Counterexample-guided Refinement of Template Polyhedra*

Sergiy Bogomolov[1,2], Goran Frehse[3], Mirco Giacobbe[2], and Thomas A. Henzinger[2]

[1] Australian National University
[2] IST Austria
[3] Univ. Grenobles Alpes - Verimag

Abstract. Template polyhedra generalize intervals and octagons to polyhedra whose facets are orthogonal to a given set of arbitrary directions. They have been employed in the abstract interpretation of programs and, with particular success, in the reachability analysis of hybrid automata. While previously, the choice of directions has been left to the user or a heuristic, we present a method for the automatic discovery of directions that generalize and eliminate spurious counterexamples. We show that for the class of convex hybrid automata, i.e., hybrid automata with (possibly nonlinear) convex constraints on derivatives, such directions always exist and can be found using convex optimization. We embed our method inside a CEGAR loop, thus enabling the time-unbounded reachability analysis of an important and richer class of hybrid automata than was previously possible. We evaluate our method on several benchmarks, demonstrating also its superior efficiency for the special case of linear hybrid automata.

1 Introduction

Template polyhedra are convex polyhedra whose defining halfspaces are orthogonal to a template, i.e., a finite set of directions. In other words, they are those conjunctions of linear inequalities where all coefficients are fixed and constants can vary. Template polyhedra naturally generalize geometrical representations like intervals or octagons, yet maintain low computational cost for several set operations. Template polyhedra have been employed for the abstract interpretation of programs [17, 38], but in particular they have recently gained popularity with the abstract interpretation of hybrid automata [27, 37, 25, 18, 12], i.e., the extension of finite automata with continuous dynamics [26]. In fact, verification of hybrid automata via template polyhedra has shown promise in practice [23, 13, 20, 8, 35], in spite of the theoretical undecidability even for the reachability question [29]. In this paper, we develop a novel abstraction refinement procedure for template polyhedra and we evaluate its use in the time-unbounded reachability analysis of hybrid automata.

* This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), by the European Commission under grant 643921 (UnCoVerCPS), and by the ARC project DP140104219 (Robust AI Planning for Hybrid Systems). The final publication will be available at link.springer.com.

Efficiency often comes at the price of precision, and template polyhedra are no exception. The precision is sensitive to the choice of template and a bad one might cause several problems. First, even computing the tightest of the template polyhedra around a set won't necessarily bring to an exact representation. This holds for linear sets, think about using intervals or octagons for representing arbitrary polyhedra, and for non linear sets, think about using any finite set of directions for representing ellipses or parabolae. Second, template polyhedra suffer from the so called wrapping effect, that is to say that even if you represent initial and guard constraints of a hybrid automaton precisely, discrete transitions and time elapse might make new directions necessary. Think about representing a box using intervals, applying a slight rotation, and representing it again using intervals. Thus the question is: how do you choose the template?

The current approaches for the abstract interpretation by means of template polyhedra are affected by multiple problems. First, they do not guarantee avoidance of spurious counterexamples. In fact, they either assume a priori fixed templates or derive directions from initial and guard constraints [38, 37]. The online refinement techniques focus on improving local errors rather than inductively eliminating and generalizing whole paths [6, 22]. Counterexample-guided methods have been developed, but not for template polyhedra [15, 4, 19, 7]. Second, they partition and bound the time domain. Differential equations are in general hard to solve, thus time partitioning is often necessary [25, 23]. Efforts in taking larger time intervals have been made, but not for unbounded time [24]. Third, the approaches to the analysis of non-linear systems do not handle template refinement, even offline. The abstraction refinement of Bernstein and Taylor expansion-based approximations relies on global parameters that are hard to infer from counterexamples [18, 39, 20, 12, 13].

We propose a method which, for the first time, discovers template directions from spurious counterexamples and adds to the template a few of them at a time. Let us look at a refinement workflow. Initially, we search for a spurious counterexample using a fixed (and possibly empty) template. Once such a counterexample is found, we extract an inductive sequence of halfspace interpolants, i.e., Craig's interpolants that consist of single linear inequalities [2]. We take their outward pointing directions and we add them to the template. Such directions eliminate the counterexample and generalize to all other counterexamples with the same switching sequence and any (and possibly unbounded) time elapse. We repeat the procedure in CEGAR fashion [16].

We target the time-unbounded reachability analysis of convex hybrid automata (CHA), i.e., hybrid automata whose flow constraints consist of differential inclusions (on derivatives only) and all constraints (flow, guards, and invariants) are (possibly non-linear) closed convex sets, and the special cases of linear hybrid automata (LHA) and quadratic hybrid automata (QHA). A large class of systems belongs to CHA, e.g., timed systems with convex non-linear clock drifts, or can be approximated as CHA, e.g., systems with Gaussian disturbances truncated by elliptic sets. The reachability analysis of LHA has a long history [5], while for QHA or beyond only bounded reachability analysis has been explored [11, 14].

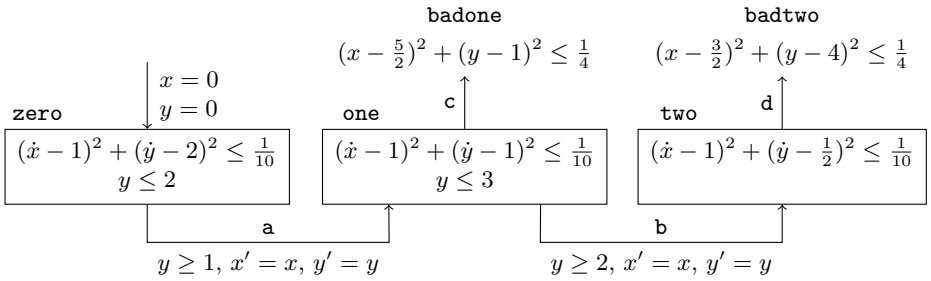


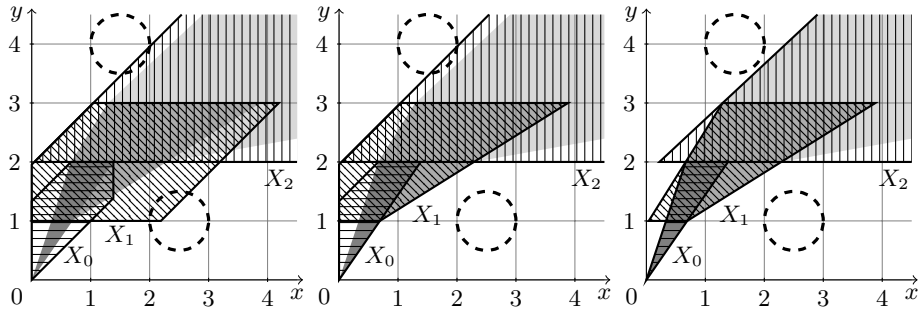
Fig. 1. A CHA with two variables x and y, three good modes `zero`, `one`, and `two`, two bad modes `badone` and `badtwo`, and four switches `a`, `b`, `c`, and `d`. The good modes have three different relative speeds for x and y with an additional spherical drift. All invariants, the jump guards of `a` and of `b` are linear and the jump guards of `c` and of `d` are spherical.

We show that (i) for every CHA halfspace interpolants suitable for refinement always exist and that (ii) they can be computed efficiently using convex optimization [10], in particular using linear programming for LHA and second-order conic programming for QHA. We implement a tool based of this technology and evaluate it on several linear and quadratic benchmarks, comparing (favorably) against PHAVer where that tool applies [21, 23], namely LHA. This gives the following new results. First, we enable the use of template polyhedra for the abstract interpretation and the abstraction refinement of CHA, thus enabling the efficient time-unbounded reachability analysis for the full class where efficient convex optimizers are available, namely QHA. Second, we achieve greater practical performance against the state-of-the-art techniques for the time-unbounded reachability of even LHA. We evaluate our tool on multiple scaling and linear and non-linear variants of three different benchmarks, namely Fischer's protocol [31], the TTEthernet protocol [9], and an adaptive cruise controller [30].

