Constructing Invariants for Hybrid Systems

Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna *

Computer Science Department
Stanford University
Stanford, CA 94305-9045
{srirams,sipma,zm}@theory.stanford.edu

Abstract. An invariant of a system is a predicate that holds for every reachable state. In this paper, we present techniques to generate invariants for hybrid systems. This is achieved by reducing the invariant generation problem to a constraint solving problem using methods from the theory of ideals over polynomial rings. We extend our previous work on the generation of algebraic invariants for discrete transition systems in order to generate algebraic invariants for hybrid systems. In doing so, we present a new technique to handle consecution across continuous differential equations. The techniques we present allow a trade-off between the complexity of the invariant generation process and the strength of the resulting invariants.

1 Introduction

Hybrid Systems are reactive systems that combine discrete mode changes with the continuous evolution of the system variables, specified in the form of differential equations. The analysis of hybrid systems is an important problem that has been studied extensively both by the control theory, and the formal verification community for over a decade. The most important analysis questions for hybrid systems are those of *safety*, i.e, deciding whether a given property ψ holds in all the reachable states, and the dual problem of *reachability*, i.e, deciding if a state satisfying the given property ψ is reachable. Both these problems are computationally hard — intractable even for the simplest of cases, and undecidable for most practical cases.

In this paper, we provide techniques to generate invariants for hybrid systems. An *invariant* of a hybrid system is a property ψ that holds in all the reachable states of the system. An *inductive assertion* of a hybrid system is an assertion ψ that holds at the initial state of the system, and is preserved by all discrete and continuous state changes of the system. Therefore, any inductive assertion being true of all the reachable states is also an invariant assertion. Furthermore, the standard technique for proving a given assertion φ invariant is to generate an

* This research was supported in part by NSF grants CCR-01-21403, CCR-02-20134 and CCR-02-09237, by ARO grant DAAD19-01-1-0723, by ARPA/AF contracts F33615-00-C-1693 and F33615-99-C-3014, and by NAVY/ONR contract N00014-03-1-0939.

inductive assertion ψ that implies φ. The advantage of an inductive assertion is that it can be checked easily [15]. Therefore, the problem of invariant generation is one of inductive assertion generation. This problem has received wide attention in the program analysis community [13, 6, 7, 19, 17, 5, 2]. The generation of linear inductive assertions for the special case of linear hybrid systems has also been studied [10]. Many other approaches that compute the exact or the approximate reach-set of a given hybrid system, can also be shown to compute reach-sets that are inductive assertions [11, 18, 14].

In this paper, we extend our previous work on non-linear inductive assertion generation [17] for discrete systems, by adapting it to generate invariants for hybrid systems. We use the theory of ideals over polynomials and standard computational techniques in algebraic geometry involving Gröbner bases to provide a technique for computing inductive assertions for a given hybrid system. The key idea behind our technique is that given a template assertion, i.e, a polynomial of bounded degree in the system variables with unknown coefficients, we derive constraints on the unknown coefficients so that any solution to these constraints is an inductive assertion. In order to keep these constraints tractable, however, we consider several restrictions on the nature of an invariant. In particular, we consider stronger conditions for inductiveness than the traditional requirements [15]. Also, we provide conceptually simple techniques to handle the continuous evolution; these techniques require neither a closed form solution to the differential equations nor an approximation thereof.

The key advantage of our technique is that it can construct inductive assertions using less time and space than traditional techniques. Depending on the nature of the consecution condition chosen, our constraint generation technique is linear in the number of modes and discrete transitions, and polynomial in the number of system variables. Furthermore, the constraints generated can range in complexity from the more intractable non-linear constraints requiring quantifier elimination to simple constraints involving only linear equalities. Of course, the more complex constraints can potentially yield stronger invariants than the simpler constraints. This trade-off is useful in practice, and contributes to making the method scale.

The rest of the paper is organized as follows: Section 2 presents our computational model and the basic theory behind ideals and Gröbner bases. Section 3 presents the constraint generation process. The nature of these constraints and their solution techniques are discussed in Section 4. In Section 5, we present some examples demonstrating the application of our techniques. Section 6 concludes with a discussion of the pros and cons.

2 Preliminaries

To model hybrid systems we use hybrid automata [12].

2.1 Computational Model: Hybrid Automata

Definition 1 (Hybrid System) A *hybrid system* Ψ: $\langle V, \mathcal{L}, \mathcal{T}, \Theta, \mathcal{D}, \mathcal{I}, \ell_0 \rangle$ consists of the following components:

- V, a set of real-valued system *variables*. The number of variables($|V|$) is called the *dimensionality* of the system;
- $\mathcal{L}$, a finite set of locations;
- $\mathcal{T}$, a set of (discrete) transitions. Each transition $\tau : \langle \ell_1, \ell_2, \rho_\tau \rangle \in \mathcal{T}$ consists of a prelocation $\ell_1 \in \mathcal{L}$, a postlocation $\ell_2 \in \mathcal{L}$, and an assertion ρ_τ over $V \cup V'$, representing the next-state relation, where V' denotes the values of V in the next state;
- Θ, an assertion specifying the *initial* condition;
- $\mathcal{D}$, a map that maps each location $\ell \in \mathcal{L}$ to a *differential rule* $\mathcal{D}(\ell)$, an assertion over $V \cup \{\dot{v} \mid v \in V\}$. The differential rule at a location specifies how the system variables evolve in that location;
- $\mathcal{I}$, a map that maps each location $\ell \in \mathcal{L}$ to a *location condition (location invariant)*, $\mathcal{I}(\ell)$, an assertion over V;
- $\ell_0 \in \mathcal{L}$, the *initial location.*

Definition 2 (Computation) A computation of a hybrid automaton is an infinite sequence of states $\langle l, \boldsymbol{x} \rangle \in \mathcal{L} \times \mathcal{R}^{|V|}$ 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 the *initiation* condition $l_0 = \ell_0$ and $\boldsymbol{x}_0 \models \Theta$ holds, and for each consecutive state pair $\langle l_i, \boldsymbol{x}_i \rangle$, $\langle l_{i+1}, \boldsymbol{x}_{i+1} \rangle$, one of the two *consecution* conditions below is satisfied.

