpre}}(\eta))$$

Whereas convex polyhedra are a natural abstract domain for subsets of $\mathbb{R}^n$, the convexity poses challenges for computing disturbance invariants. Therefore, we resort to disjunctive polyhedra that use a disjunction of multiple polyhedra for each discrete mode.

3.1 Disjunctive Polyhedral Domains

To circumvent the limitations imposed by a convex set representation for possibly non-convex set, we allow our approach to use a finite union of polyhedra as abstractions of sets. In theory, the disjunctive polyhedral domain works by simply replacing a single polyhedron per discrete mode to a finite disjunction of polyhedra per discrete mode. Let $\mathsf{fin}(P)$ represent all finite subsets of a set P.

DEFINITION 8 (CONVEX POLYHEDRA DOMAIN). *The disjunctive polyhedral domain consists of all maps* $\pi : Q \to \mathsf{fin}(\text{POLY}(X))$ *mapping each mode* $q \in Q$ *to a finite set of linear assertions* $\pi(q) : \{P_1, \ldots, P_k\}$, *where* $P_1, \ldots, P_k \in \text{POLY}(X)$.



(a) W[17] (b) Scaled W[17]

Figure 4: Disjunctive polyhedron obtained during the computation of the disturbance invariant set.

$$\textit{Let } \pi_1 \preceq \pi_2 \textit{ iff } (\forall q \in Q) \bigcup_{P_i \in \pi_1(q)} P_i \subseteq \bigcup_{R_j \in \pi_2(q)} R_j . \tag{4}$$

Once again, it is possible to define abstract domain operators. In particular, the join operation can avoid computing convex hulls in favor of simple set theoretic union of the individual polyhedra. Further details on the construction of disjunctive polyhedral domains are provided elsewhere [3].

While disjunctive polyhedra are good representations for non-convex sets, their complexity remains a key drawback. For instance, deciding the inclusion in (4) is known to be co-NP hard. Furthermore, as the iteration progresses, the number of disjuncts can increase rapidly, making the computation impractical.

EXAMPLE 3. *Figure 4 shows the disjunctive polyhedra for an intermediate step in the fixed point computation. The computation took many hours, without yielding a fixed point. Even if a fixed point were obtained, the resulting controller can be prohibitively complex, defeating our goal of a real time implementation.*

Techniques for managing the size and the complexity of the polyhedra have been discussed by Bagnara et al. [3] and implemented in the Parma Polyhedra Library. However, our preliminary experiments conclude that a direct application of the disjunctive completion is unsuitable for computing disturbance invariant sets.

We now present a disjunctive domain based on a state space partitioning. The state space partitioning naturally controls the complexity of the domain operations, and allows us to synthesize controller implementations with bounded worst-case execution times.

4. PARTITIONED DISJUNCTIVE DOMAINS

In this section, we construct a disjunctive numerical domain using two basic steps: (a) partition the safe set S into finitely many subsets, and (b) use the partitions to define a disjunctive abstract domain. In theory, our domain works with any partition. We will define a partitioning scheme based on repeated precondition computations on the inequalities defining the unsafe set.

Let $S \subseteq X$ be the safety specification and $U : X \setminus S$. Our domain captures sets of the form $W : U \uplus (P_1 \cup P_2 \cup \cdots P_k)$ wherein U, the unsafe set is implicitly assumed to be part of every disturbance invariant, and the sets $P_1, \ldots, P_k \in \text{POLY}(S)$.

Next, the disjuncts $P_1, \ldots, P_k$ are considered in a restricted form. We consider partitions of S into subsets $\{C_1, \ldots, C_N\}$ such that (a) each C_i is a polyhedron represented by the linear assertion ψ_i, (b) $\bigcup_{i=1}^{N} C_i \equiv S$, (c) For all pairs i, j, $C_i \cap C_j$ is empty.

EXAMPLE 4 (RECTANGULAR PARTITIONING). *Suppose* S *is an hyper-rectangle* $[\ell_1, u_1] \times \cdots \times [\ell_n, u_n]$ *as in the DC-DC*

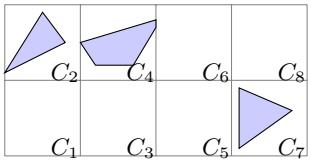


Figure 5: Partitioning the safe set into cells, depicted by the grid, and a partitioned disjunctive polyhedron with three disjuncts shown in blue.

converter example, then a simple partitioning scheme is based on subdividing each interval $[\ell_i, u_i]$ into p equally spaced sub-intervals $[\ell_i, y_{i,1}], [y_{i,1}, y_{i,2}], \ldots, [y_{i,p-1}, u_i]$. As a result, the safe set S is partitioned into hyper-rectangular cells.

DEFINITION 9 (PARTITIONED DISJUNCTIVE DOMAIN). *Let S be partitioned into cells $\mathcal{C} : \{C_1, \ldots, C_N\}$. A $\mathcal{C}$ disjunctive polyhedron is a tuple $\langle P_1, \ldots, P_N \rangle$, such that each P_i is a convex polyhedron and $P_i \subseteq C_i$.*

Finally, given a tiling $\mathcal{C}$, we consider the abstract domain of maps η from discrete modes $q \in Q$ to a $\mathcal{C}$-disjunctive polyhedron $\eta(q) : \langle P_1, \ldots, P_N \rangle$.

Figure 5 illustrates the idea behind a partitioned domain schematically. Let $\mathcal{P} : \langle P_1, \ldots, P_N \rangle$ and $\mathcal{R} : \langle R_1, \ldots, R_N \rangle$ be two $\mathcal{C}$-disjunctive polyhedra. We say that $\mathcal{P} \preceq \mathcal{R}$ iff $P_j \subseteq R_j$ for each $j \in [1, N]$. The ordering of maps η, π from modes to $\mathcal{C}$ disjunctive polyhedron is defined as

$$\eta \preceq \pi \text{ iff } \eta(q) \preceq \pi(q), \forall q \in Q$$

Partitioning places an upper bound on the size of the representation while allowing us to directly use the operations form the basic non-disjunctive polyhedral domain.

Given two $\mathcal{C}$-disjunctive polyhedra $\mathcal{P}$ and $\mathcal{R}$, we define $\mathcal{P} \oplus \mathcal{R} : \langle P_1 \oplus R_1, \ldots, P_N \oplus R_N \rangle$ where $\oplus \in \{\sqcap, \sqcup, \nabla\}$.

Disturbance Precondition: Let π be a map that associates each mode q to a $\mathcal{C}$-disjunctive polyhedron. The disturbance pre condition over π, $\widetilde{\mathsf{dpre}}(\pi)$ is computed as a new map $\hat{\pi}$, following Eq. (3)

$$\hat{\pi}(q) : \prod_{(q,q_j) \in E} \mathsf{Proj}_{\mathbf{d} \in D} \left(\mathsf{pre}(q, \pi(q_j), \mathbf{d}) \right) \tag{5}$$

Here, the pre operator is applied to each disjunct of $\pi(q_j)$. The result is temporarily a disjunctive polyhedron involving $\mathbf{x}, \mathbf{d}$. The projection operator existentially projects $\mathbf{d}$ from each disjunct and yields a $\mathcal{C}$-disjunctive polyhedron over $\mathbf{x}$.