In summary, our contribution is threefold. First, we develop the first complete counterexample-guided procedure for the discovery of template directions. Second, we enable, for the first time, abstraction refinement for the time-unbounded reachability analysis for all CHA. Third, we build an efficient tool for the new class of time-unbounded QHA verification, which shows superior performance also for the special case of LHA.

2 Motivating Example

Consider a system with two real-valued variables x and y whose dynamics follows some differential equation, which in turn is discontinuously switched by an automaton with three modes. Figure 1 shows such an example. The trajectory starts in the origin and enters mode `zero` and follows any differential equation whose derivative is $\dot{x}=1$ and $\dot{y}=2$ with possibly some drift in the ball of radius $10^{-\frac{1}{2}}$ around this value. The invariant allows the trajectory to stay in mode `zero`



(a) octagonal abstraction (b) w/o path to `badone` (c) w/o path to `badtwo`

Fig. 2. Template-polyhedral abstraction refinement for the CHA in Fig. 1. In dark gray, gray, and light gray the points reachable on the modes `zero`, `one`, and `two`, resp., and the striped polyhedra X_0, X_1, and X_2 are the resp. template polyhedra. The lower and the upper dashed circles are, resp., the guards of the switches `c` and `d` to the bad modes. The variant (a) show the octagonal abstraction, and (b) and (c) show resp. the results of the templates obtained after refinement of the paths to `badone` and then to `badtwo`.

as long as $y \leq 2$. The trajectory can take `a` if $y \geq 1$ and switch to mode `one`, where the derivative of y halves. The dynamics continues similarly on mode `one`, switch `c`, and mode `two`, and similarly can take a switch to `badone` and `badtwo` when the respective guards are satisfied. We know that there does not exists a trajectory that leads to one of the bad modes, namely the system is *safe*. We want to prove it automatically by means of template polyhedra.

The set of states that are respectively reachable on modes `zero`, `one`, and `two` are the cones spanned by the points that enter the mode and take any possible trajectory, as respectively depicted in Fig. 2 in three shades on gray. We abstract the whole systems by representing each of these sets using template polyhedra. But first, we need to discover a suitable template. In fact, different templates produce different abstractions and not all of them can prove safety. Figure 2 shows three different such abstractions (striped polyhedra), but (a) and (b) hit the guards (dashed circled) to the bad modes while only (c) accomplishes the task. Our goal is to construct a good template like in (c).

We begin with abstraction (a) which uses the octagonal template, i.e., the 8 orthogonal directions to the facets of an octagon. The abstract interpreter will produce several abstract paths (sequences of pairs of modes and polyhedra interleaved by switches) among which will occur the path $\texttt{zero}, \texttt{a}, \texttt{one}, \texttt{c}, \texttt{badone}$, for the regions $X_0, X_1 \subseteq \mathbb{R}^n$ where $X_0 = \mathsf{init}_{\texttt{zero}}$ abstracts the flow on `zero`, and $X_1 = \mathsf{post}_{\texttt{a}}(X_0)$ abstracts the flow on `one` (see Fig. 2a). This path reaches a bad mode, but it is spurious, namely it does not have a concrete counterpart. We prove it by computing a sequence of halfspace interpolants, i.e., two halfspaces H_0 and H_1 such that $\mathsf{init}_{\texttt{zero}} \subseteq H_0$ and $\mathsf{post}_{\texttt{a}}(H_0) \subseteq H_1$ and H_1 does not intersect with the guard of the switch `c` (see Fig. 3b). The outward pointing directions d_0 and d_1 of H_0 and H_1 are the directions that generalize and eliminate all counterexamples

with the switching sequence `zero`, `a`, `one`, `c`, `badone` (see Fig. 3c). We add them to the template and we recompute the abstraction, obtaining a necessarily different counterexample (see Fig. 2b). We repeat and eventually obtain Fig. 2c, finally proving the safety of the hybrid automaton.

In the next section we define the modeling and the (template-polyhedral) abstraction framework for CHA. In Sec. 4 we present our interpolant-based refinement technique and in Sec. 5 we phrase it as a convex optimization problem. In Sec. 6 we instantiate it to QHA and in Sec. 7 we show our experimental results.

3 Template-polyhedral Abstractions for Convex Systems

Hybrid automata extend finite automata adding constraints on the (discrete and continuous) dynamics of a set of real variables [26]. Convex hybrid automata (CHA) are the class whose constraint define non-linear convex sets that exclusively constrain either variables or variable derivatives, as it is the case for the well-know class of linear hybrid automata (LHA) [26], which is thus generalized by CHA.

Definition 1 (Convex hybrid automata). *A* convex hybrid automaton $\mathcal{H}$ *with n real-valued variables consists of a finite directed multigraph (V, E) where the vertices $v \in V$ are called control modes and the edges $e \in E$ are called control switches. Each $v \in V$ is decorated by an initial constraint $Z_v \subseteq \mathbb{R}^n$, an invariant constraint $I_v \subseteq \mathbb{R}^n$, and a flow constraint $F_v \subseteq \mathbb{R}^n$, each $e \in E$ is decorated by a jump constraint $J_e \subseteq \mathbb{R}^{2n}$, and all constraints define closed convex sets.*

A finite *control path* $v_0, e_1, v_1, \ldots, e_k, v_k$ of the CHA $\mathcal{H}$ is a path of the control graph of $\mathcal{H}$, i.e., for all $0 \leq i \leq k$ it holds that $v_i \in V$ and for all $1 \leq i \leq k$ it holds that $e_i \in E$ and is a switch with source v_{i-1} and destination v_i. When a control path is clear from the context, we abbreviate any object indexed by v_i or e_i as the same object indexed by i, e.g., we abbreviate F_{v_i} as F_i. The semantics associates modes to points $x \in \mathbb{R}^n$. For every two points $x, x' \in \mathbb{R}^n$, for every control mode $v \in V$ we say that x' is a *v-successor* of x if there exists a derivable function $f \colon \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and a time delay $\delta \in \mathbb{R}_{\geq 0}$ such that $f(0) = x$, $f(\delta) = x'$, and for all $0 \leq \gamma \leq \delta$ it holds that $\dot{f}(\gamma) \in F_v$ and $f(\gamma) \in I_v$, and for every control switch $e \in E$ we say that x' is an *e-successor* of x if $(x, x') \in J_e$.

Definition 2 ($\mathcal{H}$-feasibility). *A finite control path $v_0, e_1, v_1, \ldots, e_k, v_k$ is $\mathcal{H}$-feasible if for some $x_0, x'_0, x_1, x'_1, \ldots, x_k, x'_k \in \mathbb{R}^n$ it holds that $x_0 \in Z_0$, and for all $0 \leq i \leq k$, x'_i is a v_i-successor of x_i and x_i is a e_i-successor of x'_{i-1}.*

The semantics of $\mathcal{H}$ is the maximal set of $\mathcal{H}$-feasible paths. A mode $v \in V$ is *reachable* if there exists an $\mathcal{H}$-feasible control path whose last mode is v, and a point $x' \in \mathbb{R}^n$ is reachable on v if x' is the last point of a sequence as in Def. 2.

The abstraction associates modes to regions of $\mathbb{R}^n$ into abstract paths whose elements are related by the init and post operator of an abstraction structure $\mathcal{A}$.

