Constraint-based Approach for Analysis of Hybrid Systems*

Sumit Gulwani[1] and Ashish Tiwari[2]

[1] Microsoft Research, Redmond, WA 98052, sumitg@microsoft.com
[2] SRI International, Menlo Park, CA 94025, tiwari@csl.sri.com

Abstract. This paper presents a constraint-based technique for discovering a rich class of inductive invariants (disjunctions of polynomial inequalities of bounded degree) for verification of hybrid systems. The key idea is to introduce a template for the unknown invariants and then translate the verification condition of the hybrid system into an $\exists\forall$ constraint over the template unknowns (which are variables over reals) by making use of the fact that vector fields must point inwards at the boundary. These constraints are then solved using Farkas lemma. We also present preliminary experimental results that demonstrate the feasibility of our approach of solving the $\exists\forall$ constraints generated from models of real-world hybrid systems.

1 Introduction

The model checking problem seeks to determine if a given system satisfies a given property. For several interesting classes of systems (and properties), the model checking problem is theoretically intractable. As a result, techniques have been developed that are relatively complete for either verification or falsification. Predicate abstraction and abstract interpretation are examples of the former, while bounded model checking is an example of the latter. An attractive feature of bounded model checking (BMC) is that it reduces the search for (bounded) falsification to a single constraint that can be solved using powerful satisfiability modulo theories (SMT) solvers. One analog of BMC for verification is k-induction. The other analog, which we pursue in this paper in the context of hybrid systems, is an approach based on using templates to search for inductive invariants.

The general approach for verification of any kind of system is based on computing inductive invariants. In the case of hybrid systems, initial work on discovering inductive invariants was based on using iterative fixed-point computation based approaches like abstract interpretation or model checking [6, 10, 2]. Recently, constraint-based approaches have been proposed that search for invariants of some given form by reducing the problem to finding a satisfying solution to some constraints over the unknowns in the templates [21, 16]. Constraint-based

* Research supported in part by the National Science Foundation under grant CNS-0720721 and by NASA under Grant NNX08AB95A.

techniques offer two main advantages over fixed-point computation based techniques. First, they are goal-directed and hence have the potential to be more efficient. Second, they do not require the use of widening heuristics that lead to an uncontrollable loss of precision in fixed-point based techniques. Furthermore, constraint-based techniques can search for "deep" invariants of a known form, whereas the other techniques are more suited for "simple" invariants of an unknown form. Since hybrid systems typically have "deep" invariants of (a small number of) known simple forms, constraint-based technique are quite appealing. Even though they have demonstrated some success in the form of discovering *equational* invariants [21] and *conservatively* discovering conjunctions of polynomial inequalities [16], constraint-based techniques have not yet achieved their full potential in verification of hybrid systems.

In this paper, we develop the constraint-based approach further and show that it can be applied to discovering a rich class of inductive invariants for verification of hybrid systems. In particular, our constraint-based technique can be used for discovering invariants that involve disjunctions of polynomial inequalities. One part of the challenge here is in formulating the inductiveness requirement—if I holds in the current state $\boldsymbol{x}$ and there is a transition from $\boldsymbol{x}$ to $\boldsymbol{x}'$, then I holds in the state $\boldsymbol{x}'$—for the *continuous* dynamics. The key insight here is that this requirement can be captured precisely as a universally quantified formula, just as it can be done for *discrete* transitions. In the continuous case, inductiveness is equivalent to requiring that the vector field points "inwards" on the "boundary" of the invariant set I.

The key steps of our constraint-based approach for verification are
(1) introduce a template for the unknown inductive invariant and express the verification conditions as satisfiability of a $\exists\forall$ formula over the reals, where the existential quantification is over the template variables and the universal quantification is over the state variables (Section 3);
(2) use a generalization of Farkas' Lemma to eliminate the $\forall$ quantifiers and convert the $\exists\forall$ formula to an $\exists$ formula (over the reals) (Section 4.1); and
(3) use the bit-vector theory in SMT solvers to search for solutions of the $\exists$ formula in a bounded range (Section 4.2).

We start by defining continuous dynamical systems and hybrid systems in Section 2. We then show that the problem of discovering invariants and verifying safety can be reduced to solving $\exists\forall$ constraints over the reals (Section 3). We present our approach for solving these constraints in Section 4. We present several nontrivial examples of continuous dynamical systems and hybrid systems that were successfully analyzed using our approach (Section 5). We compare with related work in Section 6 before concluding.

2 Continuous Dynamical and Hybrid Systems

A *continuous dynamical system* is a tuple $\langle X, \mathtt{Init}, \mathtt{Inv}, f \rangle$ where X is a finite set of variables interpreted over the reals $\mathbb{R}$, $\mathbf{X} = \mathbb{R}^X$ is the set of all valuations of the variables X, $\mathtt{Init} \subseteq \mathbf{X}$ is the set of initial states, $\mathtt{Inv} \subseteq \mathbf{X}$ is the state invariant, and $f : \mathbf{X} \mapsto \mathbf{X}$ is a vector field that specifies the continuous dynamics (as $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$). We assume that f satisfies the standard assumptions for existence and uniqueness of solutions to ordinary differential equations. The set $\mathtt{Inv}$ specifies the domain where the system is defined. The semantics of a continuous dynamical system are standard.