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 $\delta \geq 0$, and a continuous and differentiable function $f : [0, \delta] \mapsto \mathcal{R}^n$, such that f evolves from $\boldsymbol{x}_i$ to $\boldsymbol{x}_{i+1}$ according to the differential rule at location ℓ, while satisfying the location condition $\mathcal{I}(\ell)$. Formally,

1. $f(0) = \boldsymbol{x_1},\ f(\delta) = \boldsymbol{x_2},$ and $(\forall\ t \in [0, \delta]),\ f(t) \models \mathcal{I}(\ell)$,
2. $(\forall t \in [0, \delta)),\ \left\langle f(t), \dot{f}(t) \right\rangle \models \mathcal{D}(\ell)$.

Example 1 (Bouncing Ball). Figure 1 shows a graphical representation of the following hybrid system, representing a ball bouncing on a soft floor ($y = 0$):

$$\begin{array}{ll}
V & = \{y, v_y, \delta\} \\
\mathcal{L} & = \{l\}, \\
\mathcal{T} & = \{\tau\}, where,\ \tau = \left\langle l, l, \begin{bmatrix} \delta > 0\ \wedge\ y = 0\ \wedge\ y' = y\ \wedge \\ v_y' = -\frac{v_y}{2}\ \wedge\ \delta' = 0 \end{bmatrix} \right\rangle \\
\Theta & = (y = 0\ \wedge\ v_y = 16\ \wedge\ \delta = 0) \\
\mathcal{D}(l) & = \left(\dot{y} = v_y\ \wedge\ \dot{v_y} = -10\ \wedge\ \dot{\delta} = 1 \right) \\
\mathcal{I}(l) & = (y \geq 0)
\end{array}$$

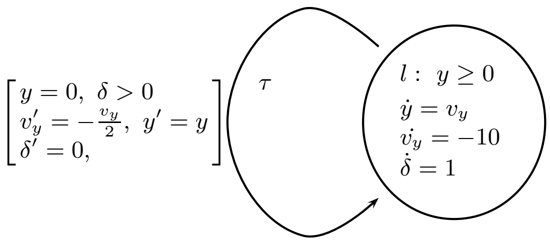

Fig. 1. The hybrid automaton for a bouncing ball

The variable y represents the position of the ball, v_y represents its velocity, and δ denotes the time elapsed since its last bounce. A bounce is modelled by the transition τ, in which the velocity v_y of the ball is halved and the ball reverses direction.

Definition 3 (Invariant) An *invariant* of a hybrid system Ψ at a location ℓ is an assertion ψ such that for any reachable state $\langle \ell, \boldsymbol{x} \rangle$ of Ψ, $\boldsymbol{x} \models \psi$.

Definition 4 (Inductive Assertion Map) An *inductive assertion map* η is a map that associates with each location $\ell \in \mathcal{L}$ an assertion $\eta(\ell)$, satisfying the following requirements:

Initiation $\Theta \models \eta(\ell_0)$,

Discrete Consecution For each discrete transition $\tau : \langle \ell_1, \ell_2, \rho \rangle$, starting from a state satisfying $\eta(\ell_1)$, ρ_τ establishes $\eta(\ell_2)$:

$$\eta(\ell_1) \ \wedge \ \rho_\tau \models \eta(\ell_2)'$$

Continuous Consecution For every location $\ell \in \mathcal{L}$, and states $\langle \ell, \boldsymbol{x_1} \rangle$, $\langle \ell, \boldsymbol{x_2} \rangle$ such that $\boldsymbol{x}_2$ evolves from $\boldsymbol{x}_1$ according to the differential rule $\mathcal{D}(\ell)$ at ℓ, if $\boldsymbol{x_1} \models \eta(\ell)$ then $\boldsymbol{x_2} \models \eta(\ell)$.

An assertion φ is *inductive* if the assertion map that maps each location to φ is an inductive assertion map. It is easy to see that an inductive assertion is an invariant. However, an invariant assertion is not necessarily inductive.

Example 2. For the hybrid system in Example 1, the assertion $y = v_y\delta - 5\delta^2$ is an inductive invariant assertion. On the other hand, the assertion $v_y \leq 16$ is an invariant but not inductive.

2.2 Algebraic Assertions

Our target invariants are algebraic assertions. The invariant generation method is based on the theory of ideals in polynomial rings and makes use of Gröbner bases. We will give a brief overview of the relevant concepts in this theory. A more detailed description can be found in our previous work [17] or in standard textbooks [8, 16, 1].

Notation The set of reals is denoted by $\mathcal{R}$ and the complex numbers by $\mathcal{C}$. The set of polynomials over variables $x_1, \ldots, x_n$, with coefficients in the field K is denoted by $K[x_1, \ldots, x_n]$ or $K[\boldsymbol{x}]$.

Definition 5 (Algebraic Assertion) An *algebraic assertion* over $\mathcal{R}[x_1, \ldots, x_n]$ is an assertion of the form $\bigwedge_{i=1}^{m}(p_i(\boldsymbol{x}) = 0)$, where each polynomial $p_i \in \mathcal{R}[\boldsymbol{x}]$ for $1 \leq i \leq m$. The same algebraic assertion can alternatively be described by the polynomial set $\{p_1, \ldots, p_m\}$.