Definition 3 (Abstraction structure). *An abstraction structure $\mathcal{A}$ for the CHA $\mathcal{H}$ consists of an init operator $\mathsf{init}_v \in \wp(\mathbb{R}^n)$ for every $v \in V$ and of a post operator $\mathsf{post}_e \colon \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ for every $e \in E$.*

Similarly as for $\mathcal{H}$, a control path with an abstract counterpart is called $\mathcal{A}$-feasible.

Definition 4 ($\mathcal{A}$-feasibility). *A finite control path $v_0, e_1, v_1, \ldots, e_k, v_k$ is $\mathcal{A}$-feasible if for some non-empty sets $X_0, X_1, \ldots, X_k \subseteq \mathbb{R}^n$ holds that $X_0 = \mathsf{init}_0$ and for all $1 \leq i \leq k$, $X_i = \mathsf{post}_i(X_{i-1})$.*

An $\mathcal{A}$-feasible path is *genuine* if it is also $\mathcal{H}$-feasible, and *spurious* otherwise. An abstraction structure $\mathcal{A}$ is *sound* if all $\mathcal{H}$-feasible control paths are $\mathcal{A}$-feasible.

The *support function* [36] in direction $d \in \mathbb{R}^n$ of a convex set $X \subseteq \mathbb{R}^n$ is

$$\rho_X(d) \stackrel{\text{def}}{=} \sup\{d \cdot x \mid x \in X\}. \tag{1}$$

The support function of X characterizes the template polyhedron [38, 25] of X for the template $\Delta \subseteq \mathbb{R}^n$ (a finite set). We call it the *Δ-polyhedron* of X, that is

$$\bigcap_{d \in \Delta} \{x \in \mathbb{R}^n \mid d \cdot x \leq \rho_X(d)\}. \tag{2}$$

We aim at computing template polyhedra for the (continuous) flow and the (discrete) jump post operators (and their compositions) of the hybrid automaton. The *flow operator* of mode $v \in V$ gives the points reachable by time elapse on v:[1]

$$\mathrm{flow}_v(X) \stackrel{\text{def}}{=} (X \oplus \operatorname{coni} F_v) \cap I_v. \tag{3}$$

The *jump operator* of switch $e \in E$ gives the points reachable through e:[2]

$$\mathrm{jump}_e(X) \stackrel{\text{def}}{=} \begin{bmatrix} 0_{n\times n} & I_n \end{bmatrix} \left(\left(\begin{bmatrix} I_n \\ 0_{n\times n} \end{bmatrix} X \oplus \begin{bmatrix} 0_{n\times n} \\ I_n \end{bmatrix} \mathbb{R}^n \right) \cap J_e \right). \tag{4}$$

Flow and jump operators are an exact symbolical characterization for the semantics of CHA, and follow as an extension of the symbolic analysis of LHA [26].

Lemma 1. *For every CHA $\mathcal{H}$ and every set $X \subseteq \mathbb{R}^n$ it holds that (i) $x' \in \mathrm{flow}_v(X)$ if and only if x' is a v-successor of some $x \in X$ for every control mode $v \in V$ and (ii) $x' \in \mathrm{jump}_e(X)$ if and only if x' is a e-successor of some $x \in X$ for every control switch $e \in E$.*

The exact symbolic analysis of CHA has in general high complexity, as it requires eliminating quantifiers, and possibly from formulae that contain non-linear constraints. For this reason we approximate them using template polyhedra.

The template-polyhedral abstraction computes the template polyhedra of the flow and jump operators above and, in our definition, using a different template for each mode, given by the *precision function* $\mathsf{prec}\colon V \to \wp(\mathbb{R}^n)$.

Definition 5 (Template-polyhedral abstraction). *The template-polyhedral abstraction for the CHA $\mathcal{H}$ and the precision function $\mathsf{prec}\colon V \to \wp(\mathbb{R}^n)$ is the abstraction structure where the init operator init_v is the $\mathsf{prec}(v)$-polyhedron of $\mathit{flow}_v(Z_v)$, and the post operator $\mathsf{post}_e(X)$ is the $\mathsf{prec}(t)$-polyhedron of $\mathit{flow}_t \circ \mathit{jump}_e(X)$ where $t \in V$ is the destination of e.*

[1] For $X \subseteq \mathbb{R}^n$, $\operatorname{coni} X$ denotes the conical combination $\{0\} \cup \{\alpha x \mid \alpha \geq 0 \wedge x \in X\}$ and for $Y \subseteq \mathbb{R}^n$, $X \oplus Y$ denotes the Minkowski sum $\{x + y \mid x \in X \wedge y \in Y\}$.

[2] For $M \in \mathbb{R}^{m\times n}$ and $X \subseteq \mathbb{R}^n$, MX denotes the linear transformation $\{Mx \mid x \in X\}$.

It is well-know that the template-polyhedral abstraction constructs a conservative over-approximation for linear systems [38], and the same holds for CHA.

Theorem 1. *For every CHA $\mathcal{H}$ and every precision function* prec *the template-polyhedral abstraction for $\mathcal{H}$ and* prec *is sound.*

The obvious difficulty is in finding a precision function that is suitable for proving or disproving reachability. In the next section, we show how to form one such automatically by means of counter-example guided abstraction refinement.

4 Refining the Template-polyhedral Abstraction

A counter-example guided abstraction refinement (CEGAR) loop [16] for a hybrid automaton $\mathcal{H}$ and a set of bad modes T consists of an abstractor and a refiner interacting with each other. At each iteration i, the *abstractor* takes an abstraction structure $\mathcal{A}_i$ and attempts to construct the finite state machine that recognizes all $\mathcal{A}_i$-feasible paths. If it terminates and it does not find a counterexample, i.e., a path leading to a bad mode, then it returns `no`. Otherwise, it passes $\mathcal{A}_i$ and a set of counterexamples W_i to the refiner. The *refiner* attempts to construct an abstraction structure $\mathcal{A}_{i+1}$ that refines $\mathcal{A}_i$ and eliminates all counterexamples in W_i. If it fails, then it reports `yes` and a set $\bar{W}_i \subseteq W_i$ of genuine counterexamples. Otherwise, it passes $\mathcal{A}_{i+1}$ to the abstractor.

The above procedure is sound (upon termination), provided $\mathcal{A}_i$ is sound, in the sense that if it reports `no` then no mode in T is reachable. It is complete (upon termination), namely if it reports `yes` then some mode in T is reachable, if it returns an abstraction $\mathcal{A}_{i+1}$ that is locally complete w.r.t. W_i when one exists.

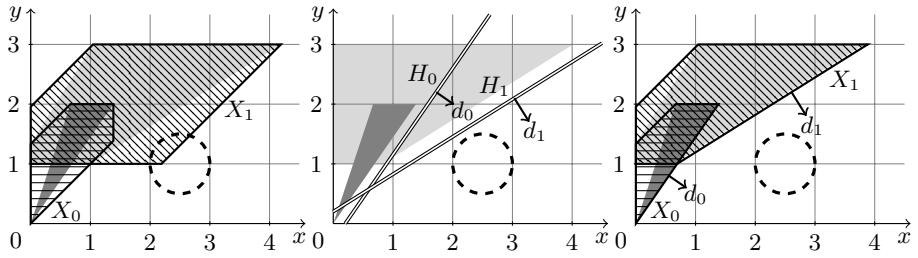
Local completeness An abstraction structure $\mathcal{A}$ for the CHA $\mathcal{H}$ is locally complete w.r.t. the set W of control paths of $\mathcal{H}$ if all $\mathcal{H}$-infeasible control paths in W are $\mathcal{A}$-infeasible.

