Symbolic Model Checking of Hybrid Systems using Template Polyhedra

Sriram Sankaranarayanan[1], Thao Dang[2] and Franjo Ivančić[1]

1. NEC Laboratories America, Princeton, NJ, USA.
2. Verimag, Grenoble, France.
{srirams,ivancic}@nec-labs.com,thao.dang@imag.fr

Abstract. We propose techniques for the verification of hybrid systems using template polyhedra, i.e., polyhedra whose inequalities have fixed expressions but with varying constant terms. Given a hybrid system description and a set of template linear expressions as inputs, our technique constructs over-approximations of the reachable states using template polyhedra. Therefore, operations used in symbolic model checking such as intersection, union and post-condition across discrete transitions over template polyhedra can be computed efficiently using template polyhedra without requiring expensive vertex enumeration.
Additionally, the verification of hybrid systems requires techniques to handle the continuous dynamics inside discrete modes. We propose a new flowpipe construction algorithm using template polyhedra. Our technique uses higher-order Taylor series expansion to approximate the time trajectories. The terms occurring in the Taylor series expansion are bounded using repeated optimization queries. The location invariant is used to enclose the remainder term of the Taylor series, and thus truncate the expansion. Finally, we have implemented our technique as a part of the tool TimePass for the analysis of affine hybrid automata.

1 Introduction

Symbolic model checking of infinite state systems requires a systematic representation for handling infinite sets of states. Commonly used representations include difference matrices, integer/rational polyhedra, Presburger arithmetic, polynomials, nonlinear arithmetic and so on. Expressive representations can better approximate the underlying sets. However, the basic operations required for symbolic execution such as intersection, image (post-condition) and so on are harder to compute on such representations.

Convex polyhedra over reals (rationals) are a natural representation of sets of states for the verification of hybrid systems [15, 30, 2, 10–12]. However, basic algorithms required to manipulate polyhedra require worst-case exponential complexity. This fact has limited the practical usefulness of symbolic model checking tools based on polyhedra. Therefore, restricted forms of polyhedra such as *orthogonal polyhedra* [3] and *zonotopes* [11] are used to analyze larger systems at a level of precision that is useful for proving some properties of interest. Other techniques, such as *predicate abstraction*, use Boolean combinations of a fixed set of predicates $p_1, \ldots, p_m$, to represent sets of states [1, 16]. Such techniques enable the refinement of the representation based on counterexamples.

In this paper, we propose *template polyhedra* as a representation of sets of states. Given a set of *template expressions* $e_1, \ldots, e_m$, we obtain a family of *template polyhedra*, each of which is represented by the constraints $\bigwedge_i e_i \leq c_i$ [29]. As with predicate abstraction, our approach assumes that the template expressions are provided as an input to the reachability problem. We then use the family of polyhedra defined by the given template expressions as our representation for sets of states. The advantage of restricting our representation to a family of template polyhedra is that operations such as join, meet, discrete post-condition and time elapse can be performed efficiently, without requiring expensive vertex enumeration. Furthermore, our initial experience suggests that commonly used domains in software analysis such as *intervals* and *octagons* provide a good initial set of templates. This set can be further refined using simple heuristics for deriving additional expressions.

In order to analyze hybrid systems, we additionally require techniques to over-approximate the continuous dynamics at some location. This paper proposes a sound flowpipe construction technique based on a Taylor series approximation. Our approach works by solving numerous linear programs. The solutions to these linear programs correspond to bounds on the terms involved in the Taylor series expansion. The expansion itself is bounded by enclosing the remainder term using the location invariant. The flowpipe construction results in a series of template polyhedra whose disjunctions over-approximate the time trajectories.

Finally, we have implemented our methods in our prototype tool TimePass for verifying safety properties of affine hybrid systems. We use our tool to analyze many widely studied benchmark systems and report vastly improved performance on them.

Related Work

Hybrid systems verification is a challenge even for small systems. Numerous approaches have been used in the past to solve reachability problems: the HyTech tool due to Henzinger et al. uses polyhedra to verify rectangular hybrid systems [15]. More complex dynamics are handled using approximations. Kurzhanski and Variaya construct ellipsoidal approximations [17]; Mitchell et al. use level-set methods [20]; the d/dt system uses orthogonal polyhedra and face lifting [2]; Piazza et al. [22] propose approximations using constraint solving based on quantifier elimination over the reals along with Taylor series expansions to handle the continuous dynamics. Lanotte & Tini [18] present approximations based on Taylor series that can be made as accurate as possible, approaching the actual trajectories in the limit.

Girard uses zonotopes to construct flowpipes [11]. The PHAVer tool due to Frehse extends the HyTech approach by repeatedly subdividing the invariant region and approximating the dynamics inside each subdivision by piece-wise constant dynamics [10]. Tiwari [31] presents interesting techniques for proving safety by symbolically integrating the dynamics of the system. Symbolic techniques for proving unreachability without the use of an explicit flowpipe approximation [28, 32, 26, 23]. These techniques can handle interesting nonlinear systems beyond the reach of many related techniques.

The problem of flowpipe construction for template polyhedra has been studied previously by Chutinan & Krogh [5]. Their technique has been implemented as a part of the tool CheckMate [30]. Whereas the CheckMate approach solves global non convex optimization problems using gradient descent, our approach solves simple convex optimization problems to bound the coefficients of the Taylor series expansion. Furthermore, our technique can be extended to some nonlinear systems to construct ellipsoidal and polynomial flowpipes. The CheckMate technique simply yields a harder nonconvex optimization problem for these cases. On the other hand, our approach loses in precision due to its approximation of functions by Taylor polynomials; CheckMate, however, is more robust in this regard.

Template polyhedra are commonly used in static analysis of programs for computing invariants. Range analysis can be regarded as template polyhedra over expressions of the form $\pm x$ [7] . Similarly, the octagon domain due to Miné [19] uses template polyhedron of the form $\bigwedge x_i - x_j \leq c$. General template polyhedra were used as an abstract domain to represent sets of states by Sankaranarayanan et al. [29].

2 Preliminaries

Let $\mathcal{R}$ denote the set of reals, and $\mathcal{R}_+ = \mathcal{R} \cup \{\pm\infty\}$. A first order assertion $\varphi[x_1, \ldots, x_n]$, over the theory of reals, represents a set $[\![\varphi]\!] \subseteq \mathcal{R}^n$. A column vector, denoted $\langle x_1, \ldots, x_n \rangle$, is represented succinctly as $\boldsymbol{x}$. Capital letters A, B, C and X, Y, Z denote matrices; A_i denotes the i^{th} row of a matrix A. A linear function $f(x)$ is the inner product of vectors $\boldsymbol{c}^T\boldsymbol{x}$. Similarly, an affine function is represented as $\boldsymbol{c}^T\boldsymbol{x} + d$.

