Constructing Invariants for Hybrid Systems *

Sriram Sankaranarayanan, Henny B. Sipma and Zohar Manna
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract. We present a new method for generating algebraic invariants of hybrid systems. The method reduces the invariant generation problem to a constraint solving problem using techniques from the theory of ideals over polynomial rings. Starting with a template invariant – a polynomial equality over the system variables with unknown coefficients – constraints are generated on the coefficients guaranteeing that the solutions are inductive invariants. To control the complexity of the constraint solving, several stronger conditions that imply inductiveness are proposed, thus allowing 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. Among 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 subclasses, and undecidable for the general case.

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 states of the system, and is preserved by all discrete and continuous state changes. Therefore, any inductive assertion is also an invariant assertion. Furthermore, the standard technique for proving a given assertion φ invariant is to generate an inductive assertion ψ that implies φ. Therefore, the problem of invariant generation is also one of inductive assertion generation. This problem has received wide attention in the

* 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.

program analysis community [18, 9, 10, 34, 28, 8, 4]. The generation of linear inductive assertions for the special case of linear hybrid systems has also been studied [14]. Many other approaches that compute the exact or the approximate reach-set of a given hybrid system can also be shown to compute inductive assertions [15, 32, 19].

In this paper, we extend our previous work on non-linear inductive assertion generation [28] for discrete systems to generate invariants for hybrid systems. We use the theory of ideals over polynomials along with standard computational techniques in algebraic geometry involving Gröbner bases to provide a technique for computing inductive assertions for hybrid systems.

The key idea behind our technique is that given a *template assertion*, i.e, a parametric polynomial with unknown coefficients and of bounded degree in the system variables, we derive constraints on the unknown coefficients so that any solution to these constraints is an inductive assertion. To keep these constraints tractable, we consider several useful restrictions on the nature of an invariant. In particular, we consider stronger conditions for inductiveness than the traditional requirements [20]. Also, we provide conceptually simpler techniques to handle the continuous evolution; these techniques require neither a closed form solution to the differential equations nor an approximation thereof.

Our technique 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 potentially yield stronger invariants than the simpler constraints. This trade-off is useful in practice, and contributes to making the method scale.

Related Work

The verification of safety (invariance and reachability) properties of timed and hybrid automata has been the subject of numerous papers, and has lead to popular verification systems such as Kronos [36], Uppaal [3], Hytech [17], d/dt [1], and CheckMate [31]. Most of this work deals with hybrid systems with piecewise-linear dynamics, although some tools, notably CheckMate, can handle more general dynamics. Non-linear systems have been traditionally dealt with by using finite-state, or linear infinite-state abstractions [15]. Techniques using template properties with undetermined coefficients along with constraint solving have appeared intermittently both in the computer science, and the control theory literature. Techniques for automatic generation of linear equality invariants for Petri-nets are a classic example [23]. Our earlier work on discrete linear systems handles the generation of linear inequality invariants using *Farkas' Lemma* [8, 29]. The work of Forsman et al. [12] uses Gröbner bases and parameterized polynomials to construct *Lyapunov functions* to prove local stability of continuous systems. Their approach works by fixing the form of the desired Lyapunov function, and computing a maximal region over which a function of the given form can establish local stability.

Invariant-generation methods for discrete programs based on algebraic geometry have appeared recently. The work of Müller-Olm and Seidl, presents a backward propagation-based method for non-linear programs without branch conditions [22]. In [7], Colón presents a method that uses forward propagation in the lattice of pseudo ideals of a given degree, to generate invariants of a bounded degree. In [27], Rodriguez-Carbonell and Kapur present a method to generate polynomial invariants for programs that contain only a special class of assignments called *solvable assignments*. In [26], a forward-propagation-based technique is presented, in which statements are represented as transformations on ideals. A widening operator that limits invariants to a given degree is used.

The work of Parillo et al. [24] presents a powerful framework based on real-algebraic geometry and semi-definite programming. Conceptually, this framework can be made to play the same role as that of Gröbner bases in this work. This has already been applied to some aspects of the safety-verification problem by the recent work of Prajna and Jadbabaie using the idea of *barrier certificates* [25]. Ideas similar to the ones contained in this work have appeared independently in the work of Tiwari and Khanna [33].

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

2.1. Computational Model: Hybrid Automata

To model hybrid systems we use hybrid automata [16].

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

- V, a set of real-valued system *variables*. The number of variables ($|V|$) is called the *dimension* of the system. A *state* is an interpretation of V, assigning to each $v \in V$ a real value. The set of all states is denoted by Σ. An *assertion* is a first-order formula over V. A state s satisfies an assertion φ, written $s \models \varphi$, if φ holds on s. We will also write $\varphi_1 \models \varphi_2$ for two assertions φ_1, φ_2 to denote that φ_2 is true at least in all the states in which φ_1 is true.

- $\mathcal{L}$, 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* (also known as a *vector field* or a *flow field*), $\mathcal{D}(\ell)$, of the form $\dot{v_i} = f_i(V)$ for each $v_i \in V$. The differential rule at a location specifies how the system variables evolve in that location.

- $\mathbf{I}$, a map that maps each location $\ell \in \mathcal{L}$ to a *location condition (location invariant)*, $\mathbf{I}(\ell)$, an assertion over V;

- $\ell_0 \in \mathcal{L}$, the *initial location*; we assume that the initial condition satisfies the location invariant at the initial location, that is, $\Theta \models \mathbf{I}(\ell_0)$.