Definition 6 (Algebraic Hybrid Systems) An *algebraic hybrid system* is a hybrid system $\Psi : \langle V, \mathcal{L}, \mathcal{T}, \Theta, \mathcal{D}, \mathcal{I}, \ell_0 \rangle$, where, for each transition $\tau : \langle \ell_1, \ell_2, \rho_\tau \rangle \in \mathcal{T}$, the relation ρ_τ is an algebraic assertion, and the initial condition Θ, and the location conditions $\mathcal{I}(\ell)$ are algebraic assertions. Furthermore each rule $\mathcal{D}(\ell)$ is an algebraic assertion of the form $\bigwedge_i \dot{x}_i = p_i(x_1, \ldots, x_n)$.

Definition 7 (Ideal) An *ideal* $I \subseteq \mathcal{R}[x_1, \ldots, x_n]$ is a set of polynomials with the following properties:

- $0 \in I$,
- If $p_1, p_2 \in I$ then $p_1 + p_2 \in I$,
- If $p \in I$ and $q \in \mathcal{R}[\boldsymbol{x}]$ then $pq \in I$.

The ideal generated by a set $P = \{p_1, \ldots, p_m\} \subseteq \mathcal{R}[\boldsymbol{x}]$, $n \geq 0$, of polynomials is written as

$$\langle P \rangle = \{g_1 p_1 + \cdots + g_m p_m \ \mid \ p_1, \ldots, p_m \in P,\ g_1, \ldots, g_m \in \mathcal{R}[\boldsymbol{x}]\}$$

Ideal I is finitely generated if $I = \langle P \rangle$ for some finite set P, called the *basis* of I. It can be shown that any ideal in $\mathcal{R}[x_1, \ldots, x_n]$ is finitely generated.

An ideal can be viewed as representing the polynomial consequences of a finite set of polynomials. It contains all polynomials whose zeroes are a superset of the common zeroes of the polynomials in the basis set.

Theorem 1. *Let I be an ideal generated by the basis $\{p_1, \ldots, p_m\}$. Then for any polynomial p, if $p \in I$ then $\{p_1 = 0, \ldots, p_m = 0\} \models (p = 0)$.*

Proof. Since $p \in I$, $p = g_1 p_1 + \ldots g_m p_m$ for some $g_1, \ldots, g_m$. Therefore, if for some $\boldsymbol{z} \in \mathcal{C}^n$, $p_1(\boldsymbol{z}) = 0 \ \wedge \ \cdots \ \wedge \ p_m(\boldsymbol{z}) = 0$ then $p(\boldsymbol{z}) = 0$.

Theorem 1 states that in order to establish the entailment $\varphi \models (p = 0)$, it is sufficient to test membership of p in the ideal generated by φ. In the remainder of this section we will describe a method for testing ideal membership. The converse also holds (under some restrictions), known as *Hilbert's Nullstellensatz* [8], but is not relevant to the soundness of our method.

Let $X = \{x_1, \ldots, x_n\}$ be a set of variables. A *monomial* over X is of the form $x_1^{r_1} x_2^{r_2} \cdots x_n^{r_n}$, where each $r_i \in N$. A *term* is of the form $c \cdot p$ where $c \in \mathcal{R}$

and p is a monomial. Our technique assumes the presence of a linear ordering among terms that satisfies certain criteria. An example of such an ordering is the *lexicographic ordering.* Given a linear ordering $\prec$ on the variables in $X($ $x_1 \succ x_2 \succ \cdots \succ x_n)$, the lexicographic extension $\prec_{lex}$ is the lexicographic ordering on the tuple $\langle r_1, \ldots, r_n \rangle$ corresponding to a term $x_1^{r_1} \cdots x_n^{r_n}$. Given a polynomial g, we define its *lead term* (denoted $\mathrm{LT}(g)$) to be the largest among all its terms w.r.t. a given term-ordering.

Definition 8 (Reduction) Let $f,\ g$ be polynomials, with a term-ordering $<$. The reduction relation over polynomials, $\xrightarrow{g}$ is defined as: $f \xrightarrow{g} f'$ iff there exists term p in f s.t. $\mathrm{LT}(g)$ divides p, where, $f' = f - \frac{p}{\mathrm{LT}(g)} g$

The reduction cancels out the term p that was selected. If no such reduction can be made, then f is said to be a normal-form w.r.t. $\xrightarrow{g}$, denoted $g \in \mathrm{NF}_P(f)$. For a finite set P of polynomials, $f \xrightarrow{P} f'$ iff $(\exists g \in P)\ f \xrightarrow{g} f'$. The transitive closure of $\xrightarrow{P}$ is denoted $\overset{P}{\twoheadrightarrow}$. The reduction $\xrightarrow{P}$ can be shown to be terminating for all P. The reduction is *confluent* if every polynomial can be shown to reduce to a unique normal form [1].