Polyhedra. A polyhedron is a conjunction of finitely many linear inequalities $\bigwedge_i e_i \leq c$, represented succinctly as $A\boldsymbol{x} \leq \boldsymbol{b}$, where A is a $m \times n$ matrix, $\boldsymbol{b}$ is a $m \times 1$ column vector and $\leq$ is interpreted entry-wise.

A linear program(LP) $P: \ \mathsf{max.}\ \boldsymbol{c}^T\boldsymbol{x}\ \mathsf{subject\ to}\ A\boldsymbol{x} \leq \boldsymbol{b}$ seeks to optimize a linear objective $\boldsymbol{c}^T\boldsymbol{x}$ over the convex polyhedron $[\![A\boldsymbol{x} \leq \boldsymbol{b}]\!]$. If $[\![A\boldsymbol{x} \leq \boldsymbol{b}]\!]$ is nonempty and bounded then the optimal solution always exists. LPs are solved using techniques such as Simplex [8] and interior point techniques [4]. The former technique is polynomial time for most instances, whereas the latter can solve LPs in polynomial time.

Vector Fields and Lie Derivatives. A *vector field* $\mathbf{D}$ over $\mathcal{R}^n$ associates each point $\boldsymbol{x} \in \mathcal{R}^n$ with a derivative vector $\mathbf{D}(\boldsymbol{x}) \in \mathcal{R}^n$. Given a system of differential equations of the form $\dot{x_i} = f_i(x_1, \ldots, x_n)$, we associate a vector field $\mathbf{D}(\boldsymbol{x}) = \langle f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}) \rangle$. A vector field is *affine* if the functions $f_1, \ldots, f_n$ are all affine in $\boldsymbol{x}$. For instance, the vector field $\mathbf{D}_0(x, y) : \langle x + y, x - 2y - 3 \rangle$ is affine.

Let $D(\boldsymbol{x}) = \langle f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\rangle$ be a vector field over $\mathcal{R}^n$. The *Lie derivative* of a continuous and differentiable function $h : \mathcal{R}^n \mapsto \mathcal{R}$ is $\mathcal{L}_D(f) = (\nabla h) \cdot D(\boldsymbol{x}) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} \cdot f_i(\boldsymbol{x})$. The Lie derivative of the function $x + 2y - 2$ over the vector field $\mathbf{D}_0(x, y)$ shown above is given by

$$\mathcal{L}_{D_0}(x + 2y - 2) = 1 \cdot (x + y) + 2 \cdot (x - 2y - 3) = 3x - 3y - 6\,.$$

Hybrid Systems. To model hybrid systems we use hybrid automata [14].

Definition 1 (Hybrid Automaton). *A* hybrid automaton $\Psi : \langle n, \mathbf{L}, \mathcal{T}, \Theta, \mathbf{D}, \mathbf{I}, \ell_0\rangle$ *consists of the following components:*

- *n is the number of continuous variables. These variables are denoted by the set $V = \{x_1, \ldots, x_n\}$.*
- *$\mathbf{L}$, a finite set of locations; $\ell_0 \in L$ is the initial location;*
- *$\mathcal{T}$, a set of (discrete) transitions. Each transition $\tau : \langle \ell_1 \rightarrow \ell_2, \rho_\tau\rangle \in \mathcal{T}$ consists of a move from $\ell_1 \in \mathbf{L}$ to $\ell_2 \in \mathbf{L}$, and an assertion ρ_τ over $V \cup V'$, representing the transition relation;*
- *Assertion Θ, specifying the* initial *values of the continuous variables;*
- *$\mathbf{D}$, mapping each $\ell \in \mathbf{L}$ to a* vector field *$\mathbf{D}(\ell)$, specifying the continuous evolution in location ℓ;*
- *$\mathbf{I}$, mapping each $\ell \in \mathbf{L}$ to a* location invariant, *$\mathbf{I}(\ell)$.*

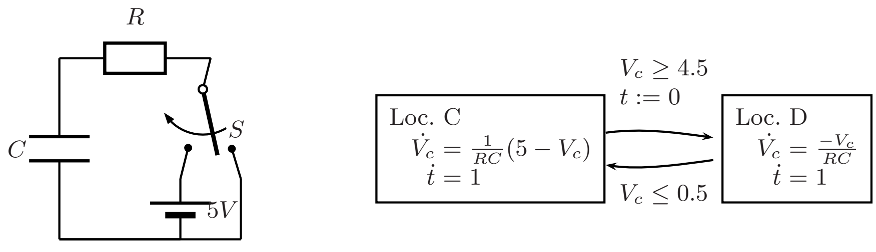
A *computation* of a hybrid automaton is an infinite sequence of states $\langle l, \boldsymbol{x}\rangle \in \mathbf{L} \times \mathcal{R}^n$ of the form $\langle l_0, \boldsymbol{x}_0\rangle$, $\langle l_1, \boldsymbol{x}_1\rangle$, $\langle l_2, \boldsymbol{x}_2\rangle$, $\ldots$, such that initially $l_0 = \ell_0$ and $\boldsymbol{x}_0 \in [\![\Theta]\!]$; and for each consecutive state pair $\langle l_i, \boldsymbol{x}_i\rangle$, $\langle l_{i+1}, \boldsymbol{x}_{i+1}\rangle$, satisfies one of the *consecution* conditions:
Discrete Consecution: there exists a transition $\tau : \langle \ell_1, \ell_2, \rho_\tau\rangle \in \mathcal{T}$ such that $l_i = \ell_1$, $l_{i+1} = \ell_2$, and $\langle \boldsymbol{x}_i, \boldsymbol{x}_{i+1}\rangle \models \rho_\tau$, or
Continuous Consecution: $l_i = l_{i+1} = \ell$, and there exists a time interval $[0, \delta)$, $\delta > 0$, and a *time trajectory* $\tau : [0, \delta] \mapsto \mathcal{R}^n$, such that τ evolves from $\boldsymbol{x}_i$ to $\boldsymbol{x}_{i+1}$ according to the vector field at location ℓ, while satisfying the location condition $\mathbf{I}(\ell)$. Formally,

1. $\tau(0) = \boldsymbol{x_1},\ \tau(\delta) = \boldsymbol{x_2},$ and $(\forall\ t \in [0, \delta]),\ \tau(t) \in [\![\mathbf{I}(\ell)]\!]$,
2. $(\forall t \in [0, \delta)),\ \frac{d\tau}{dt} = \mathbf{D}(\ell)|_{\boldsymbol{x}=\tau(t)}$.