Moreover, if it ensures local completeness w.r.t. $\cup\{W_j | 0 \leq j \leq i\}$, then it ensures progress of the procedure if the counterexamples are given one by one.

Whenever we find a spurious counterexample, we augment the precision of the modes along the path with additional template directions, so to make it $\mathcal{A}$-infeasible. First of all, we start with finding a sequence of Craig's interpolants and only Craig's interpolants that are halfspaces [2]. Formally, let $w = v_0, e_1, v_1, \ldots, e_k, v_k$ be a control path of $\mathcal{H}$, then a sequence of *halfspace interpolants* for w is a sequence of sets $H_0, H_1, \ldots, H_k \subseteq \mathbb{R}^n$ such that each element is either the universe, a closed halfspace, or the empty set and

$$\text{flow}_0(Z_0) \subseteq H_0, \text{flow}_1 \circ \text{jump}_1(H_0) \subseteq H_1, \ldots, \text{flow}_k \circ \text{jump}_k(H_{k-1}) \subseteq H_k, \quad (5)$$

and $H_k \subseteq \emptyset$. If such sequence exists, then the path is clearly $\mathcal{H}$-infeasible. Conversely, it is not trivial that for every $\mathcal{H}$-infeasible path such sequence exists.



(a) octagonal abstraction (b) halfspace interpolants (c) refined abstraction

Fig. 3. Refinement for the control path `zero`, `a`, `one`, `b`, `badone` of the CHA in Fig. 1. In dark gray, the points reachable on mode `zero`. In (a), (b), and (c), in light gray are the points reachable on mode `one` resp. from X_0, H_0, and X_0. In (a) the spurious path, in (b) the interpolants, and in (c) the abstraction with the outward pointing directions.

Lemma 2. *For every CHA $\mathcal{H}$ and every control path w of $\mathcal{H}$ it holds that w is $\mathcal{H}$-infeasible if (and only if)*[3] *there exists a sequence $H_0, H_1, \ldots, H_k \subseteq \mathbb{R}^n$ of halfspace interpolants for w as in Eq. 5.*

Indeed, existence relies on further technical conditions that are out of the scope of this paper [36]. With this in mind, simply assuming all non-linear constraints to be bounded (e.g., Fig. 1) ensures existence, yet without preventing time-unbounded reachability. Computing interpolants is the subject of the next section.

The refining directions are the outward pointing directions of the halfspace interpolants, respectively for each mode along the path. In fact, it is enough to observe that every abstraction we obtain after adding such directions also satisfy

$$\mathsf{init}_0 \subseteq H_0, \mathsf{post}_1(H_0) \subseteq H_1, \ldots, \mathsf{post}_k(H_{k-1}) \subseteq H_k. \tag{6}$$

Figure 3 shows such an example. The path is the one leading to `badone` from the CHA of Fig. 1, which is spurious with octagonal template (see Fig. 3a), and, in fact, a sequence H_0 and H_1 of halfspace interpolants exists (see Fig. 3b). The halfspace H_1 is disjoint from the guard of `c` (dashed circle) and includes the points reachable from H_0 (light gray), which in its turn includes the points reachable from $Z_{\texttt{zero}}$, i.e., $\mathrm{flow}_{\texttt{zero}}(Z_{\texttt{zero}}) \subseteq H_0$, $\mathrm{flow}_{\texttt{one}} \circ \mathrm{jump}_{\texttt{a}}(H_0) \subseteq H_1$, and $\mathrm{jump}_{\texttt{c}}(H_1) \subseteq \emptyset$. Taking the supporting halfspaces in the same directions preserves these inclusions, hence adding d_0 to $\mathsf{prec}(\texttt{zero})$ and d_1 to $\mathsf{prec}(\texttt{one})$ causes $\mathsf{init}_{\texttt{zero}} \subseteq H_0$, $\mathsf{post}_{\texttt{a}}(H_0) \subseteq H_1$, and $\mathsf{post}_{\texttt{c}}(H_1) \subseteq \emptyset$. Thus d_0 and d_1 eliminate the counterexample, and regardless of whether prec contains further directions (see Fig. 3c).

Definition 6 (Template-polyhedral refinement). *Let $\mathcal{H}$ be a CHA and let $w = v_0, e_1, v_1, \ldots, e_k, v_k$ be a control path. Define the precision function* prec *such that for some (if one exists) sequence of halfspace interpolants $H_0, H_1, \ldots, H_k \subseteq \mathbb{R}^n$ for w as in Eq. 5 then for all $0 \leq i \leq k$ set $d_i \in \mathsf{prec}(v_i)$ where d_i is the outward pointing direction of H_i. We define the template-polyhedral refinement for $\mathcal{H}$ and w as the template-polyhedral abstraction for $\mathcal{H}$ and* prec.

[3] We exclude the pathological cases of disjoint convex sets w/o separating hyperplane.

Local completeness w.r.t. a single path easily generalizes to local completeness w.r.t. multiple paths by taking the union of the discovered directions.

Theorem 2. *For every CHA $\mathcal{H}$ and every set W of finite control paths of $\mathcal{H}$ the union*[4] *over all $w \in W$ of the template-polyhedral refinements for $\mathcal{H}$ and w is locally complete w.r.t. W.*

Summarizing, we search for abstract counterexamples and we accumulate all outward pointing directions of the respective halfspace interpolants. If either the abstractor finds a fixpoint or interpolation fails, then we obtain a sound and complete answer. In the following section, we show how to compute init and post operators and sequences of halfspace interpolants by using convex optimization.

5 Craig's Interpolation as Convex Optimization

The support function is a central actor both in abstraction, as it defines template polyhedra, and refinement, as it gives a powerful formalism to talk about inclusion in halfspaces and separation of convex sets. In either case, the sets we deal with are arbitrary compositions of flow and jump operators, which in their turn are compositions of Minkowski sums, linear transformations, conical combinations, and intersections. We characterize the support functions of such operations as convex programs, with the aim of characterizing abstraction and refinement as convex programs.

We present a characterization of support functions that is compositional for the set operations above. The *classic* support function representation framework[5] offers a very similar machinery [25], but it suffers from the following shortcomings. First, it requires the operand sets in Minkowski sums and intersections to be compact (i.e., closed and bounded) and boundedness cannot be easily relaxed, e.g., $\rho_{\mathbb{R}^n}(d) + \rho_{\emptyset}(d) = +\infty - \infty$ while $\rho_{\mathbb{R}^n \oplus \emptyset}(d) = -\infty$ for every $d \neq 0$. Since we aim at time-unbounded reachability, is would be too restrictive to assume boundedness. Second, substituting boundedness with nonemptiness might cause uncorrect results, e.g., for the sets $A = \{(x, y) \mid x \leq -1\}$, $B = \{(x, y) \mid x \geq 1\}$, and the direction $c = (0, 1)$ we obtain $\inf\{\rho_A(c - a) + \rho_B(a)\} = +\infty$, while $\rho_{A \cap B}(c) = -\infty$. We relax both the assumptions of boundedness and nonemptiness by characterizing the support function $\rho_X(d)$ with a convex program

$$\begin{array}{l} \text{minimize} \;\; \bar{\rho}_X(\lambda) \\ \text{subject to } (\lambda, d) \in \Lambda_X, \end{array} \tag{7}$$

with objective function $\bar{\rho}_X : \mathbb{R}^m \to \mathbb{R}$ and constraint $\Lambda_X \subseteq \mathbb{R}^{m+n}$. The minimum of $\bar{\rho}_X(\lambda)$ over λ characterizes $\rho_X(d)$ for directions in which X is bounded, while Λ_X characterizes boundedness. This is encapsulated by the notion of duality.