Theorem 2 (Ideal Membership). *Let $I = \langle P \rangle$ be an ideal and f be a polynomial. If $f \overset{P}{\twoheadrightarrow} 0$ then $f \in I$.*

Example 3. Assume a set of variables x, y, z with ordering $x > y > z$. Let $I = \left\langle f : x^2 - y,\ g : y - z,\ h : x + z \right\rangle$ and $p = x^2 - y^2$. The lead term in the polynomial $f :\ x^2 - z$ is x^2, which divides the term $t :\ x^2$ in p. Thus, $p \xrightarrow{f} p'$, where

$$p' = \underbrace{(x^2 - y^2)}_{p} - \underbrace{\frac{x^2}{x^2}}_{\frac{t}{\mathrm{LT}(f)}} \underbrace{(x^2 - y)}_{f} = (-y^2 + y)$$

The following sequence of reductions shows the membership of p in I

$$p \xrightarrow{h} -zx - y^2 \xrightarrow{h} z^2 - y^2 \xrightarrow{g} -yz + z^2 \xrightarrow{g} -z^2 + z^2 \equiv 0$$

thus $p \overset{I}{\twoheadrightarrow} 0$ and hence $p \in I$. However, the reduction sequence

$$p \xrightarrow{f} -y^2 + y \xrightarrow{g} -yz + y \xrightarrow{g} -z^2 + y \xrightarrow{g} -z^2 + z$$

reaches a normal-form without showing the ideal membership.

For an arbitrary ideal basis P, the reduction relation $\xrightarrow{P}$ may not be confluent, and hence, may not provide a decision procedure for ideal membership. However, given an ideal I, there is a special basis G, called the Gröbner basis, such that $I = \langle G \rangle$, and the reduction relation $\xrightarrow{G}$ is confluent.

Theorem 3 (Gröbner Basis). *Let $I = \langle P \rangle$ be an ideal and f be a polynomial. Let G be the Gröbner basis of I. Then $f \overset{G}{\twoheadrightarrow} 0$ iff $f \in I$.*

A proof of this theorem can be found in any standard text or survey on this topic [8, 16]. The standard algorithm for computing the Gröbner basis of an ideal is known as the *Buchberger algorithm* with numerous implementations [20].

Example 4. Consider again the ideal $I = \left\langle x^2 - y,\ y - z,\ x + z \right\rangle$ from Example 3. The Gröbner basis for I is $G = \left\langle z^2 - z,\ y - z,\ x + z \right\rangle$. With this basis, every reduction of $p:\ x^2 - y^2$ will yield a normal form 0.

2.3 Algebraic Templates

Our technique for invariant generation aims to find polynomials that satisfy certain properties, namely the conditions for invariance. The method starts with a candidate invariant that is a generic polynomial template with coefficients that are linear expressions over a set of template variables. Satisfaction of the desired properties then imposes constraints on the template variables, and the solution to these constraints provides the coefficients of the target invariant.

We give a brief overview how to extend the theory of ideals over polynomial rings to templates. A more detailed description can be found in [17].

Definition 9 (Template) Let A be a set of *template variables* and $\mathcal{L}(A)$ be the domain of all *linear expressions* over variables in A of the form $c_0 + c_1 a_1 + \ldots + c_n a_n,\ c_i \in \mathcal{R}$. A *template* over A, X is a polynomial in $\mathcal{L}(A)[\boldsymbol{x}]$. An A-*environment* is a map α that assigns a real value to each variable in A, and by extension, maps each expression in $\mathcal{L}(A)$ to a real value, and each template in $\mathcal{L}(A)[x_1, \ldots, x_n]$ to a polynomial in $\mathcal{R}[x_1, \ldots, x_n]$.

Example 5. Let $A = \{a_1, a_2, a_3\}$, hence $\mathcal{L}(A) = \{c_0 + c_1 a_1 + c_2 a_2 + c_3 a_3 \mid c_0, \ldots, c_3 \in \mathcal{R}\}$. An example template is $(2a_2 + 3)x_1 x_2^2 + (3a_3)x_2 + (4a_3 + a_1 + 10)$. The environment $\alpha \equiv \langle a_1 = 0, a_2 = 1, a_3 = 2 \rangle$, maps this template to the polynomial $5x_1 x_2^2 + 6x_2 + 18$.

The reduction $\xrightarrow{g}$ for polynomials can be extended to templates as follows:

Definition 10 (Reduction of Templates) Let p be a polynomial in $\mathcal{R}[x_1, \ldots, x_n]$ and f, f' be templates over A and $\{x_1, \ldots, x_n\}$. The reduction relation is defined as: $f \xrightarrow{p} f'$ iff the lead term $\textsc{lt}(p)$ divides a term $c(a_0, \ldots, a_m) \cdot t$ in f and $f' = f - \frac{c \cdot t}{\mathrm{LT}(p)} p$

Example 6. Let p be the polynomial $x^2 - y$, with $\textsc{lt}(p) = x^2$. Consider the template $f:\ ax^2 + by^2 + cz^2 + dz + e$. The lead-term of p ($\textsc{lt}(p)$) divides the term ax^2 in f. Therefore, $f \xrightarrow{p} f'$, where

$$f' : (ax^2 + by^2 + cz^2 + dz + e) - \frac{ax^2}{x^2}(x^2 - y) = by^2 + cz^2 + dz + e + ay$$

In [17] it is shown that confluence of the reduction relation w.r.t. Gröbner bases extends to templates in the natural way.

The heart of our invariant generation method is to establish the conditions on the template variables that identify all A-environments for which the corresponding template instance belongs to a given ideal. Recall that a polynomial p belongs to an ideal I iff its normal form w.r.t. a Gröbner basis for I is identically zero.

Theorem 4. *A polynomial $p(x_1, \ldots, x_n)$ is zero for all the possible values of $x_1, \ldots, x_n$ iff all its coefficients are identically zero.*

Thus given a template f and an ideal I with Gröbner basis G all instances of the template belong to I if the coefficients of all terms in the normal from of f are zero.

Example 7. Consider the template $f = a_1x^2 + a_2y^2 + a_3xy + a_4y$ and the ideal I given by the Gröbner basis $\langle x^2 - 2y, y^2\rangle$. The normal form of f is $\mathrm{NF}_G = (2a_1 + a_4)y + a_3xy$. Thus for all A-environments for which $2a_1 + a_4 = 0$ and $a_3 = 0$ the corresponding template instance belongs to $\mathcal{I}$.