Example 1. Consider the following adaptive cruise controller where a car is following a leading car maintaining a safe distance [9, 18]. Let gap, v_f, v, and a respectively represent the gap between the two cars, the velocity of the leading car, and the velocity and acceleration of the rear car. The system dynamics are given by the following differential equations [18]:

$$\dot{v} = a, \;\; \dot{a} = -3a - 3(v - v_f) + (gap - (v + 10)), \;\; g\dot{a}p = v_f - v, \;\; \dot{v_f} = a_f$$

where a_f is the acceleration of the leading car and an input in this model. Formally, we have a linear continuous dynamical system $\langle X, \mathtt{Init}, \mathtt{Inv}, f\rangle$ where $X = \{v, v_f, a, gap, a_f\}$, f is defined by the right-hand sides of the above differential equations, $\mathtt{Inv} = \{v \geq 0, \;\; v_f \geq 0, \;\; -2 \leq a \leq 5, \;\; -2 \leq a_f \leq 5\}$, $\mathtt{Init} = \{gap = 5, v = v_f, a = 0\}$. The invariant $\mathtt{Inv}$ captures the physical constraints that the cars do not move backwards and that the acceleration of the two cars is bounded from above and below. The initial states indicate when the above control law may be invoked. The problem is to verify that the rear car would never collide with the car in front, i.e., always $gap > 0$. We note that reachability is decidable for certain classes of linear dynamical systems [12], but this example does not fall in these decidable classes. ■

A *hybrid system* $\mathtt{HS} = (\mathbf{Q}, X, \mathtt{Init}, \mathtt{Inv}, t, f)$ consists of a finite set of modes $\mathbf{Q}$, a finite set X of variables — that together define the state space $\mathbf{Q} \times \mathbf{X} := \mathbf{Q} \times \mathbb{R}^X$ of the system — a mapping $\mathtt{Init} : \mathbf{Q} \mapsto 2^{\mathbf{X}}$ that defines the initial states (in each mode), a mapping $\mathtt{Inv} : \mathbf{Q} \mapsto 2^{\mathbf{X}}$ that defines the state invariant of each mode, a mapping $f : \mathbf{Q} \mapsto (\mathbf{X} \mapsto \mathbf{X})$ that specifies the continuous dynamics in each mode, and a mapping $t : \mathbf{Q} \times \mathbf{Q} \mapsto 2^{\mathbf{X}}$ that specifies the discrete transitions. Specifically, for any two modes $q, q' \in \mathbf{Q}$, the system can jump from a state $(\mathbf{q}, \mathbf{x})$ to any state $(\mathbf{q}', \mathbf{x})$ if $\mathbf{x} \in t(\mathbf{q}, \mathbf{q}')$. Note that, for simplicity of presentation, we are forcing the discrete transitions to have identity reset maps (that is, $\boldsymbol{x}$ is not updated), but our method works in the other case as well. Hence, $t(\mathbf{q}, \mathbf{q}')$ is just the *guard*, or *switching condition*, for going from mode $\mathbf{q}$ to mode $\mathbf{q}'$. We assume that the semantics of hybrid systems and the set of *reachable states* are defined in the standard way, see [1].

Example 2. We consider a model of adaptive cruise control coupled with transmission from [24]. The hybrid system here is described by $\langle \mathbf{Q}, X, \mathtt{Init}, \mathtt{Inv}, t, f\rangle$ where $\mathbf{Q} := \{normal, max brake, max acc\} \times \{1st, 2nd, 3rd, 4th\} \times \{acc, cc\}$ and $X := \{gap, v, v_f, a_f\}$. Thus, the hybrid system has 24 modes depending on the gear of the rear car (1st, 2nd, 3rd, 4th), its cruise control mode (regular cruise control, cc, or adaptive cruise control, acc), and its mode of operation (normal, max-braking, or max-acceleration). The dynamics in the 24 modes of the adaptive cruise control model is defined as follows:

$\dot{gap} = v_f - v$, in all modes
$\dot{v} = -3.5$, in all maxbraking modes
$\dot{v} = 6 - i$, in all max-acceleration and i-th gear modes
$\dot{v} = 0.9(v_{des} - v)$, in all normal, regular cruise control (cc) modes, and
$\dot{v} = -0.66v + 0.08gap - 0.4 + 0.26v_f$, in all normal, adaptive (acc) modes
where v_{des} is a parameter set to the desired velocity in the cruise control mode. The set $\mathtt{Inv}(q)$ is the conjunction of all the following applicable facts:

$-3.5 > -0.66v + 0.08gap - 0.4 + 0.26v_f$	maxbrake, acc, all gears
$-3.5 \leq -0.66v + 0.08gap - 0.4 + 0.26v_f \leq 6 - i$	normal, acc, i-th gear
$-0.66v + 0.08gap - 0.4 + 0.26v_f > 6 - i$	maxacc, acc, i-th gear
$-3.5 > 0.9(v_{des} - v)$	maxbrake, cc, all gears
$-3.5 \leq 0.9(v_{des} - v) \leq 6 - i$	normal, cc, i-th gear
$0.9(v_{des} - v) > 6 - i$	maxacc, cc, i-th gear
$0 \leq v_f \leq 60, -3.5 \leq a_f \leq 5, gap \leq 40$	acc
$gap \geq 38$	cc
$0 \leq v \leq 6.7$	1st gear
$6.7 \leq v \leq 14.2$	2nd gear
$14.2 \leq v \leq 29.8$	3rd gear
$29.8 \leq v \leq 60$	4th gear

All discrete transitions have identity reset maps (that is, the continuous variables do not change values on discrete transitions), and the guards of discrete transitions can be obtained as negations of the state invariants. For example, there is a transition from *normal, acc, 1st-gear* to *normal, acc, 2nd-gear* if $v > 6.7$. For more details on the complete model, see [24]. ■

We use the notation $K[X]$ to denote the set of polynomials with coefficients in K and variables in X. We use $\mathbb{Q}$ to denote the set of rationals and $\mathbb{Z}$ ($\mathbb{Z}^+$) to denote the set of (positive) integers.