Thus far, we have presented the basic ideas behind a partitioned disjunctive domain. However, the question of which partition to use remains to be resolved. We will now discuss a precondition-guided partitioning scheme that is inspired, in part, by the need to repeatedly compute the disturbance precondition computation.

Precondition-Guided Partitioning: Precondition-Guided Partitioning is a partitioning scheme based on repeatedly computing the disturbance precondition of unsafe region U w.r.t each control mode $q \in Q$. The main idea is to characterize sets that will reach the unsafe region U in finitely many steps under the repeated action of each control input $q \in Q$.

For each control mode $q \in Q$, we will partition S into a set of polyhedra $P_{q,1}, \ldots, P_{q,N}$ for some fixed $N > 0$. The overall partition has cells C_j, given by all nonempty intersections of the form $C_j : P_{q_1,j_1} \cap P_{q_2,j_2} \cap \cdots P_{q_M,j_M}$ $(M = |Q|)$.

For the sake of simplicity, let us assume that the unsafe region U is a single linear inequality. For each control mode $q \in Q$, we initialize the region $R_{q,0} : S$. We iteratively divide $R_{q,i}$ into $P_{q,i+1}$ and $R_{q,i+1}$:

$$P_{q,i+1} := R_{q,i} \cap \mathsf{dpre}^{(i)}(U, q),\ R_{q,i+1} := R_{q,i} \cap \overline{\mathsf{dpre}^{(i)}(U, q)})$$

where $\mathsf{dpre}^{(i)}(U, q)$ is the ith precondition of U which is also a linear inequality. We repeat this for $N - 1$ steps to yield the regions, $P_{q,1}, \ldots, P_{q,N-1}$. We terminate by setting $P_{q,N} := R_{q,N-1}$ to yield a partition of S through the regions $P_{q,1}, \ldots, P_{q,N}$. However, U can be expressed as a union of k linear inequalities $U : U_1 \cup \cdots \cup U_k$, repeating the process outlined above for each mode q_j and for each linear inequality U_i yields the desired partition. Figure 6 shows the partition for the DCDC converter example (left), and the post-fixed point computed by the abstract interpretation (right). The unshaded region is the control invariant.

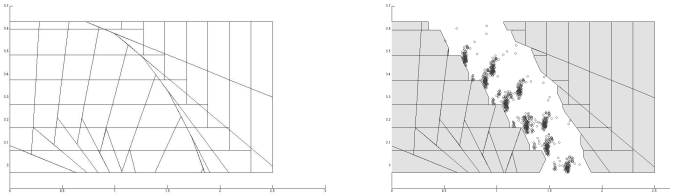


Figure 6: (left) final partition of the safe region for DC-DC converter and (right) disturbance invariant computed, shown by the shaded region. Black circles are states of the plant during a simulation

Normally, this process can potentially produce a large number of cells in our partition. However, we find that (a) most of these cells are empty, in practice, and (b) we can choose a subset $\hat{Q} \subseteq Q$ of modes, a subset of half-space U_j and control the limit N to also control our partition.

5. CONTROLLER SYNTHESIS

Thus far, we have provided abstract interpretation scheme to compute a disturbance invariant W. Now, we wish to compute a controller that implements a switching function to maintain the control invariant $C : X \setminus W$.

Recall the closed loop model from Figure 1. We note that at each time step, the control function inputs the current plant state $(q, \mathbf{x})$ and decides on a next step control input q'. Our goal is to ensure that q' is chosen so that $(q', \mathsf{next}(q, \mathbf{x}, \mathbf{d})) \in C$. Therefore, the controller code is a loop:

$$\mathsf{find}\ q'\ \mathsf{s.t}\ (q, q') \in E\ \mathsf{and}\ \forall\ \mathbf{d} \in D,\ (q', \mathsf{next}(q, \mathbf{x}, \mathbf{d})) \in C$$

For any $(q, \mathbf{x}) \in C$, a control invariant, such a q' is guaranteed to exist. Here, we seek this q' as a function of $(q, \mathbf{x})$. First, we extend the final disturbance invariant W obtained to the set

$$\widehat{W} : \left\{ (q, \mathbf{x}, q') \,\middle|\, (\exists \mathbf{d} \in D)\ (q', \mathbf{x}) \in \mathsf{pre}(q, W, \mathbf{d}) \right\}.$$

Therefore, q' *cannot* be chosen as the control input for $(q, \mathbf{x})$ iff $(q, \mathbf{x}, q') \in \widehat{W}$. In other words, checking that forall $\mathbf{d} \in D$, $(q', \mathsf{next}(q, \mathbf{x}, \mathbf{d})) \in C$ is equivalent to checking if $(q, \mathbf{x}, q') \notin \widehat{W}$.

Now, it remains to efficiently implement a data-structure that, given $(q, \mathbf{x})$ finds a $q' \in Q$ such that $(q, \mathbf{x}, q') \notin \widehat{W}$.

Data Structure for $\widehat{W}$ We assume that $\widehat{W}$ is stored as a disjunction $\widehat{W}_{q_1}, \ldots, \widehat{W}_{q_M}$, wherein for $q \in Q$,

$$\widehat{W}_q : \left\{ (q', \mathbf{x}) \mid (q, \mathbf{x}, q') \in \widehat{W} \right\}$$

In turn, each $\widehat{W}_q$ is stored as a map from each mode $q' \in Q$ to a partitioned disjunctive polyhedron that represents the possible values for $\mathbf{x}$. In other words, $\widehat{W}_q(q') : \left\{ \mathbf{x} \mid (q, \mathbf{x}, q') \in \widehat{W} \right\}$. Given a current state $(q, \mathbf{x})$, we find q' as follows:

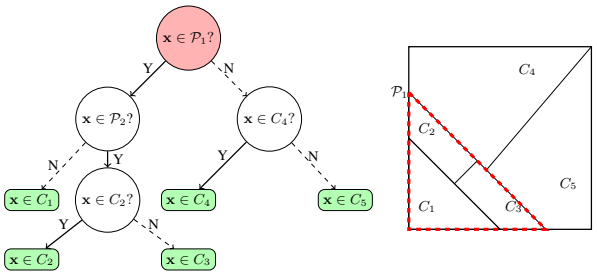


Figure 7: A BDD for deciding the cell ID of a node in the partition. $\mathcal{P}_1$ is the convex polyhedron formed by $\{C_1, C_2, C_3\}$, $\mathcal{P}_2$ is the convex polyhedron formed by $\{C_2, C_3\}$.

1. Select the cell C_j in the partition that contains $\mathbf{x}$.
2. Iterate through each mode $q' \in Q$, such that $(q, q') \in E$, we check if $\mathbf{x} \in \widehat{W}_q(q')$.
   (a) Since $\widehat{W}_q(q')$ is a partitioned polyhedron, there is precisely one polyhedron P_j included in C_j.
   (b) Check if $\mathbf{x} \in P_j$.

Therefore, given a partitioning $\mathcal{C} : C_1, \ldots, C_N$, and a point $\mathbf{x}$, a core primitive is to find a cell $C_j \in \mathcal{C}$, that contains $\mathbf{x}$. One solution is to run through the cells in order and test for inclusion. However, this is quite expensive given that N can often be a large number.