3 Constraint-Generation

Our invariant generation algorithm consists of the following steps:

1. fix a template map for the candidate invariant;
2. encode the conditions for invariance as an ideal-membership question;
3. derive the constraints on the template variables that guarantee the appropriate ideal-membership from (2);
4. solve the constraints to obtain the desired class of invariants.

In this section we describe and illustrate the the first three steps; the last step is presented in section 4.

3.1 Template Map

The first step in our method is to fix the shape of the desired invariants. Let $A = \{a_1, a_2, \ldots\}$ be a set of template variables and let Ψ be an algebraic hybrid system with location set $\mathcal{L} = \{\ell_1, \ldots, \ell_m\}$ and variables $V = \{x_1, \ldots, x_n\}$. A generic degree-k template over A and V is the sum of all monomials of degree k or less, written as

$$\sum_{i_1+\cdots+i_n \leq k} a_{\langle i_1, i_2, \ldots, i_n\rangle} x_1^{i_1} \cdots x_n^{i_n}$$

with a total of $\binom{n+k}{k}$ terms, and as many template variables. For example, a degree-2 template for 5 variables has 21 terms.

A *template map* η associates each location ℓ with a template. For maximum generality, the template variables in the templates should be all different. However, templates for different locations may have different degree.

Example 8. For the system introduced in Example 1 we fix the template map η as follows:

$$\eta(l) = a_1 y^2 + a_2 v_y^2 + a_3\delta^2 + a_4 y v_y + a_5 v_y\delta + a_6 y\delta + a_7 y + a_8 v_y + a_9\delta + a_{10}$$

with the objective to identify the values of the coefficients $a_1 \dots a_{10}$ for which the assertion

$$a_1 y^2 + a_2 v_y^2 + a_3\delta^2 + a_4 y v_y + a_5 v_y\delta + a_6 y\delta + a_7 y + a_8 v_y + a_9\delta + a_{10} = 0$$

is an invariant at location l.

3.2 Encoding Invariance Conditions

The second step in our technique is the encoding of the conditions for invariance as an ideal membership statement. The idea is, given a template $\eta(\ell)$, to recast the invariance conditions in the form

$$\eta(\ell) \in \langle p_1, \dots, p_k\rangle$$

where $p_1, \dots, p_k$ are appropriate polynomials representing the condition. This is equivalent to $\mathrm{NF}_G(\eta(\ell)) \equiv 0$, where G is the Gröbner basis of $\langle p_1, \dots, p_k\rangle$.

Initiation: The initiation condition, $\Theta \models (\eta(\ell_0) = 0)$, is encoded by $\eta(\ell_0) \in \langle\Theta\rangle$, that is, the template must belong to the ideal generated by the initial condition, and thus $\mathrm{NF}_\Theta(\eta(\ell_0)) \equiv 0$.

Example 9. The initial condition for the bouncing ball example (Example 1) is $(y = 0,\ v_y = 16,\ \delta = 0)$. Taking the template from Example 8 the normal form w.r.t. Θ is $\mathrm{NF}(\eta(l_0)) = a_{10} + 256a_2 + 16a_8$. Hence the constraint corresponding to initiation is $a_{10} + 256a_2 + 16a_8 = 0$.

Discrete Consecution: The consecution condition states that the invariant map must be preserved by all transitions, that is, for each transition $\langle\ell_1, \ell_2, \rho\rangle$, $(\eta(\ell_1) = 0) \wedge \rho \models (\eta(\ell_2)' = 0)$ must hold. Encoding this exactly would require the reduction of one template $(\eta(\ell_2))$ w.r.t. to another template $(\eta(\ell_1))$. As noted in our previous work [17], this leads to complex constraints which are hard to solve in general. As an alternative we propose to use stronger conditions for consecution that imply the original consecution condition, but avoid the template in the antecedent. It is easily shown that this is sound [17]. However, it may sacrifice completeness, because some invariant may satisfy the general condition of consecution but not the stronger condition.

Figure 2 shows the different consecution conditions together with their encodings. The first condition (LC) states that the transition simply establishes the invariant at the post location, without any assumptions on the precondition. Invariants that can be established in this way are also known as *local* invariants or *reaffirmed* invariants. The constant-value (CV) condition states that the value of the polynomial at the prelocation $(\eta(\ell_1))$ and post location $(\eta(\ell_2))$ is not

Name	Condition	Encoding
Local (LC)	$\rho \models (\eta(l_2)' = 0)$	$\text{NF}_\rho(\eta(l_2)') \equiv 0$
Constant-Value(CV)	$\rho \models (\eta(l_1) = \eta(l_2)')$	$\text{NF}_\rho(\eta(l_1) - \eta(l_2)') \equiv 0$
Constant-Scale(CS)	$(\exists\ \lambda)\ \rho \models (\eta(l_2)' = \lambda\eta(l_1))$	$(\exists\lambda)\ \text{NF}_\rho(\eta(l_2)' - \lambda\eta(l_1)) \equiv 0$
Polynomial-Scale(PS)	$(\exists\ f)\ \rho \models (\eta(l_2)' = f \cdot \eta(l_1))$	

Fig. 2. Consecution Conditions for Algebraic Templates

changed by the transition. Hence, if it is zero before the transition, then it will be zero after the transition, thus preserving the invariant map. The constant-scale condition states that the value of the polynomial may only change by a constant factor, λ. The last condition states that the transition may change the value of the polynomial by a polynomial factor. In all these cases, if the value of the polynomial is zero before the transition is taken, it will be zero after-wards. In the last two cases new unknowns, namely λ and f are added to the constraint solving problem, generally rendering the constraint problem non-linear, as we will see in the next section.