3 Verification of Hybrid Systems

Given a *hybrid system* $\mathtt{HS} = (\mathbf{Q}, X, \mathtt{Init}, \mathtt{Inv}, t, f)$, and a safety property $S : \mathbf{Q} \rightarrow 2^{\mathbf{X}}$, the problem of hybrid system verification is to determine if the set of reachable states of the hybrid system in each mode $q \in \mathbf{Q}$ is a subset of $S(q)$.

The classical approach for solving the verification problem involves finding an inductive invariant map $I : \mathbf{Q} \rightarrow 2^{\mathbf{X}}$ such that the following constraints, referred to as the *verification condition*, hold for each mode $q \in \mathbf{Q}$.

A1. (Initial Constraint) $\mathtt{Init}(q) \subseteq I(q)$.
A2. (Transition Constraint) For all modes $q' \in \mathbf{Q}$, $I(q) \cap t(q, q') \subseteq I(q')$.
A3. (Safety Constraint) $I(q) \subseteq S(q)$.
A4. (Inductive Constraint) If the system is in a state from the set $I(q) \cap \mathtt{Inv}(q)$, then it stays in the set $I(q)$ at any time in the future as per the dynamics $f(q)$ (such a set $I(q)$ is called positively invariant in control systems [5]).

In this section, we present a constraint based technique for discovering an inductive invariant map that maps different modes to *closed semi-algebraic* invariants of the form $\bigwedge_i \bigvee_j p_{ij} \geq 0$, where $p_{ij} \in \mathbb{Q}[X]$ are polynomials of bounded degrees over X. We further assume that the initial conditions $\mathtt{Init}(q)$, the safety conditions $S(q)$, and the transition conditions $t(q, q')$ are semi-algebraic, and that the flow f is given by polynomials. This class of *polynomial hybrid systems* is very general and covers a wide variety of examples.

The key idea of our technique is to translate the verification condition into a $\exists\forall$ constraint over real variables. (Section 4 then describes how to solve such formulas using Farkas lemma.) This is achieved by choosing a template, $\mathcal{I} : \mathbf{Q} \mapsto 2^{\mathbf{U},\mathbf{X}}$, for the inductive invariant I, where U is a finite set of new template parameters and $\mathcal{I}(q) := \bigwedge_i \bigvee_j p'_{ij} \geq 0$ with $p'_{ij} \in \mathbb{Q}[U, X]$. The first three constraints in the verification condition can be easily translated into a $\exists\forall$ constraint over real variables by simply substituting the invariant template $\mathcal{I}(q)$ in place of $I(q)$ and replacing $\subseteq$ relation by $\Rightarrow$ relation. (This is because the existence of I gets translated to existence of the unknown parameters U.) The challenge is to do this for the *invariant constraint* (A4). For that, we make use of continuity and obtain the following critical (necessary and sufficient) *verification condition for continuous dynamical systems.* We simplify presentation by just considering that each conjunct in the invariant contains two disjuncts, $p_1 \geq 0 \vee p_2 \geq 0$.

Proposition 1. *Let $\langle X, \mathit{Init}, \mathit{Inv}, f\rangle$ be a continuous dynamical system and p_1, p_2 be two polynomials in $\mathbb{Q}[X]$. The set $I := \bigwedge_i p_1^i \geq 0 \vee p_2^i \geq 0$ is positively invariant for the continuous dynamical system iff for all i:*

$$\begin{array}{ll}
I(\boldsymbol{x}) \wedge p_1^i(\boldsymbol{x}) = 0 \wedge p_2^i(\boldsymbol{x}) < 0 \wedge \mathit{Inv}(\boldsymbol{x}) \Rightarrow dp_1^i(\boldsymbol{x})/dt \geq 0, \quad \textit{and} & \\
I(\boldsymbol{x}) \wedge p_1^i(\boldsymbol{x}) < 0 \wedge p_2^i(\boldsymbol{x}) = 0 \wedge \mathit{Inv}(\boldsymbol{x}) \Rightarrow dp_2^i(\boldsymbol{x})/dt \geq 0, \quad \textit{and} & \text{(A4')} \\
I(\boldsymbol{x}) \wedge p_1^i(\boldsymbol{x}) = 0 \wedge p_2^i(\boldsymbol{x}) = 0 \wedge \mathit{Inv}(\boldsymbol{x}) \Rightarrow dp_1^i(\boldsymbol{x})/dt \geq 0 \vee dp_2^i(\boldsymbol{x})/dt \geq 0 &
\end{array}$$