Our approach to find the cell is to represent the partition succinctly as a binary decision diagram (BDD) so that the cell C_j that $\mathbf{x}$ belongs to in the partition can be identified by checking querying the inclusion of $\mathbf{x}$ in a $O(\log N)$ polyhedra as opposed to $O(N)$ queries. We wish to find BDDs with bounds on their depth to enable optimal search for the cell in the partition.

Note that each cell in the precondition-guided partition is constructed as an intersection of polyhedra $\bigcap P_{q_i,j}$. In fact, the process of constructing a precondition-guided partition also yields polyhedra $R_{q_i,j}$ that "separates" the polyhedra $P_{q_i,1}, \ldots, P_{q_i,j}$ from the rest $P_{q_i,j+1}, \ldots, P_{q_i,m}$. These properties are exploited to construct a series of balanced decision trees that given $\mathbf{x}$ finds the polyhedra $P_{q_i,j}$ whose intersections form the cell C_j that contains $\mathbf{x}$.

5.1 BDD Construction

Here we are given a state $\mathbf{x} \in S$ and asked to find a cell C_j that it belongs to. The BDD we seek has the structure depicted in Figure 7. It is a rooted binary tree with *terminal nodes* (or leaves) and internal nodes. Each internal node n has three associated properties: (a) polyhedron φ_n, (b) left node $\mathsf{left}(n)$, and (c) $\mathsf{right}(n)$, the right node. Given a state $\mathbf{x}$, the ID of the cell C_j that contains $\mathbf{x}$ is given by a simple procedure that walks the BDD: Starting from root, at each step, we consider an internal node V and if $\mathbf{x} \in V$.polyhedron we go to node V.left and otherwise we go to node V.right. By carrying this procedure iteratively, we finally reach a terminal node that yields the requested cell. An illustrative example of a BDD for a partition is shown in Figure 7.

BDD construction In general, our partitioning scheme computes sets of polyhedra: $\mathcal{P}_i : \{P_{i1}, \ldots, P_{iN}\}$, for $i \in [1, M]$. Specifically, $\mathcal{P}_i$ represents subdivision for a mode $q_i \in Q$ for the precondition guided scheme. The fallowing properties easily obtainable from definitions of C_j, $R_{i,j}$ and $P_{i,j}$:

(A) Each cell is a nonempty intersection $C_j : \bigcap_{P_{i,j_i} \in \mathcal{P}_i} P_{i,j_i}$ of elements from the sets $\mathcal{P}_i$, $i \in [1, M]$.

(B) For each $j \in [1, N]$ the polyhedra $P_{i1}, \ldots, P_{ij} \in \mathcal{P}_i$ are separated from $P_{i,j+1}, \ldots, P_{iN} \in \mathcal{P}_i$ through a polyhedron $R_{i,j}$. I.e, $P_{i,k} \cap R_{i,j} = \emptyset$ for $k \leq j$, while $P_{i,k} \subseteq R_{i,j}$ for $k > j$.

Therefore, given a state $\mathbf{x}$, the corresponding cell C_k is identified by identifying a single polyhedron $P_{i(\mathbf{x})} \in \mathcal{P}_i$ for each i, so that $C_k : \bigcap_{i=1}^{M} P_{i(\mathbf{x})}$.

For each $\mathcal{P}_i$, the single polyhedron containing $\mathbf{x}$ is identified in $\log(|\mathcal{P}_i|) = \log(N)$ steps by a binary search process that is encoded in a BDD. At the root, first we find a j s.t. $R_{i,j}$ contains half of $P_{i,k}$s. Then we check if $\mathbf{x} \in R_{i,j}$: if yes, we know that $\mathbf{x} \in P_{i,k}$ where $k > j$. Otherwise, we know that $\mathbf{x} \in P_{i,k}$ where $k \leq j$. In each case, we narrow down the number of possibilities by half. The full search tree can be succinctly represented by a BDD with N nodes and of depth at most $\log_2(N)$.

Overall, we construct M BDDs, one for each of the sets in $\mathcal{P}_1, \ldots, \mathcal{P}_M$ with each BDD providing an element of $P_{i(\mathbf{x})} \in \mathcal{P}_i$ that contains the given point $\mathbf{x}$. Finally, a hash table can be used to lookup the cell ID, given the indices $i(\mathbf{x})$ of the polyhedra in each $\mathcal{P}_i$, that intersect to form the cell.

Using these ideas, the cell ID for the precondition guided partitioning scheme can be discovered using at most $\sum_{i=1}^{M} \log(|\mathcal{P}_i|)$ polyhedral membership tests, or $|Q| \log(N)$ where N is an upper bound on $|\mathcal{P}_i|$. Note that the number of cells is $O(|Q|^N)$ in the worst case.

6. EVALUATION

Implementation Figure 8 presents a block diagram description of our implementation of the ideas presented thus far. The input is a simple text-based description of the plant Ψ and the safe set S. Our implementation currently takes the edge set $E = Q \times Q$, assuming effectively that the controller can switch between any pair of modes. The fixed-point computation for the disturbance invariant first partitions the safe set S and builds the abstract domain. Currently, we support two kinds of partitioning, namely, grid-based and precondition-guided partitioning, as described in Section 4.

After partitioning, fixed point engine implements the partitioned disjunctive abstract domain from section 4. Our implementation uses the exact polyhedral manipulation primitives implemented in the Parma Polyhedra Library [5]. To control the complexity of our implementation, we use octagons instead of polyhedra for each of the cells in the partition for systems with 3 or fewer dimensions [25], and boxes inside each cell for higher dimensions. The application of the widening is delayed for 20 iterations to enhance precision. The simplified structure of the plant model allows us to carry out the projection of $\mathbf{d} \in D$ efficiently by assuming that the set D is convex and bounded. After finding a disturbance invariant, the BDD generator constructs a BDD for finding a partition containing a given state using a modification of the algorithm discussed in section 5. The modification ensures the construction of a single BDD for the entire partition rather than a family of BDDs. It will be presented in our extended version.

The output of the BDD generator is fed to the code generator, which generates the code for the controller in the destination platform. Currently, our implementation generates MATLAB™code that executes inside a simulator environment also implemented in MATLAB™.

6.1 Results

First, we briefly explain a set of benchmarks over which the evaluation has been carried out. Each benchmark yields multiple controller synthesis instances for different variations on the plant model, disturbance set sizes and number of control inputs to be used by our controller. Further details about the benchmarks and experimental results over variations of these benchmarks are available in

Table 2: Summary of results of running our implementation on the benchmark suite. **Legend**: n: # state variables, $|Q|$: # modes, Δ: discretization time. All timings are in seconds. All the experiments were carried out on a Macbook pro laptop with 2.9 GHz Intel Core i7 processor, 8GB of RAM, running MACOSX 10.9.