The encodings for the consecution conditions involve the system variables and the primed system variables. To ensure that the primed variables are eliminated as much as possible, a variable ordering must be chosen such that $V' > V$.

Example 10. The transition relation for the discrete transition τ in Example 1 is

$$\left[\delta > 0\ \wedge\ y = 0\ \wedge\ y' = y\ \wedge\ \ v_y' = -\tfrac{v_y}{2}\ \wedge\ \delta' = 0\right]$$

Omitting the conjunct $\delta > 0$ to make the transition relation algebraic (note that it is sound to weaken the antecedent), the reduction of the template given in Example 8 according to local consecution yields a normal form

$$\text{NF}_\rho(\eta'(\ell)) \;=\; \frac{a_2}{4}v_y^2 - \frac{a_8}{2}v_y + a_{10}$$

Reduction of the same template according to the constant scale consecution condition leads to the normal form

$$\text{NF}_\rho(\eta'(\ell) - \lambda\eta(\ell)) \;=\; \frac{4a_2\lambda - a_2}{4}v_y^2 - a_5\lambda v_y\delta - \frac{2a_8\lambda + a_8}{2}v_y - a_3\lambda\delta^2 - a_9\lambda\delta - a_{10}(\lambda - 1)$$

Continuous Consecution: The continuous consecution condition states that if the invariant holds at some state $\langle \ell, \boldsymbol{x_1} \rangle$ then it must hold at any state $\langle \ell, \boldsymbol{x_2} \rangle$ where $\boldsymbol{x_2}$ can be reached from $\boldsymbol{x_1}$ according to the differential rule $\mathcal{D}(\ell)$, while satisfying $\mathcal{I}(\ell)$. As in the discrete case, encoding this exactly is not practical, and therefore we impose stronger conditions, shown in Figure 3.

The first condition states that the value of the polynomial is constant throughout the continuous move, expressed by the condition that the derivative of the template invariant with respect to time is zero, thus guaranteeing that the assertion is preserved. Noting that the system variables are functions of time only,

Name	Condition	Encoding
Constant Value (CV)	$\mathcal{I}(\ell) \models \dot{\eta(\ell)} = 0$	$\text{NF}_{\mathcal{I}(\ell)}(\dot{\eta(\ell)}) \equiv 0$
Constant Scale (CS)	$(\exists\, \lambda)\ \mathcal{I}(\ell) \models \dot{\eta(\ell)} - \lambda\eta(\ell) = 0$	$(\exists\lambda)\text{NF}_{\mathcal{I}(\ell)}(\dot{\eta(\ell)} - \lambda\eta(\ell)) \equiv 0$

Fig. 3. Continuous consecution conditions

the derivative of the template can be obtained by the chain rule as follows:

$$\dot{\eta(\ell)} = \sum_i \left(\frac{\partial \eta(\ell)}{\partial x_i} \dot{x_i} \right)$$

Recall that in algebraic hybrid systems the differential rule is a conjunction of the form $\bigwedge_i \dot{x_i} = p_i(x_1, \ldots, x_n)$, yielding the following template for $\dot{\eta(\ell)}$:

$$\dot{\eta(\ell)} = \sum_i \left(\frac{\partial \eta(\ell)}{\partial x_i} p_i(x_1, \ldots, x_n) \right)$$

The second condition, CS, makes use of the fact that the value of the polynomial is zero, resulting in the more general condition that requires only that the difference between the derivative and a constant factor times the invariant itself be zero.

The encodings are similar to those for the discrete case. In both cases we compute the normal form with respect to the location invariant and equate the result to zero to obtain the constraints.

Example 11. Returning to the bouncing-ball system from Example 1, and the template from Example 8, the derivative of the template is

$$\dot{\eta(l)} = \begin{pmatrix} (2a_1 y + a_4 v_y + a_6 \delta + a_7)\ \dot{y}\ + \\ (2a_2 v_y + a_4 y + a_5 \delta + a_8)\ \dot{v_y}\ + \\ (2a_3 \delta + a_5 v_y + a_6 y + a_9)\ \dot{\delta} \end{pmatrix}$$

which, with $\mathcal{D}(l):\ \dot{y} = v_y\ \wedge\ \dot{v_y} = -10\ \wedge\ \dot{\delta} = 1$ gives

$$\dot{\eta(l)} = \begin{pmatrix} a_4 v_y^2 + 2a_1 y v_y + a_6 \delta v_y + (-20a_2 + a_5 + a_7) v_y + \\ (2a_3 - 10a_5)\delta + (-10a_4 + a_6) y + (a_9 - 10a_8) \end{pmatrix}$$

For this system the location condition $\mathcal{I}(l)$ does not have any algebraic conjuncts and therefore $\text{NF}_{\mathcal{I}(l)}(\dot{\eta(l)}) = \dot{\eta(l)}$.

3.3 Deriving the constraints

The constraints on the template coefficients $a_1, \ldots, a_n$ are derived by equating to zero the coefficients of the normal forms of the template w.r.t. the initial condition, and the consecution conditions for all discrete transitions and continuous moves. By Theorem 4 the solutions to these constraints provide all combinations of values of the template coefficients for which the corresponding assertion satisfies the imposed conditions.