Definition 2 (Affine Hybrid Automaton). *A hybrid automaton Ψ is* affine *if the initial condition, location invariants and transition relations are all represented by a conjunction of linear inequalities; and furthermore, the dynamics at each location $\mathbf{D}(\ell)$ is an* affine *vector field.*

The rest of the paper focuses solely on affine systems. However, our results also extend to the non-affine case.

Example 1. Affine hybrid systems are used to represent a variety of useful systems. Consider the oscillator circuit shown in Figure 1(a). The circuit consists of a capacitor that may be charged or discharged using a voltage triggered switch S that is controlled by the voltage across the capacitor V_c. The corresponding



(a) **(b)**

Fig. 1. An oscillator circuit (left) and its affine hybrid automaton model.

affine hybrid automaton H has two modes C and D corresponding to the charging and discharging; and two variables V_c modeling the voltage of the capacitor and t modeling time. Switching between each mode takes place when the capacitor has charged (or discharged) to 90% (10%) of its final charge. We assume the mode invariants $\mathbf{I}(C) :\ 0 \leq V_c \leq 4.5$ and $\mathbf{I}(D) :\ 0.5 \leq V_c \leq 5$.

The *post-condition* and *time elapse* operations are the two fundamental primitives for over-approximating the reachable sets of a given hybrid automaton. Given an assertion φ over the continuous variables, its *post-condition* across a transition $\tau :\ \langle \ell, m, \rho \rangle$ is given by $\mathsf{post}(\varphi, \tau)[\boldsymbol{y}] :\ (\exists\, \boldsymbol{x})\ (\varphi(\boldsymbol{x})\ \wedge\ \rho(\boldsymbol{x}, \boldsymbol{y}))$. The post-condition of a polyhedron is also polyhedral. It is computed using intersection and existential quantification.

Similarly, given an assertion φ, the set of possible time trajectories inside a location ℓ with invariant $\mathbf{I}(\ell)$ and dynamics $\mathbf{D}(\ell)$ is represented by its *time elapse* $\psi :\ \mathsf{timeElapse}(\varphi, \langle \mathbf{D}, \mathbf{I} \rangle)$. However, for affine hybrid systems, the time elapse of a polyhedron need not be a polyhedron. Therefore, the time elapse operator is hard to compute and represent exactly. It is over-approximated by the union of a set of polyhedra. Such an approximation is called a *flowpipe approximation*.

Using post-conditions and time elapse operators as primitives, we can prove unreachability of unsafe states using a standard forward propagation algorithm. Such an algorithm is at the core of almost all safety verification tools for hybrid systems [15, 2, 30, 10]

Template Polyhedra. The goal of this paper is to implement symbolic model checking on hybrid systems using template polyhedra. We now present the basic facts behind template polyhedra, providing algorithms for checking inclusion, intersection, union and post-condition. Additional details and proofs are available from our previous work [29].

A *template* is a set $H = \{h_1(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x})\}$ of linear expressions over $\boldsymbol{x}$. We represent a template as an $m \times n$ matrix H, s.t. each row H_i corresponds to the linear expression h_i. Given a template, a family of template polyhedra may be obtained by considering conjunctions of the form $\bigwedge_i h_i(\boldsymbol{x}) \leq c_i$. Each polyhedron in the family may be obtained by choosing the constant coefficients $c_1, \ldots, c_m$.

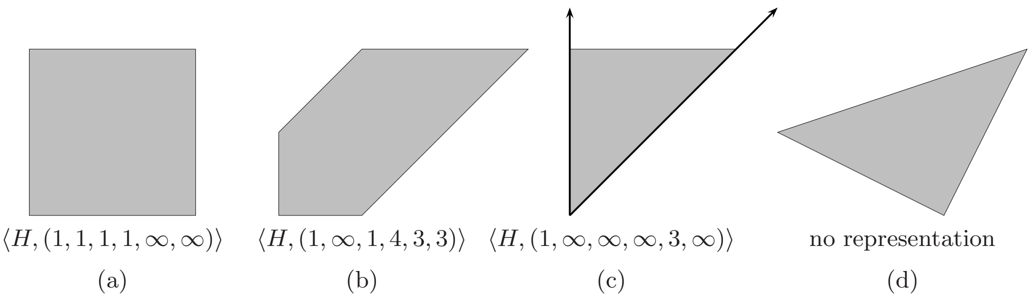


Fig. 2. Polyhedra (a), (b) and (c) are template instances for the template H shown in Example 2, whereas (d) is not.

Definition 3 (Template Polyhedron). *A* template polyhedron *over a template H is a polyhedron of the form $H\boldsymbol{x} \leq \boldsymbol{c}$, wherein $\boldsymbol{c} \in \mathcal{R}_+^m$. Such a polyhedron will be represented as $\langle H, \boldsymbol{c}\rangle$.*

Example 2. Consider the template $H = \{x, -x, y, -y, y - x, x - y\}$. The unit square $-1 \leq x \leq 1 \wedge -1 \leq y \leq 1$ may be represented by the template polyhedron $\langle H, (1, 1, 1, 1, \infty, \infty)\rangle$. Figure 2 shows three polyhedra that are instances, and one that is not.

Let $\boldsymbol{c}_1 \leq \boldsymbol{c}_2$ signify that for each row $i \in [1, |\boldsymbol{c}_1|]$, $\boldsymbol{c}_{1i} \leq \boldsymbol{c}_{2i}$.

Lemma 1. *If $\boldsymbol{c}_1 \leq \boldsymbol{c}_2$ then $\langle H, \boldsymbol{c}_1\rangle \subseteq \langle H, \boldsymbol{c}_2\rangle$. However, the converse need not hold.*

Example 3. The set $C : \; x = 0 \wedge y = 0$ may be represented using the template $H = \{x, -x, y, -y, x + y\}$ using the instances vectors $\boldsymbol{c} : \langle 0, 0, 0, 0, 0\rangle$, $\boldsymbol{c}_1 : \langle 0, 0, 0, 0, 100\rangle$, $\boldsymbol{c}_2 : \langle -10, 0, 0, 0, 0\rangle$, and $\boldsymbol{c}_3 : \langle 0, -100, 0, 0, 0\rangle$. In each case $\langle H, \boldsymbol{c}_i\rangle \subseteq \langle H, \boldsymbol{c}\rangle$. However $\boldsymbol{c}_i \not\leq \boldsymbol{c}$. Intuitively, "fixing" any four of the rows to 0 renders the remaining constraint row redundant.