[4] The union of the abstractions $\mathcal{A}_1, \ldots, \mathcal{A}_i$ for $\mathcal{H}$ and resp. the precisions $\mathsf{prec}_1, \ldots, \mathsf{prec}_i$ is the abstraction for $\mathcal{H}$ and the precision $\lambda v.\mathsf{prec}_1(v) \cup \cdots \cup \mathsf{prec}_i(v)$.

[5] $\rho_{X \oplus Y}(d) = \rho_X(d) + \rho_Y(d)$, $\rho_{MX}(d) = \rho_X(M^{\mathsf{T}} d)$, and $\rho_{X \cap Y}(d) = \inf\{\rho_X(a) + \rho_Y(d - a)\}$.

Duality Let $X \subseteq \mathbb{R}^n$ be a nonempty closed convex set. The convex program of Eq. 7 is *dual* to ρ_X if for all $d \in \mathbb{R}^n$ it holds that
(i) $\rho_X(d) = +\infty$ if and only if there does not exist λ such that $(\lambda, d) \in \Lambda_X$,
(ii) $\rho_X(d) < +\infty$ if and only if $\rho_X(d) = \min\{\bar{\rho}_X(\lambda) \mid (\lambda, d) \in \Lambda_X\}$.

We define inductive rules for constructing dual convex programs for the support functions of set operations, provided dual convex programs for their operands (whose instantiation for sets defined by symbolic constraints is subject of Sec. 6):

$$\begin{aligned} \bar{\rho}_{X \oplus Y}(\lambda, \mu) &\stackrel{\text{def}}{=} \bar{\rho}_X(\lambda) + \bar{\rho}_Y(\mu), \\ \Lambda_{X \oplus Y} &\stackrel{\text{def}}{=} \{(\lambda, \mu, d) \mid (\lambda, d) \in \Lambda_X, (\mu, d) \in \Lambda_Y\}, \end{aligned} \tag{8}$$

$$\begin{aligned} \bar{\rho}_{MX}(\lambda) &\stackrel{\text{def}}{=} \bar{\rho}_X(\lambda), \\ \Lambda_{MX} &\stackrel{\text{def}}{=} \{(\lambda, d) \mid (\lambda, M^\mathsf{T} d) \in \Lambda_X\}, \end{aligned} \tag{9}$$

$$\begin{aligned} \bar{\rho}_{\operatorname{coni} X}(\lambda) &\stackrel{\text{def}}{=} 0, \\ \Lambda_{\operatorname{coni} X} &\stackrel{\text{def}}{=} \{(\lambda, d) \mid \bar{\rho}_X(\lambda) \leq 0, (\lambda, d) \in \Lambda_X\}, \end{aligned} \tag{10}$$

$$\begin{aligned} \bar{\rho}_{X \cap Y}(\lambda, \mu) &\stackrel{\text{def}}{=} \bar{\rho}_X(\lambda) + \bar{\rho}_Y(\mu), \text{ and} \\ \Lambda_{X \cap Y} &\stackrel{\text{def}}{=} \{(\lambda, \mu, a, d) \mid (\lambda, a) \in \Lambda_X, (\mu, d - a) \in \Lambda_Y\}. \end{aligned} \tag{11}$$

Nevertheless, duality is not sufficient to characterize operations producing the empty set. Considering the examples above, the constraint $\Lambda_{\mathbb{R}^n \oplus \emptyset}$ is infeasible for every direction $d \neq 0$ and the constraint $\Lambda_{A \cap B}$ is infeasible for direction c, contradicting (i). However, it suffices that the convex program is unbounded for at least $d = 0$, providing an alternative for deciding emptiness beforehand.

Alternativity The convex program of Eq. 7 is *alternative* to $\rho_\emptyset$ if for every $\epsilon < 0$ there exists $(\lambda, 0) \in \Lambda_\emptyset$ such that $\bar{\rho}_\emptyset(\lambda) \leq \epsilon$.

Altogether, we compute the support of X in direction d as follows. We decide whether there exists a negative solution in direction 0. If so we return $-\infty$, otherwise we decide whether Λ_X is infeasible in direction d. If so we return $+\infty$, otherwise we solve the convex program. This is permitted on any combination of the set operations above, as our construction preserves duality and alternativity.

Lemma 3. *Let $X, Y \subseteq \mathbb{R}^n$ be closed convex sets. If the convex programs for $\bar{\rho}_X, \Lambda_X$ and $\bar{\rho}_Y, \Lambda_Y$ are dual and alternative to resp. ρ_X and ρ_Y then the convex programs for Eq. 8, 9, and 10 are dual and alternative to the respective support functions. If either X and Y intersect or they admit a separating hyperplane then also the convex program for Eq. 11 is dual and alternative to $\rho_{X \cap Y}$.*

In addition, the construction allows us to inductively extract separating hyperplanes and therefore sequences of halfspace interpolants.

The emptiness check or more generally deciding whether a support function is below a threshold permits us to inductively extract interpolants. For each of the four set operation we wish first to prove inclusion within a given halfspace (or the empty set) H and then to find a second halfspace H' which interpolates the operand. For instance, for an intersection $X \cap Y$ such that $X \cap Y \subseteq H$, we wish to find a H' such that $X \subseteq H'$ and $H' \cap Y \subseteq H$. Indeed, we just need their outward pointing directions, and our construction carries this information.

Lemma 4. *Let $X, Y \subseteq \mathbb{R}^n$ be closed convex sets. Let the convex programs for $\bar{\rho}_X, \Lambda_X$ and $\bar{\rho}_Y, \Lambda_Y$ be dual and alternative to ρ_X and ρ_Y. Let H be the set $\{x \in \mathbb{R}^n \mid d \cdot x \leq \epsilon\}$, which is empty if and only if $d = 0$ and $\epsilon < 0$.*

- *If either X and Y are both nonempty or H is empty then for every $(\lambda^\star, \mu^\star, d) \in \Lambda_{X \oplus Y}$ such that $\bar{\rho}_{X \oplus Y}(\lambda^\star, \mu^\star) \leq \epsilon$ there exists $H' = \{x \in \mathbb{R}^n \mid d \cdot x \leq \epsilon'\}$ such that $\bar{\rho}_X(\lambda^\star) \leq \epsilon'$ and $H' \oplus Y \subseteq H$.*
- *If either X is nonempty or H is empty then for every $(\lambda^\star, d) \in \Lambda_{MX}$ such that $\bar{\rho}_{MX}(\lambda^\star) \leq \epsilon$ there exists $H' = \{x \in \mathbb{R}^n \mid (M^\mathsf{T} d) \cdot x \leq \epsilon'\}$ such that $\bar{\rho}_X(\lambda^\star) \leq \epsilon'$ and $MH' \subseteq H$.*
- *For every $(\lambda^\star, d) \in \Lambda_{\operatorname{coni} X}$ such that $\bar{\rho}_{\operatorname{coni} X}(\lambda^\star) \leq \epsilon$ there exists $H' = \{x \in \mathbb{R}^n \mid d \cdot x \leq \epsilon'\}$ such that $\bar{\rho}_X(\lambda^\star) \leq \epsilon'$ and $\operatorname{coni} H' \subseteq H$.*
- *If either X and Y intersect or H is empty and they admit a separating hyperplane then for every $(\lambda^\star, \mu^\star, a^\star, d) \in \Lambda_{X \cap Y}$ such that $\bar{\rho}_{X \cap Y}(\lambda^\star, \mu^\star) \leq \epsilon$ there exists $H' = \{x \in \mathbb{R}^n \mid a^\star \cdot x \leq \epsilon'\}$ such that $\bar{\rho}_X(\lambda^\star) \leq \epsilon'$ and $H' \cap Y \subseteq H$.*

We can extract the outward pointing directions by looking at the arguments instantiated by an emptiness check. Inductively, if d is the outward pointing direction of H, then the outward pointing direction of H' is d for the Minkowski sum, $M^\mathsf{T} d$ for the linear transformation, d for the conical combination, and a for the intersection. As a result, we can extract sequences of interpolants for arbitrary combinations of basic set operations from one single emptiness check.