Here $\frac{dp}{dt}$ denotes the time derivative of p, also called the Lie derivative *of p, in the vector field defined by f; that is, $\frac{dp}{dt} := \sum_{x \in X} \frac{\partial p}{\partial x} \frac{dx}{dt} := \sum_{x \in X} \frac{\partial p}{\partial x} f_x$.*

Proposition 1 essentially says that the vector field should point "inwards" on the boundary of the invariant set $I(q)$. The boundary of $p_1 \geq 0 \vee p_2 \geq 0$ is contained in the union of the three sets defined by $p_1 = 0 \wedge p_2 < 0$, $p_1 < 0 \wedge p_2 = 0$, and $p_1 = 0 \wedge p_2 = 0$. For each set, we have a formula in Constraint A4' stating that the vector field is pointing inwards. For instance, when $p_1 = 0 \wedge p_2 < 0$, then the vector field points inwards iff the Lie derivative, $\frac{dp_1}{dt}$, of p_1 is non-negative.

The Lie derivatives, $\frac{dp}{dt}$, in Equation A4' get reduced to polynomials as $\frac{dp}{dt} := \sum_{x \in X} \frac{\partial p}{\partial x} \frac{dx}{dt}$, and this summation simplifies into a polynomial if p is a polynomial and $\frac{d\boldsymbol{x}}{dt} := f(\boldsymbol{x})$ contains only polynomials. Figure 1 shows the construction of the $\exists\forall$ formula over real variables using the Constraints A1,A2, and A3, and (a stronger variant of) Constraint A4'. Since the full first-order theory of reals is decidable [25], it follows that the problem of discovering inductive invariants for hybrid systems over the class of positive boolean combination of (non-strict) polynomial inequalities of bounded degree is decidable. However, in the next

```
HS2ExistsForall(HS,S,I) =
// Inputs: HS := (Q, X, Init, Inv, t, f), Safety property S : Q → 2^X,
      Template I : Q → 2^{U,X}, where I(q) := ⋀_{i=1}^{n} ⋁_{j=1}^{m} p_ij ≥ 0, p_ij ∈ Q[U, X]
ans := true
for all q ∈ Q do
  l1 : ans := ans ∧ (Init(q) ⇒ I(q)) ∧ (I(q) ∧ Inv(q) ⇒ S(q))
  l2 : for all q' ∈ Q do ans := ans ∧ (I(q) ∧ t(q, q') ⇒ I(q'))
  l3: Lp := Σ_{x∈X} (∂p_ij/∂x) f_x(q)
  l4 : ans := ans ∧ ⋀_{i=1,j=1}^{i=n,j=m} (I(q) ∧ Inv(q) ∧ p_ij = 0 ∧ ⋀_{k=1,k≠j}^{k=m} p_ik ≤ 0 ⇒ Lp ≥ 0)
return(∃U∀X ans)
```

Fig. 1. Translating safety verification to satisfiability of an $\exists\forall$ formula.

section, we present a more *practical* technique for solving the $\exists\forall$ constraints generated above.

Example 3 (Verification to $\exists\forall$ Constraint). Consider the dynamical system from Example 1 and the verification problem stated therein. Let us assume a template that searches for linear invariants:

$$\mathcal{I} := \alpha v_f + \beta v + \gamma a + \delta gap >= \epsilon$$

where $U := \{\alpha, \beta, \gamma, \delta, \epsilon\}$. The safety of the adaptive cruise control law reduces to the satisfiability of the following $\exists\forall$ constraint which essentially says that $\mathcal{I} \wedge gap \geq 0$ is an inductive invariant.

$$\begin{array}{ll}\exists U \forall X : ((gap = 5 \wedge v = v_f \wedge a = 0 \Rightarrow \mathcal{I} \wedge gap \geq 0) & (A1)\\ \quad \wedge(\alpha v_f + \beta v + \gamma a + \delta gap = \epsilon \ \wedge \ gap \geq 0 \ \wedge \ \texttt{Inv} \Rightarrow p \geq 0) & (A4')\\ \quad \wedge(\alpha v_f + \beta v + \gamma a + \delta gap \geq \epsilon \ \wedge \ gap = 0 \ \wedge \ \texttt{Inv} \Rightarrow v_f - v \geq 0)) & (A4')\end{array}$$

where p is the Lie derivative of the template and is equal to $\alpha \dot{v_f} + \beta \dot{v} + \gamma \dot{a} + \delta \dot{gap} = \alpha a_f + \beta a - 3\gamma a - 3\gamma v + 3\gamma v_f + \gamma gap - \gamma v - 10\gamma + \delta v_f - \delta v$. Similarly, $v_f - v$ is the Lie derivative of gap. Note that X, `Inv` are defined in Example 1. ■

4 Solving $\exists\forall$ formulas

We check for satisfiability of the $\exists\forall$ formula in two steps. First we eliminate the inner universal quantifier and next we check for satisfiability of the resulting existential formula over a finite domain using a satisfiability modulo theories (SMT) solver.