Condition	Restriction	Constraint types
Initiation		linear equalities
Consecution	Local (LC)	linear equalities
	Constant Value (CV)	linear equalities
	Constant Scale (CS)	eigenvalue problems
	Polynomial Scale (PS)	non-linear algebraic

Fig. 4. Constraints obtained from different conditions for inductive assertions

Example 12. Consider again the system presented in Example 1 and the template of Example 8. From above we have that the normal forms representing the initial condition, discrete consecution (LC) of τ and continuous consecution (CV) at location ℓ are

$$\begin{array}{ll} \mathrm{NF}(\eta(l_0)) & = a_{10} + 256a_2 + 16a_8 \\ \mathrm{NF}_\rho(\eta'(\ell)) & = \frac{a_2}{4}v_y^2 - \frac{a_8}{2}v_y + a_{10} \\ \mathrm{NF}_{\mathcal{I}(\ell)}(\dot{\eta(l)}) = & \begin{pmatrix} a_4v_y^2 + 2a_1yv_y + a_6\delta v_y + (-20a_2 + a_5 + a_7)v_y + \\ (2a_3 - 10a_5)\delta + (-10a_4 + a_6)y + (a_9 - 10a_8) \end{pmatrix} \end{array}$$

yielding the set of constraints

$$\begin{array}{l} a_{10} + 256a_2 + 16a_8 = 0,\ \ a_2 \ = \ a_8 \ = \ a_{10} \ = a_4 \ = \ a_1 \ = \ a_6 \ = \ 0 \\ -20a_2 + a_5 + a_7 = 0,\ a_3 - 5a_5 = 0,\ a_6 - 10a_4 = 0,\ a_9 - 10a_8 = 0 \end{array}$$

A solution to this set of constraints is presented in the next section.

4 Solving Constraints

The encodings presented in the previous section can generate several types of constraints. Figure 4 shows the different types obtained for the various conditions and encodings. Solution techniques for linear inequalities are well understood and tend to be computationally inexpensive. For example Gaussian elimination can be done efficiently in polynomial time. The set of solutions is finitely generated by a set of basis vectors. On the other hand, techniques for solving non-linear equalities are complex, either requiring specialized techniques as in the case of eigenproblems, or generic elimination techniques such as quantifier elimination over complex or real numbers [3].

The non-linearities in the constraints are due to the use of scale parameters (λ) or parametric polynomials (in PS consecution). The constraints corresponding to constant scale (CS) consecution yield a generalized eigenvalue problem of the form $\boldsymbol{Ax} = \lambda \boldsymbol{Bx}$. Solution techniques to these problems may be numerical or symbolic. Numerical solution techniques are faster and more easily implemented. However errors in numerical computation may create errors in the final result, and plugging in an inaccurate value for λ will yield only the trivial solution. On the other hand symbolic techniques depend on polynomial root solving

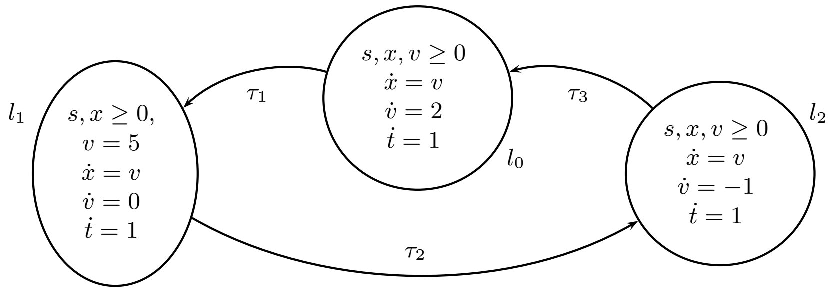

Fig. 5. Hybrid Automaton for the Train System

techniques which work only in special cases. In our experience the use of numerical techniques are adequate provided the values obtained for λ are rational or algebraic of a low degree.

General non-linear constraints are much harder to solve than linear constraints. For small to medium size systems these constraints can be handled using a combination of simplification and quantifier elimination, with the help of a tool suited to low degree polynomials such as REDLOG[9]. For higher-degree polynomials generic quantifier elimination techniques over the reals are required, which have been shown to be successful for small sized problems [4].

Example 13. The generator of the solutions to the linear equality constraints derived in example 12 is given by

$$a_5 = 1, a_3 = 5, a_7 = -1, a_1 = a_2 = a_4 = a_6 = a_8 = a_9 = a_{10} = 0$$

corresponding to the inductive assertion $-y + 5\delta^2 + v_y\delta = 0$.

5 Applications

We demonstrate the results of our technique on some application examples.

Train System: Figure 5 shows a hybrid automaton modeling a train accelerating (location l_0), travelling at constant speed (l_1), and decelerating (l_2). The system has three continuous variables: x, the position of the train, v, the train's velocity, and t, a masterclock. The system has one discrete variablem s, representing the number of stops made so far. The initial condition is given by $x = s = t = v = 0$. There are three discrete transitions, τ_1, τ_2, and τ_3, with transition relations

$$\begin{aligned}
&\rho_{\tau_1} : v = 5 \\
&\rho_{\tau_2} : \mathit{true} \\
&\rho_{\tau_3} : v = 0 \ \wedge \ s' = s + 1 \ \wedge \ t' = t + 2
\end{aligned}$$

Application of our technique resulted in the following assertion map:

$$\begin{aligned}
&\eta(l_0) : v^2 - 4x - 10v + 115s - 20t = 0 \\
&\eta(l_1) : 5v^2 + 4xv + 115vs - 20vt = 0 \\
&\eta(l_2) : 2v^2 + 4x - 20v + 115s - 20t + 75 = 0
\end{aligned}$$

With $v = 5$ at l_1 the assertion $\eta(l_1)$ can be simplified to $4x+115s-20t+25 = 0$.