Consider a region $C \subseteq \mathcal{R}^n$ and template H. There exists a smallest template polyhedron $\langle H, \boldsymbol{c}\rangle$, with the least instance vector $\boldsymbol{c}$, that over-approximates C, denoted $\boldsymbol{c} = \alpha_H(C)$. Furthermore, for any template polyhedra $\langle H, \boldsymbol{d}\rangle$ that over-approximates C, $\boldsymbol{c} \leq \boldsymbol{d}$. Each component c_i of $\alpha_H(C)$ may be computed using the optimization problem $c_i : \; \mathsf{max.}\; h_i(\boldsymbol{x}) \;\mathsf{s.t.}\; \boldsymbol{x} \in C$. Note that if C is a polyhedron, then its best over-approximation according to a template H is obtained by solving $|H|$ linear programs.

Lemma 2. *For any closed set $C \subseteq \mathcal{R}^n$, the polyhedron $H\boldsymbol{x} \leq \alpha_H(\boldsymbol{c})$ is the* smallest *template polyhedron that includes C.*

Example 4. Let $H = \{x, -x, y, -y\}$ be a template. Consider the set $C : \; (x^2 + y^2 \leq 1)$ of all points inside the unit circle. The smallest template polyhedron containing C is the unit square that may be represented with the instance vector $\langle 1, 1, 1, 1\rangle$. Additionally, if the expressions $x + y, x - y, -x - y, x + y$ are added to the set H, the smallest template polyhedron representing C is the octagon inscribed around the circle.

It is algorithmically desirable to have a unique representation of each set by a template polyhedron. Given a template polyhedron $\langle H, \boldsymbol{c}\rangle$, its *canonical form* is given by $\mathsf{can}_H(\boldsymbol{c}) = \alpha_H(H\boldsymbol{x} \leq \boldsymbol{c})$. An instance vector is *canonical* iff $\boldsymbol{c} = \mathsf{can}_H(\boldsymbol{c})$.

Lemma 3. *(a)* $\langle H, \boldsymbol{c}\rangle \equiv \langle H, \boldsymbol{d}\rangle$ *iff* $\mathsf{can}_H(\boldsymbol{c}) = \mathsf{can}_H(\boldsymbol{d})$*, and (b)* $\langle H, \boldsymbol{c}\rangle \subset \langle H, \boldsymbol{d}\rangle$ *iff* $\mathsf{can}_H(\boldsymbol{c}) < \mathsf{can}_H(\boldsymbol{d})$*.*

Thus, canonicity provides an unique representation of template polyhedra. Any representation can be converted into a canonical representation in polynomial time using optimization problems.

The *union* of $\langle H, \boldsymbol{c}_1\rangle$ and $\langle H, \boldsymbol{c}_2\rangle$ (written $\boldsymbol{c}_1 \sqcup \boldsymbol{c}_2$) is defined as $\boldsymbol{c} = \max(\boldsymbol{c}_1, \boldsymbol{c}_2)$, where max denotes the entry-wise minimum. Similarly, intersection of two polyhedra $\boldsymbol{c}_1, \boldsymbol{c}_2$ is represented by $\boldsymbol{c} = \min(\boldsymbol{c}_1, \boldsymbol{c}_2)$.

Given a template polyhedron $P_0 :\ \langle J, \boldsymbol{c}\rangle$, and a discrete transition relation τ, we wish to compute the smallest template polyhedron $P_1 :\ \langle H, \boldsymbol{d}\rangle$ that over-approximates the post-condition $\mathsf{post}(P_0, \tau)$. Note that the templates J and H need not be identical. The *post-condition* $\boldsymbol{d} :\ \mathsf{post}_H(\langle J, \boldsymbol{c}\rangle, \tau)$ is computed by posing an optimization query for each $\boldsymbol{d}_i$: $\mathsf{max.}\ H_i\boldsymbol{y}\ \mathsf{subj.\ to}\ J\boldsymbol{x} \leq \boldsymbol{c}\ \wedge\ \rho_\tau(\boldsymbol{x}, \boldsymbol{y})$. The resulting $\boldsymbol{d}$ is always guaranteed to be canonical.

Lemma 4. *The polyhedron* $\mathsf{post}_H(P_0, \tau)$ *is the smallest template polyhedron containing* $\mathsf{post}(P_0, \tau)$*.*

In program analysis, template polyhedra with a fixed set of template have been used previously. For instance, given variables $x_1, \ldots, x_n$, intervals are obtained as template polyhedra over the set $H_I = \{x_1, -x_1, x_2, \ldots, x_n, -x_n\}$ [7]. Similarly, the *octagon* domain is obtained by considering the template expressions $H_O = H_I \cup \{\pm x_i \pm x_j | 1 \leq i < j \leq n\}$ [19]. Other domains based on template polyhedra include the *octahedron* domain consisting of all linear expressions involving the variables $x_1, \ldots, x_n$ with unit coefficients [6].

3 Flowpipe Construction

We now consider flowpipe construction techniques to over-approximate the time trajectories of affine differential equations. An instance of flowpipe construction problem: $\langle H, \boldsymbol{c}_0, \mathsf{inv}, \mathbf{D}, \delta\rangle$ consists of the template H, an initial region $\langle H, \boldsymbol{c}_0\rangle$, the location invariant $\langle H, \mathsf{inv}\rangle$ and an affine vector field $\mathbf{D}$ representing the dynamics and a *time step* $\delta \geq 0$. We assume that $\langle H, \mathsf{inv}\rangle$ and $\langle H, \boldsymbol{c}_0\rangle$ are nonempty and bounded polyhedra.

Example 5. Consider the oscillator circuit model from Example 1. An instance consists of a template $H = \{v, -v, t, -t, v - t, t - v\}$, initial condition $v \in [0, 0.1], t = 0$ and location invariant $v \in [0, 5],\ t \in [0, 100]$.

Let $\mathfrak{F}(t)$ denote the set of states reachable, starting from $\langle H, \boldsymbol{c}_0 \rangle$, at some time instant $t \geq 0$. Similarly, $\mathfrak{F}[t, t+\delta)$ denotes the set of reachable states for the time interval $[t, t+\delta)$.

Formally, we wish to construct a series of flowpipe segments

$$\langle H, \boldsymbol{d}_0 \rangle, \langle H, \boldsymbol{d}_1 \rangle, \langle H, \boldsymbol{d}_2 \rangle, \ldots, \langle H, \boldsymbol{d}_N \rangle, \ldots$$

such that each segment $\boldsymbol{d}_j$ over-approximates $\mathfrak{F}[j\delta, (j+1)\delta)$. There are two parts to our technique:

Flowpipe Approximation: Approximate $\mathfrak{F}[0, \delta)$ given $\langle H, \boldsymbol{c}_0 \rangle$.
Set Integration: Given an approximation $\mathfrak{F}[i\delta, (i+1)\delta)$, approximate the next segment $\mathfrak{F}[(i+1)\delta, (i+2)\delta)$.