	Problem			Precondition-guided Partitioning			Grid-based Partitioning	
Benchmark	n	|Q|	Δ	# Cells	Time	BDD Depth	# Cells	Time
DCDC (Single Output)	2	2	2×10^{-5}	98	10.9	9	96	6.7
	2	2	2×10^{-5}	95	11.0	9	96	5.7
DC Motor	2	3	10^{-4}	138	1.3	6	275	1.5
	2	3	10^{-4}	791	14.1	9	275	4.0
Inverted Pendulum	2	3	0.05	44	0.9	6	131	1.9
	2	3	0.05	137	0.8	3	617	2.6
DCDC (Double Output)	3	3	10^{-5}	974	402.3	13	10659	5368.6*
	3	3	5×10^{-5}	4950	3022.7	15	10659	6526.4*
Helicopter	2	3	0.1	186	1.2	4	243	1.4
	4	9	0.1	636	136.2	10	1776	261.2
Propofol Delivery	4	2	60	10423	347.0	12	10659	13494.0
	6	2	120	10680	2585.6	12	-	-**
Multi-level Converter	3	6	0.002	545	8.3	3	734	5.4
	4	5	0.002	555	40.7	9	1740	115.8
Room Heating	4	5	300	85	57.3	8	1955	63.7
	5	16	480	808	509.8	11	1294	219.3

(*) The gridding method could not find a disturbance invariant (**) The procedure timed-out ($> 10hrs$)

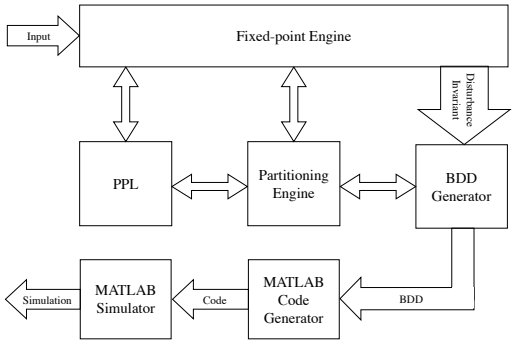


Figure 8: Block diagram of the synthesizing procedure

the extended version.

6.1.1 Benchmarks

We briefly present the benchmarks used in our evaluation.

DCDC Boost Converter We consider two sets of DC-DC boost converter examples, based on Example 1, taken from Senesky et al. [32]. As explained earlier, the control objective is to maintain the current and voltage(s) of a circuit in a proper range by toggling the switches in a circuit. The first benchmark set has a single output voltage, whereas the second set has two output voltages. In single-output DCDC converter example, we consider different instances with varying disturbance ranges D, and for the second set we consider different instances with varying discretization time steps.

DC Motor In this example (taken from the PESSOA tool benchmarks), we want to maintain the velocity of a DC Motor by changing the source voltage [24]. We discretize the continuous input voltage into finitely many discrete voltages. Different instances are generated by varying the range of the control input voltages.

Inverted Pendulum This benchmark is a linearized model of an inverted pendulum, once again taken from the PESSOA-tool benchmark suite [35]. The goal is to keep an inverted pendulum upright by providing forces in appropriate directions at appropriate times. Different instances of this benchmark consider different values for the safe set S.

Propofol Delivery This benchmark was proposed recently by Maidens et al. [22], and detailed by Gan et al. [14] the patient is modeled as a 4 variables linear delay differential equation that is approximated using a standard Padé approximation. The input is the infusion rate of the Propofol and the goal is concentration of the Propofol in the effect chamber of the pharmacokinetic model. Different instances for this benchmark were derived by changing the order of the Padé approximation.

Helicopter The helicopter example (taken from [10]), is a linear model of a helicopter, with the control objective placing bounds on the helicopter's x, y position and velocities. The control inputs are the roll and the pitch. Our benchmark uses two instances that vary in the number of dimensions of the problem.

Multi-level Converter This example is also a model of an electrical circuit, wherein we want to regulate voltages in the circuit under varying loads [12, 13]. We consider different instances based on varying number of discrete modes.

Room Heating In this example, inspired by [11], we consider the problem of maintaining temperature inside a building with many rooms within a given range. The temperature of each room changes according to Newton's cooling law. We have limited number of portable heaters and we want to keep temperature of each room in an acceptable bound by moving the heaters and turning them on and off. We consider different instances that vary in the number of heaters (control modes) and the number of rooms (state variables).

6.1.2 Evaluation

Table 2 summarizes the results of our implementation running on the benchmark suite, comparing the two approaches for the partitioning: precondition-guided partitions against grid-based partitions. The performance on all instances of all benchmarks is presented in our extended version. Here, we present two representative instances for each benchmark to provide a succinct summary.

The results clearly demonstrate the ability of our method to successfully compute disturbance invariants using the disjunctive domain and extract controller code from these. As expected, the performance varies depending on the number of state variables, modes and control inputs. Overall, however, our implementation can handle all

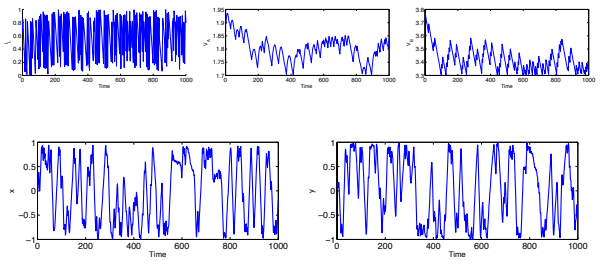


Figure 10: Simulation traces (Top) the double output DC-DC converter benchmark (Bottom) for the Helicopter benchmark.

benchmark instances in under an hour. The grid-based partitioning scheme is seen to perform well for smaller systems but degrades fast. Whereas the precondition-based scheme works well on almost all the benchmarks, even though the number of grid cells remains comparable for both approaches. This demonstrates the usefulness of precondition-guided partitioning as a more natural partitioning of the state-space for our computations.

Also as demonstrated in Figure 9, the precondition-guided partitioning method gives a bigger control invariant (and more similar to the maximum control invariant) which is another witness to the preciseness of this method compared to grid-based method.

The synthesis of the controller itself is a tiny fraction of the overall computation time, which is dominated by the fixed-point computation. In most cases, control synthesis took less than 1% of the overall computation time.

In most cases, the depth of the BDDs are small as demonstrated in Table 2. The depth of the BDD dictates the number of nested conditional statements in the control code. Therefore, the resulting controller implementation is more amenable to a real-time control. Example simulation traces for the double output DC-DC converter and the helicopter example are shown in Figure 10 .

Comparison with PESSOA A direct comparison of our method to the PESSOA toolbox [24] is not possible, since PESSOA has different semantics and supports a richer specification language. On the other hand, disturbances are not supported by PESSOA. Nevertheless, some of the benchmarks can be solved by both approaches for similar (but not identical) control specifications.

For the DC-Motor, the inverted pendulum and the Propofol delivery benchmarks, our approach and PESSOA produce results with comparable timings. Note that we set disturbance inputs to 0 to enable this comparison. For the helicopter benchmark, PESSOA failed to find a controller when the space quantization step was 0.04. Reducing this to 0.03, however, cause a out-of-memory error.

Our future work will consider a tighter integration using a tool such as PESSOA to handle liveness properties while computing a maximally permissive safety controller using fixed point techniques over polyhedra, as presented here.

7. CONCLUSION

To conclude, we present a disjunctive polyhedral abstract interpretation for computing disturbance invariant sets for discrete time controllers. Our future work will extend these ideas beyond safety to consider liveness properties that are also important in controller synthesis. There are many ways to improve the proposed abstract domain. We are investigating dynamic state-partitioning approaches during the course of fixed point computation. Consideration of performance metrics and synthesizing controllers to minimize costs is another fruitful line of future work in this area.