We build such a construction for arbitrary sequences of flow and jump operators induced by control paths. More concretely, let $w = v_0, e_1, v_1, \ldots, e_k, v_k$ be a control path of some CHA $\mathcal{H}$ then the *path operator* of w is

$$P_w \stackrel{\text{def}}{=} \operatorname{flow}_k \circ \operatorname{jump}_k \circ \cdots \circ \operatorname{flow}_1 \circ \operatorname{jump}_1 \circ \operatorname{flow}_0(Z_0). \tag{12}$$

Similarly to Lem. 2, we assume every path operator to be either nonempty or to admit a separating hyperplane at some intersection. By applying the above rules, we construct the convex program for the support function of P_w as follows:

$$\begin{array}{lll} \text{minimize} & \bar{\rho}_{Z_0}(\lambda_{Z_0}) + \sum_{i=1}^{k} \bar{\rho}_{J_i}(\lambda_{J_i}) + \sum_{i=0}^{k} \bar{\rho}_{I_i}(\lambda_{I_i}) & \\ \text{subject to} & (\lambda_{Z_0}, a_0 - b_0) & \in \Lambda_{Z_0}, \\ & (\lambda_{J_i}, \left[-a_{i-1}, a_i - b_i\right]^\mathsf{T}) & \in \Lambda_{J_i} \quad \text{for each } i \in [1..k], \\ & \bar{\rho}_{F_i}(\lambda_{F_i}, a_i - b_i) & \leq 0 \quad \text{for each } i \in [0..k], \\ & (\lambda_{F_i}, a_i - b_i) & \in \Lambda_{F_i} \quad \text{for each } i \in [0..k], \\ & (\lambda_{I_i}, b_i) & \in \Lambda_{I_i} \quad \text{for each } i \in [0..k], \\ & a_k & = d. \end{array} \tag{13}$$

Duality and alternativity is preserved, therefore we can use such construction to compute the support functions for init and post (which are special cases of path).

Lemma 5. *For every CHA $\mathcal{H}$, every control path w of $\mathcal{H}$, if the convex programs for every constraint X along the path are dual and alternative to ρ_X then the convex program in Eq. 13 is dual and alternative to ρ_{P_w}.*

We identify the arguments that determine a suitable sequence of halfspace interpolants after the emptiness check.

Lemma 6. *For every CHA $\mathcal{H}$, every control path w of $\mathcal{H}$, every $\epsilon < 0$, and every $(\lambda^\star, 0) \in \Lambda_{P_w}$ whose projection on $a_0, a_1, \ldots, a_k$ is $a_0^\star, a_1^\star, \ldots, a_k^\star \in \mathbb{R}^n$, if the convex programs for the constraints X along the path are dual and alternative to ρ_X then $\bar{\rho}_{P_w}(\lambda^\star, 0) \leq \epsilon$ if and only if $a_0^\star, a_1^\star, \ldots, a_k^\star$ are the outward pointing directions of a sequence of halfspace interpolants $H_0, H_1, \ldots, H_k$ for w as in Eq. 5.*

In summary, we search by convex optimization for an argument for which the convex program of Eq. 13 for $d = 0$ has negative solution. If so, the argument $a_i^\star$ for the parameter a_i is the outward pointing direction for the interpolant at mode v_i. Adding $a_i^\star$ to $\mathsf{prec}(v_i)$ eliminates the spurious counterexample w.

In this section, we have built a refiner for every spurious path of every CHA, assuming dual and alternative convex programs for the constraints along the path. In the following section, we discuss such functions and show how to instantiate interpolation for the special case of quadratic hybrid automata.

6 Abstraction Refinement for Quadratic Systems

The interpolation technique in Sec. 5 relies on the notions of duality and alternativity. Duality and alternativity are preserved by Minkowski sum, linear transformation, conical combination, and intersection, but whether they hold in the first place depends on the constraint of the automaton. We discuss these properties for (convex) quadratic programs, and we show their implications to the classes of quadratic and linear hybrid automata.

Closed convex quadratic sets are sets of the form $\bigcap_{i=1}^m \{x \in \mathbb{R}^n \mid xQ_ix^\mathsf{T} + p_i^\mathsf{T}x \leq r_i\}$ where $Q_1, \ldots, Q_m \in \mathbb{R}^{n \times n}$ are positive semidefinite matrices of coefficients, $p_1, \ldots, p_m \in \mathbb{R}^n$ are vectors of coefficients, and $r_1, \ldots, r_m \in \mathbb{R}$ are constants. Closed convex quadratic sets characterize quadratic hybrid automata.

Definition 7 (Quadratic hybrid automata). *A quadratic hybrid automaton (QHA) is a CHA whose constraints define closed convex quadratic sets.*

The support function of a convex quadratic set is a quadratically constrained (convex) quadratic program, which is known to cast to second-order conic programming (SOCP) [3]. We cast the support function to an optimization problem over a (rotated) second-order cone and we take its dual [3], so obtaining

$$\begin{array}{ll} \text{minimize} & r_1\lambda_1 + \cdots + r_m\lambda_m \\ \text{subject to} & p_1\lambda_1 + L_1^\mathsf{T}\mu_1 + \cdots + p_m\lambda_m + L_m^\mathsf{T}\mu_m = d, \\ & \lambda_1 \geq \|\mu_1\|_2^2, \ldots, \lambda_m \geq \|\mu_m\|_2^2, \end{array} \tag{14}$$

where $L_1, \ldots, L_m$ are the Cholesky decompositions of $Q_1, \ldots, Q_m$ respectively, and $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ and $\mu_1, \ldots, \mu_m \in \mathbb{R}^n$ are the optimization arguments. Under the regularity conditions for non-linear optimization, e.g., Slater's condition, duality and alternativity hold [10, 3]. Encodings that do not need such conditions exist [34], but are not discussed in this paper.

Every algorithm that solves feasibility and optimization of SOCP solves init and post computation and halfspace interpolation for QHA, thus enabling their template-polyhedral abstraction and abstraction refinement.

Theorem 3. *Let $\mathcal{H}$ be a QHA with n variables and m inequalities. Let the time complexity of SOCP be $\mathsf{socp}(\alpha, \beta, \gamma)$ for α variables, β equalities, and γ cones.*

- *Init and post operators time complexity is $p \times \mathsf{socp}(n \times m, n, m)$ where $p = \max\{|\mathsf{prec}(v)| \mid v \in V\}$ for the precision function prec.*
- *Refinement time complexity is $c \times \mathsf{socp}(n \times m, n \times k, m \times k)$ where $c = |W|$ and $k = \max\{|w| \mid w \in W\}$ for the set of counterexamples W.*

Nevertheless, the complexity SOCP remains an open problem on the Turing machine, while it is known to be in NP $\cap$ coNP on the real number model [34]. On the other hand, several efficient (but incomplete) numerical procedures are available, therefore in practice we can obtain support functions and interpolants, but with weaker guarantees. We are in a better position for the case of linear hybrid automata (LHA) [26], i.e., the special case of QHA where all constraints define polyhedra. For linear hybrid automata, the program of Eq. 14 is always a linear program, i.e., all cones are positive orthants, where duality holds, alternativity is given by Farkas' lemma, and time complexity is polynomial. Hence, for LHA, init operator, post operator, and refinement time complexities are as well polynomial.