Together, they may be used to incrementally construct the entire flowpipe.
Set Integration. By convention, the j^{th} order Lie derivative of a function f is written $f^{(j)}$. Let $f:\ \boldsymbol{c}^T\boldsymbol{x}$ be a linear function. By convention, we denote its j^{th} order derivative as $\boldsymbol{c}^{(j)}\boldsymbol{x}$.

Definition 4 (Taylor Series). *Let h be a continuous function and differentiable at least to order $m+1$. It follows that*

$$h(t) = h(0) + h^{(1)}(0)t + h^{(2)}(0)\frac{t^2}{2!} + \cdots + h^{(m)}(0)\frac{t^m}{m!} + h^{(m+1)}(\theta)\frac{t^{m+1}}{(m+1)!},$$

where $\theta \in [0, t)$. The last term of the series is known as the remainder.

Let $S_k:\ \langle H, \boldsymbol{d}_k \rangle$ be an over-approximation of $\mathfrak{F}[k\delta, (k+1)\delta)$. We wish to compute an approximation S_{k+1} for the time interval $[(k+1)\delta, (k+2)\delta)$. In other words, we require an upper bound for the value of each template row $H_i\boldsymbol{x}$. Let $\boldsymbol{x}(t)$ be the state at time instant t. Using a Taylor series expansion, we get:

$$H_i\boldsymbol{x}(t+\delta) = H_i\boldsymbol{x}(t) + \cdots + \frac{\delta^m}{m!}H_i^{(m)}\boldsymbol{x}(t) + \frac{\delta^{m+1}}{(m+1)!}H_i^{(m+1)}\boldsymbol{x}(t+\theta), \quad (1)$$

where $0 \leq \theta < \delta$. Note that the first m terms are functions of $\boldsymbol{x}(t)$, whereas the remainder term, is a function of $\boldsymbol{x}(t+\theta)$. The exact value of θ is not known and is conservatively treated as a nondeterministic input. In other words, we may write $H_i\boldsymbol{x}(t+\delta)$ as a sum of two expressions $H_i\boldsymbol{x}(t+\delta) = \boldsymbol{g_i}^T\boldsymbol{x}(t) + \boldsymbol{r_i}^T\boldsymbol{x}(t+\theta)$, wherein $\boldsymbol{g_i}$ represents the sum of the first m terms of the Taylor series and $\boldsymbol{r_i}$ represents the remainder term.

Assuming $t \in [j\delta, (j+1)\delta)$, we have $\boldsymbol{x}(t) \in S_k$. Therefore, an upper bound on $\boldsymbol{g_i}$ is obtained by solving the following LP:

$$g_i^{max} = \mathsf{max.}\ \boldsymbol{g_i}^T\boldsymbol{x}\ \mathsf{subj.to.}\ \boldsymbol{x} \in S_k \quad (2)$$

Similarly, even though the remainder term cannot be evaluated with certainty, we know that $\boldsymbol{x}(t+\theta) \in \langle H, \mathsf{inv} \rangle$. A bound on $\boldsymbol{r_i}\boldsymbol{x}(t+\delta)$ is, therefore, obtained by solving the optimization problem

$$r_i^{max} = \mathsf{max.}\ \boldsymbol{r_i}^T\boldsymbol{y}\ \mathsf{subj.to}\ \boldsymbol{y} \in \langle H, \mathsf{inv} \rangle \quad (3)$$

The overall bound on $H_i\boldsymbol{x}(t+\delta)$ is $g_i^{max}+r_i^{max}$. Finally, the over-approximation S_{k+1} is obtained by computing $g_i^{max}+r_i^{max}$ for each template row $i \in [1, |H|]$.

Note that in the optimization problem above, the time step δ is an user-input constant, each Lie-derivative $g_i^{(m)}$ is affine and S_k is a template polyhedron. As a result, the optimization problems for affine vector fields are linear programs.

Example 6. Following Example 5, consider a flowpipe segment $v \in [0, 0.2] \wedge t \in [0, 0.1]$ by $\delta = 0.1$, according to the differential equation $\dot{v} = \frac{5-v}{2}, \dot{t} = 1$. The first row of the template is $H_1 : v$. The first 6 Lie derivatives of H_1 are tabulated below:

0	1	2	3	4	5	6
v	$\frac{5-v}{2}$	$\frac{-5+v}{4}$	$\frac{5-v}{8}$	$\frac{-5+v}{16}$	$\frac{5-v}{32}$	$\frac{-5+v}{64}$

Following, Eq. 1, we use exact arithmetic to obtain

$$\begin{aligned} v(t+\delta) &= v + \tfrac{5-v}{2}\delta + \tfrac{-5+v}{4}\tfrac{\delta^2}{2!} + \cdots + \tfrac{5-v}{32}\delta^5 5! + \tfrac{-5+v(\theta)}{64}\tfrac{\delta^6}{6!} \\ &\sim \underbrace{0.951229424479167v(0) + 0.24385288020833}_{\boldsymbol{g}_0} + \underbrace{0.131 \times 10^{-7}v(\theta)}_{r_0} \end{aligned}$$

Now observing that $v(0) \in [0, 0.2]$, we obtain $g_0^{\max} = 0.4341$ (upto 4 decimal places). Similarly, using the location invariant $v(\theta) \in [0, 5]$, we obtain $r_0^{max} = 0.131 \times 10^{-8} \times 5$. As a result, we obtain a bound $v(t + 0.1) \leq 0.4341$ (upto 4 decimal digits). Repeating this process for every template row gives us the required flowpipe approximation for the segment $[0.1, 0.2)$.

Flowpipe Approximation

We now seek an approximation $\langle H, \boldsymbol{d}_0 \rangle$ for $\mathfrak{F}[0, \delta)$. Therefore, for each template row H_i, we wish to bound the function $H_i\boldsymbol{x}$ as an univariate polynomial of degree $m + 1$ over the time interval $[0, \delta)$. Let $a_{i,j}$, $0 \leq j \leq m$ be the result of the optimization $a_{i,j} = \mathsf{max}\ \frac{H_i^{(j)}(\boldsymbol{x})}{j!}\ \mathsf{subj.to.}\ \boldsymbol{x} \in \langle H, \boldsymbol{c}_0 \rangle$ and $a_{i,m+1} = \mathsf{max}\ \frac{H_i^{(m+1)}(\boldsymbol{y})}{m+1!}\ \mathsf{subj.to.}\ \boldsymbol{y} \in \langle H, \mathsf{inv} \rangle \cdot$

Each optimization problem is an LP and can be solved efficiently. Consider the polynomial $p_i(t) = \sum_{j=0}^{m} a_{ij}t^j + a_{i,m+1}t^{m+1}$.