Acknowledgments: This work was supported by the US National Science Foundation (NSF) under award number CNS-0953941. All opinions are those of the authors and not necessarily of the NSF.

8. REFERENCES

[1] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88(7):1011–1025, July 2000.

[2] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems II, LNCS 999*, pages 1–20. Springer, 1995.

[3] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. In *Static Analysis Symposium (SAS)*, volume 2694 of *LNCS*, pages 337–354. Springer, 2003.

[4] R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. In *VMCAI*, volume 2937 of *LNCS*, pages 135–148. Springer-Verlag, 2004.

[5] R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill. Possibly not closed convex polyhedra and the parma polyhedra library. In *SAS*, pages 213–229, 2002.

[6] T. A. Beyene, S. Chaudhuri, C. Popeea, and A. Rybalchenko. A constraint-based approach to solving games on infinite graphs. In *POPL*, pages 221–234. ACM, 2014.

[7] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. ISOP'76*, pages 106–130, 1976.

[8] P. Cousot and R. Cousot. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[9] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among the variables of a program. In *POPL'78*, pages 84–97, Jan. 1978.

[10] J. Ding, E. Li, H. Huang, and C. J. Tomlin. Reachability based synthesis of feedback policies for motion planning under bounded disturbances. In *ICRA*, pages 2160–2165, 2011.

[11] A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In *HSCC*, pages 326–341. Springer, 2004.

[12] G. Feld, L. Fribourg, D. Labrousse, S. Lefebvre, B. Revol, and R. Soulat. Control of multilevel power converters using formal methods. Research Report LSV-12-14, Laboratoire Spécification et Vérification, ENS Cachan, France, June 2012. 14 pages.

[13] G. Feld, L. Fribourg, D. Labrousse, B. Revol, and R. Soulat. Correct-by-design control of 5-level and 7-level power converters. Technical Report LSV-12-25, LSV, ENS Cachan, France, 2012.

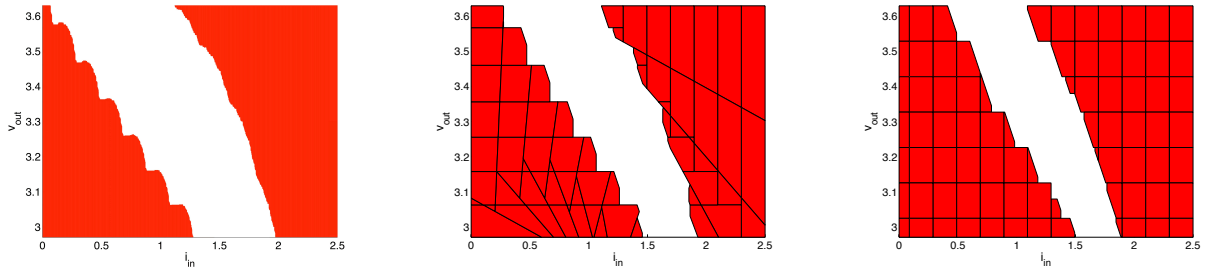
[14] V. Gan, G. A. Dumont, and I. M. Mitchell. Benchmark problem: A PK/PD model and safety constraints for anesthesia delivery, 2014. Benchmark presented at the ARCH 2014 workshop.

[15] E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games: A guide to current research*. Springer, 2002.

[16] C. L. Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250 – 262, 2010.

[17] N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2):157–185, 1997.

[18] M. Herceg, M. Kvasnica, C. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zurich, Switzerland, July



(a) Maximum Control Invariant (b) Computed CI (Precondition-based partitioning) (c) Computed CI (Grid-based partitioning)

Figure 9: Accuracy of control invariants found by partitioning methods with the same number of cells

17–19 2013.

[19] B. Jeannet, N. Halbwachs, and P. Raymond. Dynamic partitioning in analyses of numerical properties. In *SAS*, volume 1694 of *LNCS*, pages 39–50, 1999.

[20] T. Le Gall, B. Jeannet, and H. Marchand. Supervisory control of infinite symbolic systems using abstract interpretation. In *CDC'05*, pages 31–35, December 2005.

[21] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35:349–370, 1999.

[22] J. N. Maidens, S. Kaynama, I. Mitchell, M. M. Oishi, and G. A. Dumont. Lagrangian methods for approximating the viability kernel in high-dimensional systems. *Automatica*, 49(7):2017 – 2029, 2013.

[23] L. Mauborgne and X. Rival. Trace partitioning in abstract interpretation based static analyzers. In M. Sagiv, editor, *ESOP'05*, volume 3444 of *LNCS*, pages 5–20. Springer-Verlag, 2005.

[24] J. Mazo, Manuel, A. Davitian, and P. Tabuada. PESSOA: a tool for embedded controller synthesis. In *CAV*, volume 6174 of *LNCS*, pages 566–569. Springer, 2010.

[25] A. Miné. The octagon abstract domain. In *AST 2001 in WCRE 2001*, IEEE, pages 310–319, October 2001.

[26] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In *HSCC*, volume 1790, pages 310–323. Springer, 2000.

[27] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. springer, 1999.

[28] C. Popeea and W.-N. Chin. Inferring disjunctive postconditions. In *ASIAN*, volume 4435 of *LNCS*, pages 331–345. Springer, 2006.

[29] P. Roux, R. Jobredeaux, P.-L. Garoche, and E. Feron. A generic ellipsoid abstract domain for linear time invariant systems. In *HSCC*, pages 105–114. ACM, 2012.

[30] S. Sankaranarayanan, F. Ivančić, I. Shlyakhter, and A. Gupta. Static analysis in disjunctive numerical domains. In *SAS*, volume 4134, pages 3–17. Springer, 2006.

[31] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *VMCAI'05*, volume 3385 of *LNCS*, January 2005.

[32] M. Senesky, G. Eirea, and T.-J. Koo. Hybrid modelling and control of power electronics. In *HSCC*, pages 450–465, 2003.

[33] A. Simon, A. King, and J. M. Howe. Two variables per linear inequality as an abstract domain. In *LOPSTR*, volume 2664 of *LNCS*, pages 71–89. Springer, 2003.

[34] A. Taly, S. Gulwani, and A. Tiwari. Synthesizing switching logic using constraint solving. *STTT Journal*, 13(6):519–535, 2011.

[35] UCLA CyPhy Laboratory. PESSOA: Software toolbox for the synthesis of correct-by-design embedded control software.

[36] R. Vidal, S. Schaffert, J. Lygeros, and S. Sastry. Controlled invariance of discrete time systems. In *HSCC*, pages 437–450. Springer Verlag, 2000.

[37] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. CDC*, pages 4607–4612, December 1997.

APPENDIX

A. BENCHMARKS AND RESULTS (DETAILS)

In this section, we describe each benchmark and in details and compare the results obtained from two different partitioning method. For the precondition-guided partitioning scheme, we indicate which modes where chosen for partitioning operation and for the grid-based scheme we indicate the gridding step size σ for each example.