7 Experimental Evaluation

We evaluate our algorithms on three main classes of benchmarks, namely Fischer's protocol [31], an adaptive cruise controller [30], and the TTEthernet protocol [9]. For each class, we consider a linear version and a non-linear version, as well as for each a safe version and an unsafe version.

Fischer's protocol is a time based protocol of mutual exclusion between processes. The protocol is correct if two processes are never in the critical section at the same time. For the linear version, the flow constraints are given by $\frac{1}{2} \leq \dot{x}_{\mathtt{1}} \leq \frac{3}{2}, \ldots, \frac{1}{2} \leq \dot{x}_{\mathtt{m}} \leq \frac{3}{2}$, where $x_{\mathtt{i}}$ is the clock of the $\mathtt{i}$-th process, and for the non-linear case, $\sqrt{\dot{x}_{\mathtt{1}}^2 + \cdots + \dot{x}_{\mathtt{m}}^2} \leq 1$. We verify the linear version up to 5 processes and the non-linear version up to 3 processes.

The adaptive cruise controller is a distributed system for safety distance of platoon of cars. Each car either cruises or recovers by slowing down. The relative velocity has a drift $|\dot{x} - \dot{x}_{\mathtt{ldr}}| \leq \frac{1}{2}$ when cruising and $|\dot{x} - \dot{x}_{\mathtt{ldr}} + \varepsilon| \leq \frac{1}{2}$ when recovering, where x and $x_{\mathtt{ldr}}$ are the positions of each car the car in front, resp, and ε is the slow-down. We check for car crashes in platoons up to 7 cars.

Finally, we consider the TTEthernet protocol for the remote synchronization of possibly drifted clocks distributed over multiple components. Similarly to previous case studies, we consider flows defined in terms of intervals and unit balls for linear and non-linear cases, respectively. We verify both linear and non-linear systems with 3, 5, 9, and 17 components.

We implemented a CEGAR loop based on our procedure in C++ and conducted the following experiments on a machine with 2.6GHz CPU and 4 GB of dedicated RAM. We use the GLPK for solving LPs and MOSEK for solving SOCPs [1, 33]. We executed our tool under the *empty* strategy and the *octagonal* strategy. With the empty strategy, the initial precision is empty, which means

Benchmark	Empty					Octagonal					PHAVer
	#spu	#dir	cgr [s]	itp [s]	ver [s]	#spu	#dir	cgr [s]	itp [s]	ver [s]	time [s]
fsr_lnr_2_sf	5	8	0.06	0.02	≈0	0	256 + 0	0.11	0	0.11	≈0
fsr_lnr_3_sf	41	69	1.12	0.02	0.02	12	3456 + 12	5.55	≈0	0.50	1.25
fsr_lnr_4_sf	259	440	33.16	0.29	0.14	221	32768 + 221	1190	0.07	23.06	135
fsr_lnr_5_sf	1379	2335	857	2.08	0.76	oot	256k	oot	oot	oot	78807
fsr_lnr_2_usf	0	0	≈0	0	≈0	0	256 + 0	0	0	0.12	≈0
fsr_lnr_3_usf	0	0	0.03	0	0.03	0	3456 + 0	0	0	0.37	1.01
fsr_lnr_4_usf	0	0	0.06	0	0.06	0	32768 + 0	0	0	1.67	300
fsr_lnr_5_usf	0	0	0.16	0	0.16	0	256k + 0	0	0	13.63	oom
fsr_qdr_2_sf	5	8	5.13	0.10	1.32	0	256 + 0	0	0	8.18	-
fsr_qdr_3_sf	41	69	226	0.44	9.04	12	3456 + 12	3599	0.15	886	-
fsr_qdr_2_usf	0	0	0.66	0	0.66	0	256 + 0	0	0	6.40	-
fsr_qdr_3_usf	0	0	1.76	0	1.76	0	3456 + 0	0	0	26.67	-
acc_lnr_2_sf	2	2	≈0	≈0	≈0	0	32 + 0	0	0	≈0	≈0
acc_lnr_3_sf	8	8	0.04	≈0	≈0	0	144 + 0	0	0	0.19	0.03
acc_lnr_4_sf	24	24	0.39	≈0	0.02	0	512 + 0	0	0	0.87	0.53
acc_lnr_5_sf	64	64	0.94	≈0	0.12	oot	1600	oot	oot	oot	21.78
acc_lnr_6_sf	160	160	42.12	0.07	0.74	oot	4608	oot	oot	oot	1455
acc_lnr_7_sf	384	384	569	0.13	4.22	oot	12544	oot	oot	oot	oot
acc_lnr_2_usf	1	1	≈0	≈0	≈0	0	32 + 0	0	0	≈0	≈0
acc_lnr_3_usf	2	2	≈0	≈0	≈0	0	144 + 0	0	0	0.05	≈0
acc_lnr_4_usf	3	3	≈0	≈0	≈0	0	512 + 0	0	0	0.37	0.18
acc_lnr_5_usf	4	4	≈0	≈0	≈0	0	1600 + 0	0	0	0.61	22.51
acc_lnr_6_usf	5	5	0.06	≈0	0.04	0	4608 + 0	0	0	1.23	4621
acc_lnr_7_usf	6	6	0.17	≈0	0.06	0	12544 + 0	0	0	2.87	oot
tte_lnr_3_sf	17	18	0.17	≈0	≈0	oot	864	oot	oot	oot	oot
tte_lnr_5_sf	49	50	0.32	≈0	≈0	oot	2400	oot	oot	oot	oot
tte_lnr_9_sf	161	162	3.47	≈0	0.06	oot	7776	oot	oot	oot	oot
tte_lnr_17_sf	577	578	239	0.06	1.27	oot	27774	oot	oot	oot	oot
tte_lnr_3_usf	18	24	0.26	≈0	0.05	0	864 + 0	0	0	0.42	oot
tte_lnr_5_usf	60	80	0.85	≈0	0.02	0	2400 + 0	0	0	0.95	oot
tte_lnr_9_usf	216	288	15.65	≈0	0.26	0	7776 + 0	0	0	4.36	oot
tte_lnr_17_usf	816	1088	1722	0.35	8.68	0	27774 + 0	0	0	109	oot
tte_qdr_3_sf	17	18	8.30	0.38	1.36	oot	864	oot	oot	oot	-
tte_qdr_5_sf	49	50	56.31	1.25	4.01	oot	2400	oot	oot	oot	-
tte_qdr_9_sf	161	162	492	3.94	12.29	oot	7776	oot	oot	oot	-
tte_qdr_17_sf	577	578	3325	12.79	47.49	oot	27774	oot	oot	oot	-
tte_qdr_3_usf	18	24	3.65	0.21	0.60	0	864 + 0	0	0	6.33	-
tte_qdr_5_usf	60	80	37.99	0.66	1.82	0	2400 + 0	0	0	21.68	-
tte_qdr_9_usf	216	288	514	2.61	7.32	0	7776 + 0	0	0	58.27	-
tte_qdr_17_usf	816	1088	15515	18.28	58.95	0	27774 + 0	0	0	78.19	-

Table 1. Results of the experimental evaluation. Empty and octagonal indicate the initial precision. #spu is number of discovered spurious counterexamples, #dir is the number of discovered directions (empty case) or initial directions + discovered directions (octagonal case). cgr is the total time spent in unsuccessful abstractions (with spurious counterexample), itp is the total time spent in discovering halfspace interpolants, ver is the time spent in successful abstractions. oot indicates out of time (24 hours), oom indicates out of memory (4Gb), and dash indicates unsupported. The benchmark names are structured as follows. `fsr` indicates Fischer's protocol, `acc` indicates adaptive cruise controller, `tte` indicates TTEthernet, `lnr` indicates linear, `qdr` indicates quadratic, the following number indicates the number of components, and `sf` and `usf` resp. indicate safe and unsafe.

that the very first abstraction computation consists of a simple exploration of the control graph. With the octagonal strategy, the precision at every mode consists of the octagonal template, with a total of $2|V|n^2$ directions over all modes. For all linear instances, we compared against PHAVer [21] (SpaceEx v0.9.8c with PHAVer scenario).