Lemma 5. *For $t \geq 0$ and $\boldsymbol{x} \in \langle H, \boldsymbol{c}_0 \rangle$, $H_i\boldsymbol{x}(t) \leq p_i(t)$.*

$$\begin{aligned} H_i\boldsymbol{x}(t) &= H_i\boldsymbol{x}(0) + tH_i^{(1)}\boldsymbol{x}(0) + \cdots + t^m\frac{H_i^{(m)}(\boldsymbol{x}(0))}{m!} + t^{m+1}\frac{H_i^{(m+1)}(\boldsymbol{x}(\theta))}{(m+1)!} \\ &\leq a_{i0} + a_{i1}t + \cdots + a_{im}t^m + a_{i,m+1}t^{m+1} \quad \because\ \frac{H_i^{(j)}\boldsymbol{x}(0)}{j!} \leq a_{ij} \text{ and } t \geq 0 \\ &\leq p_i(t) \end{aligned}$$

The required bound for the function $H_i\boldsymbol{x}$ for the time interval $t \in [0, \delta)$ may now be approximated by maximizing the univariate polynomial $p_i(t)$ over the interval $[0, \delta)$. The maximum value of an univariate polynomial p in a time interval $[T_1, T_2]$ may be computed by evaluating the polynomial at the end points T_1, T_2 and the roots (if any) of its derivative p' lying in that interval. The maxima in the interval is guaranteed to be achieved at one of these points.

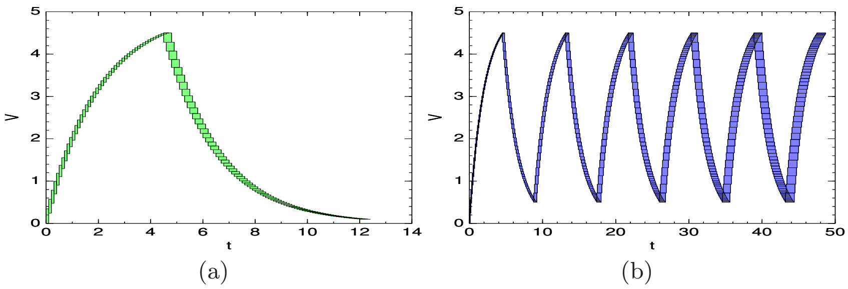


Fig. 3. Flowpipes for Example 1: (a) one complete charge/discharge cycle and (b) the time interval $[0, 49]$.

Example 7. Consider the problem instance shown in Example 5. We wish to compute an over-approximation of $\mathfrak{F}[0, 0.1)$ given $v(0) \in [0, 0.1]$ and $t(0) = 0$. We consider a bound $H_1 : v$ over $t \in [0, 0.1)$. Example 6 shows the lie derivatives. The following table shows the bounds $a_1, \ldots, a_6$ corresponding to the initial condition and invariant regions (accurate to 4 decimal places).

0	1	2	3	4	5	6
0.1	2.5	−0.6125	0.1058	−0.01276	0.0013	−0.00106

As a result, we have that $v(t) \leq -0.00106t^6 + 0.0013t^5 - 0.01276t^4 + \cdots + 0.1$ for all $t \in [0, 0.1)$. This polynomial is increasing in this range and has its maximum value at $t = 0.1$. This yields a bound $v \leq 0.3439$ for the segment $\mathfrak{F}[0, 0.1)$. Similarly, we can compute bounds for all the rows in the template.

Thus, given an instance of the flowpipe problem, we compute an initial flowpipe segment $\langle H, \boldsymbol{d}_0 \rangle \supseteq \mathfrak{F}[0, \delta)$ by computing univariate polynomials, one per template row, that upper bound the Taylor series and in turn finding the maxima of these polynomials. This initial flowpipe segment is then advanced by using set integration. Following this scheme, Fig. 3 shows the flowpipe constructed for the instance in Example 5. Let $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_N$ be the results of the flowpipe construction on a given instance.

Theorem 1. *The disjunction* $\bigvee_{i=0}^{N} \langle H, \boldsymbol{d}_i \rangle$ *contains all the time trajectories starting from* $\langle H, \boldsymbol{c}_0 \rangle$ *and evolving according to* $\mathbf{D}$ *inside* $\langle H, \mathsf{inv} \rangle$.

Termination. In theory, the flowpipe construction produces infinitely many segments. However, we may stop this process if the flowpipe "exits" the invariant, i.e, $\langle H, \boldsymbol{d}_N \rangle \cap \langle H, \mathsf{inv} \rangle = \emptyset$ for some $N > 0$; or "cycles" back into itself, i.e., $\langle H, \boldsymbol{d}_N \rangle \subseteq \langle H, \boldsymbol{d}_j \rangle$ for $j < N$. The flowpipe construction can be stopped upon encountering a cycle since each subsequent segment will lie inside a previously encountered segment.

Extensions. Our technique is directly applicable to cases where the templates may consist of nonlinear functions and the dynamics may be nonlinear. In each

Table 1. Optimization problems for flowpipe construction.

Dynamics ($\mathbf{D}$)	Template (h_i)	Invariants ($\mathbf{I}$)	Optimization Problem.
Affine	Linear	Polyhedral	Linear Programming
Affine	Ellipsoidal	Polyhedral	Quadratic Programming [4]
Polynomial	Polynomial	Semi-Algebraic	Sum-of-Squares Optimization (SOS) [21]
Continuous	Continuous	Rectangular	Interval Arithmetic [13]

case, we encounter different types of optimization problems with differing objectives and constraints. Table 1 summarizes the different optimization problems that are encountered.

Matrix Exponentiation. Set integration can also be computed using *matrix exponentiation* for affine systems [5]. In this approach, we compute a matrix exponential $T = e^{A\delta}$, corresponding to the dynamics $\mathbf{D}(\boldsymbol{x}) = A\boldsymbol{x}$. Given the initial segment S_0, approximating $\mathfrak{F}[0, \delta)$, we may compute successive sets as $S_{i+1} = TS_i$. However, computing this transformation requires an expensive vertex representation of S_i. On the other hand, our approach works purely on the constraint representation of template polyhedra using LPs for set integration.

Location Invariant Strengthening. The location invariant bounds the remainder term in our construction. Therefore, tighter bounds on the remainder can result from stronger location invariants. Such a strengthening can be computed prior to each flowpipe construction using a *policy iteration* technique. Using invariant strengthening, each flowpipe construction instance can be performed more accurately using a better bound for the location invariant. Curiously, a stronger invariant region may result in fewer flowpipe segments and quicker termination, thus reducing the overall time taken by our technique. The details of the invariant strengthening technique appear elsewhere [27].