A.1 Single output DCDC boost converter

The single output DCDC boost converter has two continuous variables (i_L and v_o) and has two modes ($q1$ and $q2$). The differential equations correspond to these modes are

$$\begin{bmatrix} \dot{i_L} \\ \dot{v_o} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} i_L \\ v_o \end{bmatrix} + \begin{bmatrix} \frac{v_{in}}{L} \\ 0 \end{bmatrix}$$

,

$$\begin{bmatrix} \dot{i_L} \\ \dot{v_o} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} i_L \\ v_o \end{bmatrix} + \begin{bmatrix} \frac{v_{in}}{L} \\ 0 \end{bmatrix}$$

respectively. The constant values and other parameters are

v_{in}	1.5
L	150×10^{-6}
C	110×10^{-6}
R	6
S	$Q \times [0, 2.5] \times [2.97, 3.63]$
Δ	20×10^{-6}

We run the experiment with different values for size of set D. The disturbance is small because the error in electrical circuits are small as well as the time step.

Using precondition-guided partitioning scheme, we were able to solve first four problems, but failed for the fifth. Then we decreased the Δ to 10×10^{-6} and a disturbance invariant was found. The results can be found in Table 3.

On the other hand, using grid-based scheme we solved the first four problems by $\sigma = 0.2 \times 0.1$. However for the fifth problem, this method also failed. Then by increasing the number of segments ($\sigma = 0.1 \times 0.05$) even with $\Delta = 20 \times 10^{-6}$ a non-empty disturbance invariant was found.

The failure in the last case by precondition-guided scheme suggests that we can be more precise by decreasing the time-steps only for partitioning purpose and we can use the original time-step for fixed-point computations and this fact should be considered for future works.

A.2 DC Motor

The DC Motor example has two continuous variables (ω and i) and the differential equations for the system are

$$\begin{bmatrix} \dot{\omega} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} -\frac{B}{J} & \frac{k}{J} \\ -\frac{k}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \omega \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u$$

The input u can be $-u_{max}$, 0 and u_{max}. The constants and other parameters are

J	250×10^{-6}
B	100×10^{-6}
k	0.05
L	1500×10^{-6}
R	0.5
S	$Q \times [-19.5, 20.5] \times [-0.7, 0.7]$
Δ	100×10^{-6}
D	$[-0.003, 0.003]^2$

In each case of the benchmark we change the value of u_{max} and the smaller the u_{max} is the less controllable will be the environment. The results are shown in Table 4. Both methods failed for $u_{max} = 1$. Since the grid-based partitioning method failed even with 21880 cells, it is probably the case that the safety property can not satisfied.

A.3 Inverted Pendulum

This example is the linearized version of the inverted pendulum example [35]. The continuous variables are θ and ω. We assume the value of θ is small and $sin(\theta) = \theta$ and $cos(\theta) = 1$. The result is the following differential equations.

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & -\frac{h}{ml^2} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml} \end{bmatrix} u$$

The input u can have values $-u_{max}$, 0 and u_{max}. The constants and other parameters are

g	9.8
l	0.5
m	0.5
h	2
u_{max}	6
Δ	0.05
D	$[-0.02, 0.02]^2$

The safe region is $Q \times [-r, r] \times [-1, 1]$. In each case of the benchmark, we change the safe region and make the r bigger in each case.

For grid-based scheme we set the σ to 0.01×0.1 and as the results shown in Table 5, both methods work well in this benchmark.

A.4 Double output DCDC boost converter

In this example, there are three continuous variables (i_L, v_A and v_B) and three modes $q1$, $q2$ and $q3$ and the differential equations for these modes are

$$\begin{bmatrix} \dot{i_L} \\ \dot{v_A} \\ \dot{v_B} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{1}{R_A C_A} & 0 \\ 0 & 0 & -\frac{1}{R_B C_B} \end{bmatrix} \begin{bmatrix} i_L \\ v_A \\ v_B \end{bmatrix} + \begin{bmatrix} \frac{v_{in}}{L} \\ 0 \\ 0 \end{bmatrix}$$

,

$$\begin{bmatrix} \dot{i_L} \\ \dot{v_A} \\ \dot{v_B} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{L} & 0 \\ \frac{1}{C_A} & -\frac{1}{R_A C_A} & 0 \\ 0 & 0 & -\frac{1}{R_B C_B} \end{bmatrix} \begin{bmatrix} i_L \\ v_A \\ v_B \end{bmatrix} + \begin{bmatrix} \frac{v_{in}}{L} \\ 0 \\ 0 \end{bmatrix}$$

,

$$\begin{bmatrix} \dot{i_L} \\ \dot{v_A} \\ \dot{v_B} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\frac{1}{L} \\ 0 & -\frac{1}{R_A C_A} & 0 \\ \frac{1}{C_B} & 0 & -\frac{1}{R_B C_B} \end{bmatrix} \begin{bmatrix} i_L \\ v_A \\ v_B \end{bmatrix} + \begin{bmatrix} \frac{v_{in}}{L} \\ 0 \\ 0 \end{bmatrix}$$

respectively. The constants and other parameters are

L	75×10^{-6}
v_{in}	1.5
R_A	6.25
R_B	34.1
C_A	800×10^{-6}
C_B	146.6×10^{-6}
S	$Q \times [0, 1] \times [1.7, 2] \times [3.3, 4]$
D	$[-0.0005, 0.0005]^3$

In this benchmark, each case has a different value for Δ. The grid-base scheme failed with $\sigma = 0.03 \times 0.02 \times 0.04$. The results (Table 6) suggest that the precondition-guided partitioning scheme is much more accurate for these problems.

A.5 Helicopter

The simplified helicopter example is the problem of keeping the helicopter near the origin. We consider two cases for this benchmark. The first one is in 1 dimension space. Continuous variables are x and v_x. The modes can change according to roll command. The differential equations are

$$\begin{bmatrix} \dot{x} \\ \dot{v_x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v_x \end{bmatrix} + \begin{bmatrix} 0 \\ g\sin(\phi) \end{bmatrix}$$

Input ϕ can have 3 different values (-10, 0 and 10 degree). The constant and other parameters are

g	9.8
S	$Q \times [-1, 1]^2$
Δ	0.2
D	$[-0.03, 0.03] \times [-0.04, 0.04]$

We also try the same problem for 2 dimensions space. More precisely, the variables are x, v_x, y and v_y and differential equations are

$$\begin{bmatrix} \dot{x} \\ \dot{v_x} \\ \dot{y} \\ \dot{v_y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} + \begin{bmatrix} 0 \\ g\sin(\phi) \\ 0 \\ g\sin(\psi) \end{bmatrix}$$

Inputs are $(\phi, \psi) \in \{-10, 0, 10\}^2$. The constants and other parameters are

g	9.8
S	$Q \times [-1, 1]^4$
Δ	0.2
D	$([-0.03, 0.03] \times [-0.04, 0.04])^2$

As results demonstrate in Table 7, for the second problem grid-based scheme with $\sigma = 0.4^4$ was not precise enough for finding a disturbance invariant.