4.1 Step 1: Eliminating Universal Quantification

The inner universal quantifier from the $\exists\forall$ formula is eliminated using the following variant of Farkas Lemma.

```
ExistsForall2Exists(ϕ) =
// Input: ϕ := ∃U∀X ⋀_{i=1}^n (⋁_{j=1}^m p_ij ≤ 0 ∨ ⋁_{k=1}^l p'_ik < 0), where p_ij, p'_ik ∈ ℚ[U, X]
V := ∅, ans := true
for i = 1 to n do
    V := V ∪ {μ_i} ∪ {μ_ij : j = 1, ..., m} ∪ {ν_ik : k = 1, ..., l}
    ans := ans∧ ElimX(μ_i + Σ_{j=1}^m μ_ij p_ij + Σ_{k=1}^l μ_ik p'_ik = 0) ∧ (μ_i > 0 ∨ ⋁_{j=1}^m μ_ij > 0)
return(∃U∃V ans)

ElimX(p = 0) = // Input p ∈ ℚ[U, X]
Let p := Σ_α p_α X^α, where p_α ∈ ℚ[U] are the coefficients of p in (ℚ[U])[X]
return(⋀_α p_α = 0)
```

Fig. 2. Translating ∃∀ formula to an ∃ formula.

Lemma 1. *For polynomials $p_j, r_k \in \mathbb{Q}[X]$, the formula $\bigwedge_{j\in J} p_j > 0 \wedge \bigwedge_{k\in K} r_k \geq 0$ is unsatisfiable (over the reals) if there exist non-negative constants μ, μ_j $(j \in J)$, and ν_k, $(k \in K)$ such that $\mu + \sum_{j\in J} \mu_j p_j + \sum_{k\in K} \nu_k r_k = 0$ and at least one of μ_j, μ is strictly positive.*

If polynomials p_j, r_k are linear (more generally, linear only in the universal variables; for example, see the constraint in Example 3), then the condition above is both necessary and sufficient. However, the condition is not necessary for unsatisfiability when p_j, r_k are arbitrary nonlinear polynomials.[3] After applying Lemma 1, the universal variables can be eliminated by just equating the coefficients of each of the power products in the following expression to zero.

$$\mu + \sum_{j\in J} \mu_j p_j + \sum_{k\in K} \nu_k r_k$$

Note that we can convert *any* universally quantified arithmetic formula $\forall X : \phi$ into an existentially quantified formula using the above lemma.[4]

Figure 2 shows the pseudo-code for transforming an ∃∀ constraint to a ∃ formula and Example 4 provides an illustration. Note that the template variables U and the multipliers μ and ν introduced by Farkas Lemma are existentially quantified.

Theorem 1. *Let HS be a hybrid system and S a safety property. For any template $\mathcal{I}$, if the constraint ExistsForall2Exists(HS2ExistsForall(HS, $S, \mathcal{I}$)) is satisfiable, then for every reachable state $(q, \boldsymbol{x})$ of HS, it is the case that $\boldsymbol{x} \in S(q)$.*

[3] There is a generalization of Farkas Lemma for arbitrary polynomials, called Positivstellensatz [14], obtained by replacing the multipliers μ_j, ν_k by sum of squares of polynomials, but we did not use it in our experiments.

[4] This is achieved by converting ϕ into conjunctive normal form $\bigwedge_i(\bigvee_j p_{ij} \geq 0 \vee \bigvee_k r_{ik} > 0)$ and noting that $\forall X : \phi \equiv \bigwedge_i \forall X(\bigvee_j p_{ij} \geq 0 \vee \bigvee_k r_{ik} > 0) \equiv \bigwedge_i(\neg(\bigvee_j p_{ij} \geq 0 \vee \bigvee_k r_{ik} > 0)$ is unsatisfiable$) \equiv \bigwedge_i((\bigwedge_j -p_{ij} > 0 \wedge \bigwedge_k -r_{ik} \geq 0)$ is unsatisfiable). We can now use Lemma 1 on each conjunct.

Example 4. Consider the $\exists\forall$ formula in Example 3. To avoid clutter, we illustrate the $\forall$ elimination on a simpler subformula from that formula:

$$\exists U \forall X : (\alpha v_f + \beta v + \gamma a + \delta gap \geq \epsilon \ \wedge \ gap = 0 \ \wedge \ 2a \geq -7 \Rightarrow v_f - v \geq 0)$$

Using Lemma 1, the validity of the above formula is equivalent to the existence of constants $V := \{\nu_1, \lambda, \nu_2, \mu_1, \mu_2\}$ such that

$$\nu_1(\alpha v_f + \beta v + \gamma a + \delta gap - \epsilon) + \lambda gap + \nu_2(2a + 7) + \mu_1(v - v_f) + \mu_2 = 0,$$

and $\nu_1, \mu_1, \mu_2 \geq 0$ and at least one of the μ's is strictly positive. By equating the coefficients to 0, we get the following existentially quantified formula,

$$\begin{aligned} \exists U \exists V : \nu_1\alpha - \mu_1 = 0 \ \wedge \ \nu_1\beta + \mu_1 = 0 \ \wedge \ \nu_1\gamma + 2\nu_2 = 0 \ \wedge \ \nu_1\delta + \lambda = 0 \wedge \\ \mu_2 - \nu_1\epsilon + 7\nu_2 = 0 \ \wedge \ \textstyle\bigwedge_i \mu_i \geq 0 \ \wedge \ \bigwedge_i \nu_i \geq 0 \ \wedge \ (\mu_1 > 0 \vee \mu_2 > 0) \end{aligned}$$

A possible solution is

$$\nu_1 = 2, \lambda = -2, \nu_2 = 1, \mu_1 = 2, \mu_2 = 1, \alpha = 1, \beta = -1, \gamma = -1, \delta = 1, \epsilon = 4.$$

Note that μ_1 is strictly positive. This corresponds to the inductive invariant $v_f - v - a + gap \geq 4$. We remark here that the full example contains additional constraints, but the above solution for U continues to be a solution and it is the solution computed by our tool. ■

4.2 Step 2: Solving the $\exists$ Constraint using an SMT Solver

We have reduced the verification problem to the satisfiability of some (existentially quantified) nonlinear constraints. The important point to note here is that we are interested in finding solutions, rather than showing unsatisfiability, of the generated existential formula.