4 Experiments

Our prototype tool TimePass implements the techniques described in this paper using template polyhedra for safety verification.

Template Construction. A larger set of template expressions provides a richer representation of template polyhedra. However, the size of each LP instance encountered is linear in the number of templates. Therefore, too many templates impacts performances.

Our template construction strategy uses two basic sources of template expressions: (a) Fixed templates such as boxes and octagons; and (b) Expressions occurring in the hybrid system description. Fixed templates used include *box templates* which include the expressions $\pm x_i$, for each continuous variable x_i in the system, and *octagon templates* of the form $\pm x_i \pm x_j$ for all $x_i \neq x_j$.

Additionally, we enrich templates by computing their Lie derivatives. This process is important since the key flowpipe construction steps involve finding

Table 2. Performance of our tool on hybrid systems benchmarks. All timings are in seconds and memory in MBs. Legend: **Inv. Str.**: Invariant Strengthening, **H**: Template size, δ: step size, **T**:Time, **Mem:** memory, **Prf?**: Property proved.

Name	Description	Size/Params					w/o Inv. Str.		Inv. Str.	
		#Var	#Loc	#Trs	$\|H\|$	δ	T	Prf?	T	Prf?
focus	[24]	2	2	1	28	0.2	0	Y	0	Y
reigen	-	3	2	1	54	0.2	0.1	Y	0.2	Y
flow	-	3	2	1	54	0.2	0.1	Y	0.1	Y
convoi	-	5	1	1	90	0.2	10	Y	18	Y
therm	[1]	2	3	4	28	0.05	1.1	Y	1.2	Y
nav01	Benchmark [9]	4	8	18	64	0.2	260	Y	22	Y
nav02	-	4	8	18	64	0.2	362	Y	23	Y
nav03	-	4	8	18	64	0.2	390	Y	20	Y
nav04	-	4	8	18	64	0.2	1147	Y	18	Y
nav05	-	4	8	18	64	0.1	7	N	513	Y
nav06	-	4	8	18	64	0.2	45	N	1420	N
nav07	-	4	15	39	64	0.2	1300	N	572	Y
nav08	-	4	15	39	64	0.2	139	N	572	Y

bounds on the Lie derivatives of the template rows (and their convex combinations). Therefore, tracking bounds for such rows as part of the template can lead to tighter bounds. The *eigenvectors* corresponding to the real eigenvalues of the RHS matrix of the differential equations also form an interesting set of template expressions. The Lie derivatives of such expressions yield back the original expression upto a constant scale factor. As a result, the Taylor polynomials for such expressions can be computed precisely without truncation.

Numerical Issues. It is possible to implement most of the algorithms described in this paper using exact arithmetic. In our experience, however, exact arithmetic LP solvers exhibit large performance overheads. Hence, our tool primarily uses a floating point implementation of the simplex algorithm. The LP solution can then be verified using the Karush-Kuhn-Tucker (KKT) conditions to lie within an error tolerance bound ($\sim 10^{-7}$). Failing, the error tolerance bounds, the verification may be performed an exact arithmetic simplex implementation. All our experiments, however, were performed with a floating point solver.

Parameters. The time step δ for flowpipe construction has the largest impact on the performance. A large time step speeds up the convergence but results in a coarser approximation. In general, the ideal choice of time step is hard to realize. Therefore, we use a trial-and-error approach to successively reduce/increase δ to arrive at a large enough time step that proves the property of interest.

Experiments. Table 2 shows the performance of our tool on some hybrid systems benchmarks consisting of small but complex systems, designed to test the accuracy of the flowpipe construction and its propagation. A detailed description is available elsewhere [24, 9]. We report on our performance with and without the use of invariant strengthening. Our tool successfully proves safety for a most of

Table 3. Flowpipe results on systems with many variables. Note: Timeout is set to 1h.

n	#Sys	$\|H\|$	#Loc	#Trs	Time(sec)			Mem (Mb)			Proved?
					Avg.	Max	Min	Avg.	Max	Min	
10	10	80	7	6	21	52	1	5	7	3	10/10
20	10	160	14	13	30	91	8	11	13	5	10/10
40	10	320	21	20	192	975	44	105	88	126	10/10
80	6	640	29	28	1386	$> 1h$	420	700	743	608	5/6

the benchmarks instances. Note that invariant strengthening plays a key role, especially for the larger examples. As expected, the use of invariant strengthening vastly reduces the time taken to prove many properties. Our timings on the other examples are quite competitive with those of PHaVer [10] and HSolver [25]. Our approach also provides the first known verification for benchmarks nav05-nav08. Figure 4 depicts the reach sets computed by our tool for the NAV05 and the NAV08 benchmark examples.

We stress test our flowpipe construction on systems with a large number of variables. Since we do not have access to meaningful models in a suitable format, we use a scheme for generating a family of systems with known behaviors and verify these using our tool. Each system H_n has $n > 0$ variables. It has a *primary mode* ℓ_0, and secondary modes $\ell_1, \ldots, \ell_m$.

The dynamics at location ℓ_0 are $\boldsymbol{x}' = A(\boldsymbol{x}-\boldsymbol{t})$, where A is an invertible matrix with negative real eigenvalues and $\boldsymbol{t}$ is a target point. These dynamics ensure that $\boldsymbol{t}$ is a stable equilibrium point. The mode invariant $\mathbf{I}(\ell_0)$ is a hypercube $|\boldsymbol{x}| \leq \boldsymbol{t}+\boldsymbol{\epsilon}$ for a parameter $\epsilon > 0$. To generate A, we choose negative eigenvalues Λ at random, and compute $A = X^{-1}\Lambda X$ for invertible X.

The secondary modes consist of regions around the corners of the primary mode hypercube, which are unreachable from the interior of the primary mode. The initial location is ℓ_0 and $\Theta : \boldsymbol{x} \in [-\epsilon_0, \epsilon_0]$. We seek to verify that the secondary modes are unreachable. We first generate many instances with varying dynamics A, target vectors $\boldsymbol{t}$ and number of secondary modes. We also fix $\epsilon = 1$ and $\epsilon_0 = 0.1$. Table 3 shows the results of running our tool on systems of varying sizes. To minimize the run-time overhead especially for large systems, these experiments were carried out without using policy iteration to strengthen the invariant region. It clearly demonstrates the scalability of our approach. Also, it demonstrates that our flowpipe is accurate enough to prove a vast majority of instances.