Table 1 shows the results. The empty strategy has on average the best runtime and always outperforms PHAVer. It also outperforms the octagonal strategy for most of the instances. Both strategies spend most of the time in the first phase (CEGAR iterations ending in a spurious counterexample), and take a very short time for the final verification step. For Fischer's protocol the octagonal strategy is always slower than the empty. For the other benchmarks the difference is less stunning, in particular for the unsafe cases of the TTEthernet benchmarks, where the first phase penalizes considerably. On the other hand, we can observe that, under the assumption that we are not aware of the safety of the systems, our method shows to be the most scalable. The octagonal strategy tends to run out of time because the higher number of directions causes the generation of bigger and bigger abstract regions. In fact, we have verified that for these instances a spurious counter-example is never found. The same argument likely holds for PHAVer, as its dump shows that new symbolic states are always found. Not surprisingly, for QHA the performance is generally worse than for LHA.

In summary, template polyhedra coupled with our abstraction refinement technique are faster than the exact polyhedral reachability analysis. Noteworthy is how negligible is the time required in the final verification step on all instances. Our tool recomputes the whole abstraction after every refinement phase, as all our efforts have been strictly focused on implementing an efficient template refinement. The final time sets a lower bound for the verification time achievable by an incremental abstraction. Furthermore, we could observe that inferring small template sets plays an important role in the convergence of the whole analysis.

8 Conclusion

We have presented the first template refinement technique that iteratively derives template directions from spurious counterexamples. These directions eliminate all counterexamples that pass through the same switching sequence, independently of any time delays. These directions can refute further spurious paths, so that a small number of directions may suffice to show safety. This is supported by our experiments, which terminate with small templates in all cases. Our procedure can be implemented efficiently for LHA and QHA using convex optimization, and in principle it applies to every CHA. Our implementation outperforms polyhedral reachability (PHAVer), and yet has room for further substantial improvement since the abstraction is constructed from scratch at each iteration and could be made incremental [28, 32]. In terms of modeling power, extending template refinement to affine or general polynomial systems also brings further challenges, as the reachable regions lose the convexity property, thus requiring more powerful techniques for halfspace interpolation [2].

References

[1] GLPK (GNU linear programming kit), www.gnu.org/software/glpk
[2] Albarghouthi, A., McMillan, K.L.: Beautiful interpolants. In: CAV (2013)
[3] Alizadeh, F., Goldfarb, D.: Second-order cone programming. Math. Program. 95(1), 3–51 (2003)
[4] Alur, R., Dang, T., Ivancic, F.: Counter-example guided predicate abstraction of hybrid systems. In: TACAS (2003)
[5] Alur, R., Henzinger, T.A., Ho, P.: Automatic symbolic verification of embedded systems. In: RTSS. IEEE Computer Society (1993)
[6] Asarin, E., Dang, T., Maler, O., Testylier, R.: Using redundant constraints for refinement. In: ATVA (2010)
[7] Bogomolov, S., Frehse, G., Greitschus, M., Grosu, R., Pasareanu, C.S., Podelski, A., Strump, T.: Assume-guarantee abstraction refinement meets hybrid systems. In: HVC (2014)
[8] Bogomolov, S., Frehse, G., Grosu, R., Ladan, H., Podelski, A., Wehrle, M.: A box-based distance between regions for guiding the reachability analysis of SpaceEx. In: CAV (2012)
[9] Bogomolov, S., Herrera, C., Steiner, W.: Benchmark for verification of fault-tolerant clock synchronization algorithms. In: ARCH (2016)
[10] Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
[11] Bu, L., Zhao, J., Li, X.: Path-oriented reachability verification of a class of nonlinear hybrid automata using convex programming. In: VMCAI (2010)
[12] Chen, X., Ábrahám, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: RTSS (2012)
[13] Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: CAV (2013)
[14] Cimatti, A., Mover, S., Tonetta, S.: A quantifier-free SMT encoding of non-linear hybrid automata. In: FMCAD (2012)
[15] Clarke, E.M., Fehnker, A., Han, Z., Krogh, B.H., Ouaknine, J., Stursberg, O., Theobald, M.: Abstraction and counterexample-guided refinement in model checking of hybrid systems. Int. J. Found. Comput. Sci. 14 (2003)
[16] Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: CAV (2000)
[17] Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL (1977)
[18] Dang, T., Salinas, D.: Image computation for polynomial dynamical systems using the bernstein expansion. In: CAV (2009)
[19] Doyen, L., Henzinger, T.A., Raskin, J.: Automatic rectangular refinement of affine hybrid systems. In: FORMATS (2005)
[20] Dreossi, T., Dang, T., Piazza, C.: Parallelotope bundles for polynomial reachability. In: HSCC (2016)
[21] Frehse, G.: PHAVer: Algorithmic verification of hybrid systems past hytech. In: HSCC (2005)
[22] Frehse, G., Bogomolov, S., Greitschus, M., Strump, T., Podelski, A.: Eliminating spurious transitions in reachability with support functions. In: HSCC (2015)
[23] Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: CAV (2011)

[24] Frehse, G., Kateja, R., Guernic, C.L.: Flowpipe approximation and clustering in space-time. In: HSCC (2013)
[25] Guernic, C.L., Girard, A.: Reachability analysis of hybrid systems using support functions. In: CAV (2009)
[26] Henzinger, T.A.: The theory of hybrid automata. In: LICS (1996)
[27] Henzinger, T.A., Ho, P.H.: A note on abstract interpretation strategies for hybrid automata. In: International Hybrid Systems Workshop (1994)
[28] Henzinger, T.A., Jhala, R., Majumdar, R., Sutre, G.: Lazy abstraction. In: POPL (2002)
[29] Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: STOC (1995)
[30] Jha, S.K., Krogh, B.H., Weimer, J.E., Clarke, E.M.: Reachability for linear hybrid automata using iterative relaxation abstraction. In: HSCC (2007)
[31] Lamport, L.: A fast mutual exclusion algorithm. ACM Transactions on Computer Systems (TOCS) 5(1), 1–11 (1987)
[32] McMillan, K.L.: Lazy abstraction with interpolants. In: CAV (2006)
[33] MOSEK ApS: The MOSEK C optimizer API manual. Version 7.1 (Revision 53). (2015), docs.mosek.com/7.1/capi
[34] Ramana, M.V.: An exact duality theory for semidefinite programming and its complexity implications. Math. Program. 77 (1997)
[35] Ray, R., Gurung, A., Das, B., Bartocci, E., Bogomolov, S., Grosu, R.: XSpeed: Accelerating reachability analysis on multi-core processors. In: HVC (2015)
[36] Rockafellar, R.T.: Convex Analysis. Princeton University Press (1970)
[37] Sankaranarayanan, S., Dang, T., Ivancic, F.: Symbolic model checking of hybrid systems using template polyhedra. In: TACAS (2008)
[38] Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Scalable analysis of linear systems using mathematical programming. In: VMCAI (2005)
[39] Sassi, M.A.B., Testylier, R., Dang, T., Girard, A.: Reachability analysis of polynomial systems using linear programming relaxations. In: ATVA (2012)