We search for solutions of the nonlinear constraints using the bit-vector decision procedure of an SMT solver. The translation of $\exists Y : \phi$ to bit-vectors is obtained in several steps. First polynomials in $\mathbb{Q}[Y]$ that occur in ϕ are converted to polynomials in $\mathbb{Z}[Y]$ by multiplying suitably by positive integer constants. Next we pick an integer lower bound $\boldsymbol{l}$ and an integer upper-bound $\boldsymbol{u}$ for the variables Y. Finally, we search for integer solutions for Y in the chosen finite range by searching for the bit-level representation. We choose a size for the bit-vectors by conservatively estimating the number of bits that would be required to evaluate the polynomials in ϕ over the range $\boldsymbol{l} \leq Y \leq \boldsymbol{u}$. The pseudo-code for the translator is given in Figure 3. The bounds l_i, u_i for the variable $y_i \in Y$ are picked based on whether y_i is a template variable or a variable introduced by Farkas Lemma.

4.3 Discussion

Comparing the overall approach to bounded model checking, we note that both approaches translate the analysis problem into a constraint satisfiability problem. In the case of BMC, the generated constraint encodes existence of a counterexample, whereas here the generated constraint encodes existence of a proof.

```
Exists2BitVector(∃Y : φ, l, u) =
// Inputs:Y := {y_1,...,y_n}, φ := ⋀_i ⋁_j p_ij ~_ij 0, where p_ij ∈ ℤ[Y], ~_ij ∈ {≥,=,>}
          l, u ∈ ℤ^n given lower- and upper-bounds for Y
forall i,j: m_ij := estimate max #bits reqd to evaluate p_ij when l ≤ Y ≤ u
Let m be the maximum of m_ij's
ans := declare each y_i to be a bit-vector of size m
return(ans, ⋀_i ⋁_j E2BVA(p_ij, ~_ij) ∧ E2BVA(Y − l, ≥) ∧ E2BVA(u − Y, ≥))

E2BVA(p, ~) = // p := p_1 − p_2, where p_1, p_2 ∈ ℤ^+[Y]
let E2BVB(p) = p' where p' is obtained by replacing *, +, ≥, >, =
                         by corresponding bit-vector operations in p
return(E2BVB(p_1) E2BVB(~) E2BVB(p_2))
```

Fig. 3. Converting satisfiability checking to bit-vector satisfiability problem. The bit-vector instance searches for all bounded integer solutions for Y in the range $\boldsymbol{l} \leq Y \leq \boldsymbol{u}$ that satisfy ϕ.

When verifying a hybrid system with a large number of discrete modes, we start by applying our technique to *one mode* using linear and quadratic templates. If we find an invariant for a particular mode, we use it as a starting point to construct refined templates for the full system (Example 6).

Our constraint-based technique for verification can be used for solving instances of the *synthesis* problem as well. The technique uniformly treats the entities of the verification condition, which includes both the inductive invariants and the description of the system. It does not matter whether the invariants are unknown or parts of the system are unknown or both of them are unknown. As long as there is sufficient information in the system description, the constraint-based approach can potentially find a solution for the unknown quantities.

Example 5. Consider the classical thermostat example, which is a hybrid system with two modes: in the "on" mode, temperature x increases as $dx/dt = 100 - x$, and in the "off" mode, it decreases as $dx/dt = -x$. We wish to synthesize the control logic that determines when to switch modes. Assume initially mode is "off" and $x = 78$. The goal is to ensure $75 \leq x \leq 80$ always. For simplicity, assume that the specified safety property, $75 \leq x \leq 80$, is also an inductive invariant (and we do not guess a template for the invariant). Assume that we guess that the guard for the transition from heater-on to heater-off mode is of the form $x \geq \alpha$ and that the guard for the reverse transition is $x \leq \beta$. We can now write the verification conditions (A1,A4') as follows:

$$\begin{aligned}\exists \alpha, \beta : \forall x : & (x = 75 \wedge x > \beta \Rightarrow -x \geq 0) \wedge (x = 80 \wedge x > \beta \Rightarrow -x \leq 0) \wedge \\ & (x = 75 \wedge x < \alpha \Rightarrow 100 - x \geq 0) \wedge (x = 80 \wedge x < \alpha \Rightarrow 100 - x \leq 0) \wedge \\ & (x = 78 \Rightarrow x > \beta)\end{aligned}$$

One solution returned by our constraint solver was $\alpha = \beta = 76$. However, this solution leads to zeno behavior. We can add additional constraints (not described in this paper) that capture the requirement that the switching logic be most liberal, in which case we get $\beta = 75$ and $\alpha = 79$. ■

5 Experimental Results

The approach described in this paper has been partially implemented in the form of two separate components. The first component takes an $\exists\forall$ formula (over arbitrary nonlinear polynomials) and returns an $\exists$ formula. The second component takes the $\exists$ formula and creates a Yices [8] formula over bit-vectors. The implementation is in Lisp. The bit-vector decision procedure of Yices is used to finally search for solutions. The front-end step of generating the $\exists\forall$ formula from a hybrid system description has not been automated yet.