Also , since the number of modes are more than necessary for the second example, we chose only 4 of these modes for partitioning. These for modes are

$$(\phi, \psi) \in \{(-10, 0), (10, 0), (0, -10), (0, 10)\}$$

A.6 Propofol Delivery

In this benchmark, our patient is modeled by four continuous variables (c_p, c_1, c_2 and c_e). The differential equations are

$$\begin{bmatrix} \dot{c_p} \\ \dot{c_1} \\ \dot{c_2} \\ \dot{c_e} \end{bmatrix} = \begin{bmatrix} -(k_{10} + k_{12} + k_{13}) & k_{12} & k_{13} & 0 \\ k_{21} & -k_{21} & 0 & 0 \\ k_{31} & 0 & -k_{31} & 0 \\ k_d & 0 & 0 & -kd \end{bmatrix} \begin{bmatrix} c_p \\ c_1 \\ c_2 \\ c_e \end{bmatrix}$$

$$+ \begin{bmatrix} \frac{1}{V_1} \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t - T_d)$$

The input u here can have two values 0, and 6000. T_d is the time delay. The constants and other parameters are

weight	35
k_{10}	$0.1527 \times \text{weight}^{-0.3}$
k_{12}	0.114
k_{13}	0.0419
k_{21}	0.055
k_{31}	0.0033
k_d	40
V_1	$458 \times$ weight
S	$S_4 = Q \times [1, 6] \times [0, 10] \times [0, 10] \times [1, 8]$
Δ	120
D	$[0.02]^4$

In the first case, we consider T_d to be zero. Then we add extra continuous variables to model a memory for estimating the delay for $T_d = 30$. More precisely in the second example we add 1 variable for modeling the memory using Pade-approximation. The set D also becomes $[0.02]^5$ and the safe region becomes $S_4 \times [0, 1]$.

Similarly in the third example we add 2 variables ($D = [0.02]^6$, $S = S_4 \times [-1, 1] \times [-1, 2]$).

Notice that the for the safe region there is no restriction on the new added variables, however, to make the safe region bounded, we add these restrictions.

Comparing to two scheme for partitioning (Table 8), the grid-based scheme takes much more time and for higher dimensions the procedure does not terminate in 10 hours.

A.7 Level converter

In this benchmark, we consider two different example of multi-level converters (4-level and 5-level converters). In these examples, there are some voltage variables ($v_1 ... v_{n-1}$) and one current variable i. Also there are n switches. We refer to one switches state by vector $Sw = (s_1, s_2, ..., s_n)^T$, and each s_i can be 0 (off) or 1 (on).

In the first example, the differential equations are

$$\begin{bmatrix} \dot{v_1} \\ \dot{v_2} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{(s_1 - s_2)}{C} \\ 0 & 0 & \frac{(s_2 - s_3)}{C} \\ \frac{(s_2 - s_1)}{L} & \frac{(s_3 - s_2)}{L} & \frac{-R_{Load}}{L} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ s_1 \frac{v_{in}}{L} \end{bmatrix}$$

The constants and other parameters are

v_{in}	200
R_{Load}	5
C	0.1
L	0.0137
S	$Q \times [130, 135] \times [65, 70] \times [0, 40]$
Δ	0.002
D	$[-0.001, 0.001]^3$

There are 8 different modes, however, we ignore two of these modes ($Sw = (1, 0, 1)$, $Sw = (1, 1, 0)$) and we try to synthesize the controller without them.

The second example have a little bit different parameters. The differential equations are

$$\begin{bmatrix} \dot{v_1} \\ \dot{v_2} \\ \dot{v_3} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} -\frac{1}{RC} & 0 & 0 & \frac{(s_1 - s_2)}{C} \\ 0 & -\frac{1}{RC} & 0 & \frac{(s_2 - s_3)}{C} \\ 0 & 0 & -\frac{1}{RC} & \frac{(s_3 - s_4)}{C} \\ \frac{(s_2 - s_1)}{L} & \frac{(s_3 - s_2)}{L} & \frac{(s_4 - s_3)}{L} & \frac{-R_{Load}}{L} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ i \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ s_1 \frac{v_{in}}{L} \end{bmatrix}$$

For this example the constants and parameters are

v_{in}	200
R_{Load}	50
R	20000
C	0.0012
L	0.2
S	$Q \times [145, 150] \times [95, 100] \times [45, 50] \times [-3, 4]$
Δ	0.002
D	$[-0.001, 0.001]^4$

There are 16 different modes, however here we only consider 5 of them for synthesizing the controller. These modes are $Sw = (0,0,0,0)^T$, $Sw = (0,0,0,1)^T$, $Sw = (0,0,1,1)^T$, $Sw = (0,1,1,1)^T$, $Sw = (1,1,1,1)^T$

The precondition-guided partitioning scheme uses 2 of 6 modes ($Sw = (0,0,1)^T$ and $Sw = (0,1,1)^T$) for partitioning and in the last case, it uses 3 of 5 modes for partitioning ($Sw = (0,0,0,1)^T$, $Sw = (0,0,1,1)^T$ and $Sw = (0,1,1,1)^T$).

Both approach work almost good (Table 9) on this benchmark, however, with the same number of cells, the precondition-guided partitioning method works more precise.

A.8 Room Heating

This benchmark contains higher dimensions examples. In these set of examples, there are some rooms in a building and we want to keep their temperature with in some acceptable bounds. Each wall of a room is surrounded with either by another room or the outside environment. Walls are usually insulated, however, they are not perfect. The temperature of each rooms changes by Newton's cooling law. More precisely, having two areas with temperature x_1 and x_2 a mutually surface, then $\dot{x_1} = k(x_2 - x_1)$, where k is the cooling constant for that surface. The cooling constant for walls is k_w and for floors is k_f. We assume that outside temperature (t_a) remains constant (As well as earth temperature (t_e) which is adjacent to the basement room). The problem in room heating is as follows. We have some limited number of heaters and we want to keep the rooms warm, by these portable heaters. Each heater provides h Fahrenheit per hour rise in the temperature of the room. The controller can move the heaters between the rooms and turn them on or off. We want to keep the temperature of basement room between 60 and 75. The temperature of attic between 55 and 65 and the other rooms between 60 and 70.

In the first example, there are 4 rooms and 1 heaters. The differential equations are

$$\begin{bmatrix} \dot{t_1} \\ \dot{t_2} \\ \dot{t_3} \\ \dot{t_4} \end{bmatrix} = \begin{bmatrix} -k_w - k_f & k_f & 0 & 0 \\ k_f & -k_w - 2k_f & k_f & 0 \\ 0 & k_f & -k_w - 2k_f & k_f \\ 0 & 0 & k_f & -k_w - k_f \end{bmatrix}$$

$$\begin{bmatrix} k_w t_e \\ k_w t_a \\ k_w t_a \\ k_w t_a \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} h$$

s_i indicates if there is one heater in room i (1) or not (0). Because we have one heater, then $\sum_{i=1}^{4} s_i \in \{0, 1\}$. Therefore, there are 5 different modes. A vector $H = (s_1, s_2, s_3, s_4)^T$ defines a mode. The constants and parameters are

t_a	48
t_e	40
k_w	0.5
k_f	0.7
h	60
Δ	300
D	$[-0.1, 0.1]^4$

For the second example, there are 5 rooms and 2 heaters. The differential equation is similarly

$$\begin{bmatrix} \dot{t_1} \\ \dot{t_2} \\ \dot{t_3} \\ \dot{t_4} \\ \dot{t_5} \end{bmatrix} =$$

$$\begin{bmatrix} -k_w - k_f & k_f & 0 & 0 & 0 \\ k_f & -k_w - 2k_f & k_f & 0 & 0 \\ 0 & k_f & -k_w - 2k_f & k_f & 0 \\ 0 & 0 & k_f & -k_w - 2k_f & k_f \\ 0 & 0 & 0 & k_f & -k_w - k_f \end{bmatrix}$$

$$\begin{bmatrix} k_w t_e \\ k_w t_a \\ k_w t_a \\ k_w t_a \\ k_w t_a \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix} h$$