5 Conclusion

Template polyhedra are shown to be an effective tool for the verification of hybrid systems by avoiding the need to perform costly vertex enumerations using template polyhedra. In the future, we hope to study heuristics for choosing template expressions that would enable application of our technique to the counterexam-

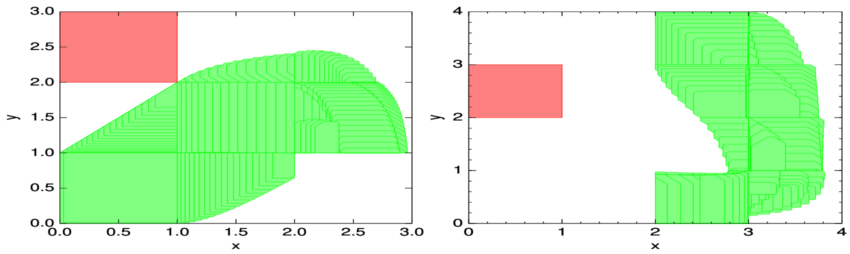


Fig. 4. Reach sets (projected over x, y) along with the unsafe cell for the NAV05 (left) and NAV08 (right) benchmarks.

ple guided refinement (CEGAR) framework. We hope to extend our techniques to nonlinear systems and apply it to more meaningful examples.

References

1. Alur, R., Dang, T., and Ivančić, F. Counterexample-guided predicate abstraction of hybrid systems. *Theor. Comput. Sci. 354*, 2 (2006), 250–271.
2. Asarin, E., Dang, T., and Maler, O. The d/dt tool for verification of hybrid systems. In *CAV* (2002), vol. 2404 of *LNCS*, Springer, pp. 365–370.
3. Bournez, O., Maler, O., and Pnueli, A. Orthogonal polyhedra: Representation and computation. *Lecture Notes in Computer Science 1569* (1999), 46–60.
4. Boyd, S., and Vandenberghe, S. *Convex Optimization*. Cambridge University Press, 2004. Online `http://www.stanford.edu/~boyd/cvxbook.html`.
5. Chutinan, A., and Krogh, B. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of IEEE CDC* (1998), IEEE press.
6. Clarisó, R., and Cortadella, J. The octahedron abstract domain. In *Static Analysis Symposium* (2004), vol. 3148 of *LNCS*, Springer, pp. 312–327.
7. Cousot, P., and Cousot, R. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming* (1976), Dunod, Paris, France, pp. 106–130.
8. Dantzig, G. B. *Programming in Linear Structures*. USAF, 1948.
9. Fehnker, A., and Ivanĉić, F. Benchmarks for hybrid systems verification. In *HSCC* (2004), vol. 2993 of *LNCS*, Springer, pp. 326–341.
10. Frehse, G. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC* (2005), vol. 2289 of *LNCS*, Springer, pp. 258–273.
11. Girard, A. Reachability of uncertain linear systems using zonotopes. In *HSCC* (2005), vol. 3414 of *LNCS*, Springer, pp. 291–305.
12. Halbwachs, N., Proy, Y., and Roumanoff, P. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design 11*, 2 (1997), 157–185.
13. Hentenryck, P. V., Michel, L., and Benhamou, F. Newton: Constraint programming over nonlinear real constraints. *Science of Computer Programming 30*, 1–2 (1998), 83–118.
14. Henzinger, T. A. The theory of hybrid automata. In *Logic In Computer Science (LICS 1996)* (1996), IEEE Computer Society Press, pp. 278–292.

15. Henzinger, T. A., and Ho, P. HyTech: The Cornell hybrid technology tool. In *Hybrid Systems II* (1995), vol. 999 of *LNCS*, Springer, pp. 265–293.
16. Ivančić, F. *Modeling and Analysis of Hybrid Systems*. PhD thesis, University of Pennsylvania, December 2003.
17. Kurzhanski, A. B., and Varaiya, P. Ellipsoidal techniques for reachability analysis. In *HSCC* (2000), vol. 1790 of *LNCS*, Springer, pp. 202–214.
18. Lanotte, R., and Tini, S. Taylor approximation for hybrid systems. In *HSCC* (2005), vol. 3414 of *LNCS*, Springer, pp. 402–416.
19. Miné, A. A new numerical abstract domain based on difference-bound matrices. In *PADO II* (May 2001), vol. 2053 of *LNCS*, Springer, pp. 155–172.
20. Mitchell, I., Bayen, A., and Tomlin, C. Computing reachable sets for continuous dynamic games using level set methods. *IEEE Transactions on Automatic Control 50*, 7 (2005), 947–957.
21. Parillo, P. A. Semidefinite programming relaxation for semialgebraic problems. *Mathematical Programming Ser. B 96*, 2 (2003), 293–320.
22. Piazza, C., Antoniotti, M., Mysore, V., Policriti, A., Winkler, F., and Mishra, B. Algorithmic algebraic model checking I: Challenges from systems biology. In *CAV* (2005), vol. 3576 of *LNCS*, Springer, pp. 5–19.
23. Prajna, S., and Jadbabaie, A. Safety verification using barrier certificates. In *HSCC* (2004), vol. 2993 of *LNCS*, Springer, pp. 477–492.
24. Ratschan, S., and She, Z. Benchmarks for safety verification of hybrid systems. cf. `http://hsolver.sourceforge.net/benchmarks` (viewed Oct, 2007).
25. Ratschan, S., and She, Z. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *HSCC* (2005), vol. 3414 of *LNCS*, Springer, pp. 573–589.
26. Rodriguez-Carbonell, E., and Tiwari, A. Generating polynomial invariants for hybrid systems. In *HSCC* (2005), vol. 3414 of *LNCS*, Springer, pp. 590–605.
27. Sankaranarayanan, S., Dang, T., and Ivancic, F. A policy iteration technique for time elapse over template polyhedra (Extended Abstract). In *HSCC* (2008), LNCS, Springer. To Appear (2008).
28. Sankaranarayanan, S., Sipma, H. B., and Manna, Z. Constructing invariants for hybrid systems. In *HSCC* (2004), vol. 2993 of *LNCS*, Springer, pp. 539–555.
29. Sankaranarayanan, S., Sipma, H. B., and Manna, Z. Scalable analysis of linear systems using mathematical programming. In *Verification, Model-Checking and Abstract-Interpretation (VMCAI 2005)* (January 2005), vol. 3385 of *LNCS*.
30. Silva, B., Richeson, K., Krogh, B. H., and Chutinan, A. Modeling and verification of hybrid dynamical system using checkmate. In *ADPM 2000* (2000). available online from `http://www.ece.cmu.edu/~webk/checkmate`.
31. Tiwari, A. Approximate reachability for linear systems. In *HSCC* (2003), vol. 2623 of *LNCS*, Springer, pp. 514–525.
32. Tiwari, A., and Khanna, G. Non-linear systems: Approximating reach sets. In *HSCC* (2004), vol. 2993 of *LNCS*, Springer, pp. 477–492.