All examples presented in this paper were analyzed automatically using the above tools. Some results are reported in Table 1.

Example 6 (Adaptive Cruise Control with Transmission). Consider the cruise control model from Example 2. The safety property to establish is that inter-vehicle separation remains positive; specifically, $gap \geq 5$. We assume that initially the rear car enters the mode *normal, acc, 4-th gear* from the mode *normal, cc, 4-th gear* and $v = v_f$. We wish to prove the safety property assuming that the velocity v_f of the leading car remains bounded between $30 \leq v_f \leq 60$.

Our tools prove the safety by generating the following invariant for each of the acc modes:

Invariant	Modes
$2gap - 2v + v_f - 2 \geq 0,\ gap \geq 5$	normal, acc, all gears
$-350 \leq -66v + 8gap - 40 + 26v_f$	normal, acc, all gears
false	max-braking, acc, all gears
$2gap - 2v + v_f - 2 \geq 0,\ gap \geq 5$	max-acceleration, acc, all gears

Note that the max-braking mode is not reachable from the chosen initial states. We did not generate the invariants for all modes in one step. We first generated invariants for single modes and that gave us an idea of the form of invariants and helped refine our template. Using a refined template, we generated invariants for all the acc-modes simultaneously. ■

Example 7 (Human Blood Glucose Metabolism). We consider the model of insulin metabolism in the body of a Type-I diabetic patient [23, 13]. For purposes of modeling insulin concentration in the human body, the body is divided into six compartments – brain (B), heart (H), gut (G), lungs (L), kidney (K), and periphery (P) – and each state variable represents the insulin concentration in one such compartment (there are two variables for the "periphery" compartment). The dynamics of the system are given as follows, see also [23]:

$$dI_B/dt = -45/26 I_B + 45/26 I_H$$
$$dI_H/dt = 45/99 I_B - 312/99 I_H + 90/99 I_L + 72/99 I_K + 105/99 I_{PV} + u$$
$$dI_G/dt = 72/94 I_H - 72/94 I_G$$

Example	Dim	Vars	Bits	Assertions	Time
disjunction Ex. 9	2	14	6	50	7ms
delta-notch	4	34	8	120	30ms
plankton Ex. 8	3	31	8	110	56ms
thermostat	1	29	20	126	.45s
thermostat synthesis Ex. 5	1	21	20	75	1.2s
ACC Ex. 1	5	28	12	95	1.3s
acc-transmission Ex. 2	4	35	24	122	4.7s
insulin Ex. 7	7	66	18	180	18s

Table 1. Experimental Results. We report the size of the Yices formulas generated in the various examples (number of variables, Vars; size of bit-vectors, Bits; and number of assertions, Assertions) and the time (Time) taken by Yices to find a model on a 64-bit Pentium 3.4GHz cpu with 2MB cache. Dim is the number of continuous variables in the example.

$$\begin{aligned}
dI_L/dt &= 18/114 I_H - 720/10000 I_H + 72/118 I_G - 2880/10000 I_G - 90/118 I_L \\
dI_K/dt &= 72/51 I_H - 72/51 I_K - 2160/10000 I_K \\
dI_{PV}/dt &= 105/74 I_H - 105/74 I_{PV} - 674/1480 I_{PV} + 674/1480 I_{PI} \\
dI_{PI}/dt &= 1/20 I_{PV} - 1/20 I_{PI} - 21231/51580 I_{PI}
\end{aligned}$$

The control input u in this case is the insulin injected into the body by an external insulin pump. Since we assume a Type-I diabetic, there is no pancreatic insulin release and hence no feedback from the glucose metabolism model. Assuming that the input u is bounded between 20 and 25, we can compute bounds or ranges for insulin concentrations in different body compartments. As remarked earlier, we can easily invert the analysis and ask for acceptable bounds on insulin injection rate that will ensure bounded insulin concentration levels in the body. ■

Example 8. Consider the following Phytoplankton Growth Model (see [3] and references therein): $\dot{x}_1 = 1 - x_1 - \frac{x_1 x_2}{4}$, $\dot{x}_2 = (2x_3 - 1)x_2$, $\dot{x}_3 = \frac{x_1}{4} - 2x_3^2$, where x_1 denotes the substrate, x_2 the phytoplankton biomass, and x_3 the intracellular nutrient per biomass. For this nonlinear dynamical system, we can immediately generate the following invariant: $0 \leq x_1 \leq 2,\ 0 \leq x_2 \leq 1,\ 0 \leq x_3 \leq 1/2$. ■

Example 9 (Disjunctive Invariants). Our technique can be used to generate disjunctive invariants. Consider the system $dx/dt = -y,\ dy/dt = -x$ with initial states given by $x \geq 3$. Using the template $x \geq \alpha \ \vee \ y \geq \beta$, we can generate the invariant $x \geq 0 \vee y \geq 0$. ■