An analytic argument for the assertion $\eta(l_0)$ is as follows. Consider the system at the state $\langle l_0, \langle s, v, x, t\rangle\rangle$. Each stop s consists of accelerating from 0 to 5 in l_0 and decelerating from 5 to 0 in l_2. The distance covered in these two modes is $\frac{25}{4}$ and $\frac{25}{2}$ respectively. Furthermore, accelerating from 0 to v in l_0 advances the position another $\frac{v^2}{4}$. Hence the total distance travelled is given by $x = s(\frac{25}{4} + \frac{25}{2}) + \frac{v^2}{4} + 5t_{l_1}$, where t_{l_1}, the time spent in location l_1 is given by $t - t_{l_0} - t_{l_2} = t - (v/2 + s(\frac{5}{2})) - (\frac{5}{1}s + 2s)$. Substituting, we obtain the inductive assertion $4x = v^2 - 115s + 20t - 10v$. Similar arguments can be provided for the other locations.

Looping the Loop: Consider a heavy particle on a circular path of radius 2 with an initial angular velocity $\omega = \omega_0$ starting at $x = 2, y = 0$. The differential equation for its motion is given by

$$\begin{aligned}
\dot{x} &= r(\dot{\cos\theta}) = -r(\sin\theta)\dot{\theta} = -y\omega \\
\dot{y} &= r(\dot{\sin\theta}) = x\omega \\
\dot{\omega} &= -\tfrac{g\sin\theta}{r} = -\tfrac{5}{2}x
\end{aligned}$$

Using CV consecution, the quadratic invariants obtained were $x^2 + y^2 = 4$ and $\omega^2 + 5y = \omega_0^2$. The former invariant is a result of our modelling (though not explicitly posed as a location invariant) and the latter is the energy conservation equation. This invariant also establishes that unless $\omega_0 \geq \sqrt{10}$, the particle will be unable to complete a full circle.

6 Conclusion

We have presented a constraint-based technique for generating inductive assertions of hybrid systems. One of the main features of this technique is that it generates constraints without solving the differential equations. These differential equations may be hard to solve symbolically in practice, and may also involve exponentials lying outside the chosen assertion domain. We also avoid the use of over-approximations to these equations, relying instead on the conceptually simpler constant value and constant scale consecutions.

The technique is not guaranteed to generate the exact solution to the reachability problem, because of the relaxations made to maintain tractability. It also requires a degree bound to be specified a priori, the choice of which may be arbitrary. We are investigating strategies for guessing optimal degree bounds. A more serious shortcoming is that, at present, the technique does not handle inequalities. Inequalities frequently occur in the guards and location conditions, and not being able to use them in the constraint generation process may weaken the resulting invariants considerably. However, preliminary investigations indicate that a relatively simple extension may be sufficient to incorporate inequalities.

References

1. Baader, F., and Nipkow, T. *Term Rewriting and* All That. Cambridge University Press, 1998.
2. Bensalem, S., Bozga, M., Fernandez, J.-C., Ghirvu, L., and Lakhnech, Y. A transformational approach for generating non-linear invariants. In *Static Analysis Symposium* (June 2000), vol. 1824 of *LNCS*, Springer Verlag.
3. Bockmayr, A., and Weispfenning, V. Solving numerical constraints. In *Handbook of Automated Reasoning*, A. Robinson and A. Voronkov, Eds., vol. I. Elsevier Science, 2001, ch. 12, pp. 751–842.
4. Collins, G. E., and Hong, H. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation 12*, 3 (sep 1991), 299–328.
5. Colón, M., Sankaranarayanan, S., and Sipma, H. Linear invariant generation using non-linear constraint solving. In *Computer Aided Verification* (July 2003), F. Somenzi and W. H. Jr, Eds., vol. 2725 of *LNCS*, Springer Verlag, pp. 420–433.
6. Cousot, P., and Cousot, R. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *ACM Principles of Programming Languages* (1977), pp. 238–252.
7. Cousot, P., and Halbwachs, N. Automatic discovery of linear restraints among the variables of a program. In *ACM Principles of Programming Languages* (Jan. 1978), pp. 84–97.
8. Cox, D., little, J., and O'Shea, D. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 1991.
9. Dolzmann, A., and Sturm, T. REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bulletin 31*, 2 (June 1997), 2–9.
10. 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.
11. Henzinger, T., and Ho, P.-H. Algorithmic analysis of nonlinear hybrid systems. In *Computer-Aided Verification*, P. Wolper, Ed., LNCS 939. 1995, pp. 225–238.
12. Henzinger, T. A. The theory of hybrid automata. In *Logic In Computer Science (LICS 1996)* (1996), IEEE Computer Society Press, pp. 278–292.
13. Karr, M. Affine relationships among variables of a program. *Acta Inf. 6* (1976), 133–151.
14. Lafferriere, G., Pappas, G., and Yovine, S. Symbolic reachability computation for families of linear vector fields. *J. Symbolic Computation 32* (2001), 231–253.
15. Manna, Z., and Pnueli, A. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.
16. Mishra, B., and Yap, C. Notes on Gröbner bases. *Information Sciences 48* (1989), 219–252.
17. Sankaranarayanan, S., Sipma, H., and Manna, Z. Non-linear loop invariant generation using Gröbner bases. In *ACM Principles of Programming Languages (POPL), to appear* (2004).
18. Tiwari, A. Approximate reachability for linear systems. In *Hybrid Systems: Computation and Control HSCC* (2003), vol. 2623 of *LNCS*, pp. 514–525.
19. Tiwari, A., Ruess, H., Saïdi, H., and Shankar, N. A technique for invariant generation. In *TACAS 2001* (2001), vol. 2031 of *LNCS*, pp. 113–127.
20. Windsteiger, W., and Buchberger, B. Gröbner: A library for computing Gröbner bases based on saclib. Tech. rep., RISC-Linz, 1993.