Since we have two heaters, then $\sum_{i=1}^{5} s_i \in \{0, 1, 2\}$. Therefore, there are 22 different modes. However, we choose only 10 of them for synthesizing the controller. These modes are $H = (0,0,0,0,0)$, $H = (1,0,1,0,0)$, $H = (1,0,0,1,0)$, $H = (0,1,0,0,1)$, $H = (0,0,1,0,1)$, $H = (1,0,0,0,0)$, $H = (0,1,0,0,0)$, $H = (0,0,1,0,0)$, $H = (0,0,0,1,0)$, $H = (0,0,0,0,1)$

All the constants and parameters are the same except for $\Delta = 480$ and $D = [-0.1, 0.1]^5$.

For the first problem, the precondition guided partitioning scheme uses two of the modes ($H = (1,0,0,0)$ and $H = (0,0,0,1)$) for partitioning and uses three modes ($H = (1,0,0,0)$, $H = (0,0,0,1)$ and $H = (1,0,0,0)$) for partitioning in the second problem. The results can be found in Table 10.

Table 3: Results for single output DCDC converter benchmark. **Legend**: D: disturbance set, #C : # cells, #NEC : # non-empty cells, #It : # iterations in fixed-point computation, T_{fp} : fixed-point computation time, Depth: depth of the BDD, #Ineq: number of inequalities need to be calculated for reaching a leaf of BDD (Worst-case), T_{BDD}: BDD construction time, Stat: status of the result, σ: the step size for gridding, NF: disturbance invariant not found

		Precondition-guided Partitioning								Grid-based Partitioning			
D	Δ	#C	#NEC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	σ	#C	T_{fp}	Stat
$0.001([-1,1]^2)$	20	98	72	137	10.9	9	42	ϵ	OK	0.2×0.1	96	6.8	OK
$0.002([-1,1]^2)$	20	99	74	197	11.7	9	43	ϵ	OK	0.2×0.1	96	6.1	OK
$0.003([-1,1]^2)$	20	97	73	112	9.6	9	43	ϵ	OK	0.2×0.1	96	7.3	OK
$0.004([-1,1]^2)$	20	95	74	121	11.1	9	43	ϵ	OK	0.2×0.1	96	5.7	OK
$0.005([-1,1]^2)$	20	94	94	63	8.8	10	48	ϵ	NF	0.2×0.1	96	9.4	NF
										0.1×0.05	294	20.2	OK
	10	275	215	111	37.7	11	56	0.29	OK	0.1×0.05	294	31.3	OK

Table 4: Results for DC Motor benchmark, **Legend**: See legend of Table 3

	Precondition-guided Partitioning								Gridding			
u_{max}	#C	#NPC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	σ	#C	T_{fp}	Stat
10	138	17	7	1.3	6	42	ϵ	OK	0.05×0.1	271	48.1	NF
									0.05×0.05	518	1.6	OK
5	253	18	4	2.1	6	38	ϵ	OK	0.05×0.05	518	1.9	OK
2	791	106	14	14.1	9	61	ϵ	OK	0.05×0.05	518	4.0	OK
1	6624	6624	174	1375.4	19	125	68.5	NF	0.005×0.005	21880	2299.7	NF

Table 5: Results for inverted pendulum benchmark, **Legend**: r : range of $\theta \in [-r, r]$ in safe set S, also see legend of Table 3

	Precondition-guided Partitioning								Gridding		
r	#C	#NPC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	#C	T_{fp}	Stat
0.1	44	23	10	0.9	6	35	ϵ	OK	131	1.9	OK
0.2	71	12	3	0.5	5	21	ϵ	OK	257	1.9	OK
0.3	94	23	22	1.3	6	33	ϵ	OK	365	10.7	OK
0.4	121	8	3	0.9	3	14	ϵ	OK	491	2.2	OK
0.5	137	8	3	0.9	3	14	ϵ	OK	617	2.6	OK

Table 6: Results for double output DCDC converter benchmark, **Legend**: See legend of Table 3

	Precondition-guided Partitioning								Gridding		
Δ	#C	#NPC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	#C	T_{fp}	Stat
5	4950	1436	124	3022.7	15	114	9.8	OK	10659	6526.4	NF
10	974	312	83	402.3	13	89	0.7	OK	10659	5368.6	NF
15	454	454	50	260.6	14	107	1.4	NF	10659	3192.5	NF

Table 7: Results for Helicopter benchmark, **Legend**: n: # state variables, $|Q|$: # modes, #MP: # modes for partitioning, also see legend of Table 3

		Precondition-guided Partitioning									Gridding			
n	$\|Q\|$	#MP	#C	#NEC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	σ	#C	T_{fp}	Stat
2	3	3	197	11	3	0.9	4	19	ϵ	OK	0.5^2	31	0.5	OK
4	9	5	636	137	4	136.2	10	107	0.3	OK	0.4^4	1308	797.0	NF
											0.35^4	1776	361.2	OK

Table 8: Results for Propofol delivery benchmark, **Legend**: TO: Timed out, see also legend of Table 3

	Precondition-guided Partitioning								Gridding			
n	#C	#NEC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	σ	#C	T_{fp}	Stat
4	10423	285	31	347.0	12	185	1.6	OK	1^4	5819	9560.3	NF
									0.88^4	10659	13494	OK
5	10429	311	32	864.6	12	214	3.8	OK	$0.88^4 \times 3$	10659	> 10 hours	TO
6	10680	365	33	2585.6	12	265	14.0	OK	$0.88^4 \times 3^2$	10659	> 10 hours	TO

Table 9: Results for Multi-level converter benchmark, **Legend**: See legend of Table 7

		Precondition-guided Partitioning									Gridding			
n	$\|Q\|$	#MP	#C	#NEC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	σ	#C	T_{fp}	Stat
3	8	2	545	10	3	8.3	3	18	ϵ	OK	0.5×0.5×8	734	5.4	OK
4	5	3	555	97	9	40.7	9	100	0.2	OK	1.3^4	886	163.2	NF
											1.2^4	1523	90.9	OK

Table 10: Results for room heating benchmark, **Legend**: See legend of Table 7

			Precondition-guided Partitioning									Gridding			
n	$\|Q\|$	Δ	#MP	#C	#NEC	#It	T_{fp}	Depth	#Ineq	T_{BDD}	Stat	σ	#C	T_{fp}	Stat
4	4	300	2	85	65	31	57.3	8	84	0.2	OK	3^4	761	182.1	NF
												2^4	1955	63.7	OK
5	10	480	5	808	214	4	509.8	11	126	1.4	OK	4^5	1294	219.2	OK