6 Related Work

The approach of using templates and generating invariants of a specific form for hybrid systems was introduced simultaneously by Sankaranarayanan et. al. [21]

and Prajna et. al. [15, 16, 17]. In all such approaches, an $\exists\forall$ formula is generated, although this may not be explicitly stated. The various approaches differ in the form of the invariants considered, the technique used to generate the $\exists\forall$ formula, and the approach for solving it. Templates are restricted to polynomial equations in [21] and Proposition 1 is not required there. The approach for solving the $\exists\forall$ constraints is based on Gröbner basis computation. Polynomial inequality templates are used in [16] and a variant of Proposition 1 is used there. However, to solve the generated constraints, the authors replaced the constraints by something stronger (essentially voiding the benefit of Proposition 1) and attempted to recover the lost generality using iterative methods. The constraint solving method is based on convex optimization and sum-of-squares computation. In essence, a slightly more general form of Lemma 1 inspired by Positivstellensatz is used in [16]. We build upon these works and explore a new translation into $\exists\forall$ constraints and the use of SMT solvers as the backend engines.

Tiwari [26] generated linear inductive invariants for linear systems. Rodriguez-Carbonell and Tiwari [19] showed that the best (strongest) possible polynomial equational invariant was computable for hybrid systems with linear dynamics in each mode. Pappas et al. have also considered the problem of computing invariants, but only for linear systems, using interesting techniques [27, 28].

In software program analysis, constraint based techniques have been successfully applied for discovering conjunctive linear arithmetic invariants [7, 20, 22], non-linear polynomial invariants [11] and invariants in the combined theory of linear arithmetic and uninterpreted functions [4].

7 Conclusion

We show that the verification technique based on guessing the form of inductive invariant and searching for invariants of that form using SMT solvers is a potent approach for verifying hybrid systems. Its extension to solving the synthesis problem is left for future work. Using efficient nonlinear constraint solvers directly could also significantly improve the efficiency of our approach and remains to be explored.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(3):3–34, 1995.

[2] R. Alur, T. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(2):971–984, July 2000.

[3] O. Bernard and J.-L. Gouze. Global qualitative description of a class of nonlinear dynamical systems. *Artificial Intelligence*, 136:29–59, 2002.

[4] D. Beyer, T. Henzinger, R. Majumdar, and A. Rybalchenko. Invariant synthesis for combined theories. In *VMCAI*, volume 4349 of *LNCS*, pages 378–394, 2007.

[5] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.

[6] A. Chutinan and B. H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *HSCC'99*, LNCS 1569.

[7] M. Colón, S. Sankaranarayanan, and H. Sipma. Linear invariant generation using non-linear constraint solving. In *CAV'03*.

[8] B. Dutertre and L. de Moura. A fast linear-arithmetic solver for dpll(t). In *CAV 2006*, volume 4144 of *LNCS*, pages 81–94, 2006. `http://yices.csl.sri.com/`.

[9] D. Godbole and J. Lygeros. Longitudinal control of the lead car of a platoon. *IEEE Transactions on Vehicular Technology*, 43(4):1125–35, 1994.

[10] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.

[11] Deepak Kapur. Automatically generating loop invariants using quantifier elimination. In *Deduction and Applications*, 2005.

[12] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computations for families of linear vector fields. *J. Symbolic Computation*, 32(3):231–253, 2001.

[13] R. S. Parker, F. J. Doyle, and N. A. Peppas. A model-based algorithm for blood glucose control in type I diabetes patients. *IEEE Trans BioMed Eng.*, 46(2), 1999.

[14] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometric methods in robustness and optimization*. PhD thesis, California Inst. of Tech., 2000.

[15] S. Prajna. Barrier certificates for nonlinear model validation. In *Proc. IEEE Conference on Decision and Control*, 2003.

[16] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492, 2004.

[17] S. Prajna, A. Jadbabaie, and G. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans Aut Control*, 2005.

[18] A. Puri and P. Varaiya. Driving safely in smart cars. In *Proceedings of the 1995 American Control Conference*, 1995.

[19] E. Rodriguez-Carbonell and A. Tiwari. Generating polynomial invariants for hybrid systems. In *HSCC*, volume 3414 of *LNCS*, pages 590–605. Springer, 2005.

[20] S. Sankaranarayanan, H. Sipma, and Z. Manna. Non-linear loop invariant generation using gröbner bases. In *POPL'04*.

[21] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In *HSCC*, volume 2993 of *LNCS*, pages 539–554, 2004.

[22] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *VMCAI'05*.

[23] J. T. Sorensen. *A physiologic model of glucose metabolism in man and its use to design and assess improved insulin therapies for diabetes*. PhD thesis, Dept. Chem. Eng., Massachusetts Inst. Technology (MIT), Cambridge, 1985.

[24] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh. Verification of a cruise control system using counterexample-guided search. *Control Engineering Practice*, 12(10):1269–78, 2004.

[25] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1948. Second edition.

[26] A. Tiwari. Approximate reachability for linear systems. In O. Maler and A. Pnueli, editors, *HSCC*, volume 2623 of *LNCS*, pages 514–525. Springer, April 2003.

[27] H. Yazarel and G. J. Pappas. Geometric programming relaxations for linear system reachability. In *Proc. 2004 American Control Conference*, 2004.

[28] H. Yazarel, S. Prajna, and G. J. Pappas. S.O.S. for safety. In *Proc. 43rd IEEE Conference on Decision and Control*, 2004.