Definition 2 (Computation). A computation of a hybrid system Ψ is an infinite sequence of states $\langle l, \vec{x} \rangle \in \mathcal{L} \times \mathcal{R}^{|V|}$ of the form

$$\langle l_0, \vec{x}_0 \rangle, \langle l_1, \vec{x}_1 \rangle, \langle l_2, \vec{x}_2 \rangle, \ldots$$

where $\vec{x_i}$ are the values assigned to the variables in V, such that

Initiation: the initial state of the computation satisfies the initial condition:

$$l_0 = \ell_0 \quad \text{and} \quad \vec{x}_0 \models \Theta$$

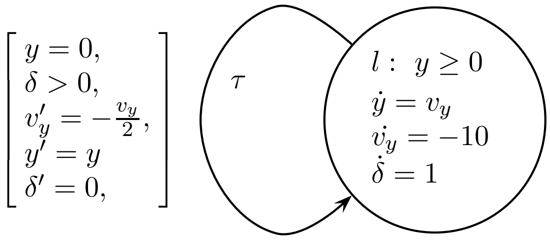


Figure 1. The hybrid automaton for a bouncing ball

Furthermore, for each consecutive state pair $\langle l_i, \vec{x}_i \rangle$, $\langle l_{i+1}, \vec{x}_{i+1} \rangle$, 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 $\langle \vec{x}_i, \vec{x}_{i+1} \rangle \models \rho_\tau$, where the unprimed variables refer to $\vec{x}_i$ and the primed variables to $\vec{x}_{i+1}$, or

Continuous Consecution: $l_i = l_{i+1} = \ell$, and there exists a time interval $\delta > 0$, along with a smooth (continuous and differentiable to all orders) function $f : \ [0, \delta] \mapsto \mathcal{R}^n$, such that f evolves from $\vec{x}_i$ to $\vec{x}_{i+1}$ according to the differential rule at location ℓ, while satisfying the location condition $\mathbf{I}(\ell)$. Formally,

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

A state $\langle \ell, \vec{x} \rangle$ is called a *reachable state* of a hybrid system Ψ if it appears in some computation of Ψ.

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

$$\begin{array}{lcl} V & = & \{y, v_y, \delta\} \\ \mathcal{L} & = & \{l\}, \\ \mathcal{T} & = & \{\tau\}, where, \ \tau = \left\langle l, l, \begin{bmatrix} \delta > 0 \ \wedge \ y = 0 \ \wedge \ y' = y \ \wedge \\ v_y' = -\frac{v_y}{2} \ \wedge \ \delta' = 0 \end{bmatrix} \right\rangle \\ \Theta & = & (y = 0 \ \wedge \ v_y = 16 \ \wedge \ \delta = 0) \\ \mathcal{D}(l) & = & \left(\dot{y} = v_y \ \wedge \ \dot{v_y} = -10 \ \wedge \ \dot{\delta} = 1\right) \\ \mathbf{I}(l) & = & (y \geq 0) \\ \ell_0 & = & l \end{array}$$

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 modeled 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 $\langle \ell, \vec{x} \rangle$ of Ψ, $\vec{x} \models \psi$.

Definition 4 (Inductive Assertion Map). An *inductive assertion map* Ψ is a map that associates with each location $\ell \in \mathcal{L}$ an assertion $\Psi(\ell)$ that holds initially and is preserved by all discrete transitions and continuous flows. More formally an inductive assertion map satisfies the following requirements:

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

Discrete Consecution For each discrete transition $\tau : \langle \ell_1, \ell_2, \rho \rangle$, starting from a state satisfying $\Psi(\ell_1)$, and taking τ leads to a state satisfying $\Psi(\ell_2)$. Formally,

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

where $\Psi(\ell_2)'$ represents the assertion $\Psi(\ell_2)$ with the current state variables $\vec{x}$ replaced by the next state variables $\vec{x}'$.

Continuous Consecution For every location $\ell \in \mathcal{L}$, and states $\langle \ell, \vec{x}_1 \rangle$, $\langle \ell, \vec{x_2} \rangle$ such that $\vec{x}_2$ evolves from $\vec{x}_1$ according to the differential rule $\mathcal{D}(\ell)$ at ℓ, if $\vec{x_1} \models \Psi(\ell)$ then $\vec{x_2} \models \Psi(\ell)$. If $\Psi(\ell)$ is an assertion of the form $f_\ell(\vec{x}) = 0$, for a real-valued smooth function f_ℓ, we can express consecution by the condition

$$\mathbf{I}(\ell)(\vec{x}) \ \wedge \ (f_\ell(\vec{x}) = 0) \models \dot{f}_\ell(\vec{x}) = 0 \ .$$

Note that $\dot{f}_\ell$ denotes the *Lie-derivative* of f_ℓ along the vector field $\mathcal{D}(\ell)$. The condition above is important, since we shall only be interested in inductive assertions of the form $p = 0$ for a polynomial p.

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 and invariant assertion. It can be shown to satisfy all the requisite conditions for being inductive. On the other hand, the assertion $v_y \leq 16$ is an invariant but not inductive. This is because, it does not satisfy consecution for the discrete transition τ.

2.2. Algebraic Assertions and Ideals

We begin by presenting some definitions and results from commutative algebra. Informally, an ideal is a set of equations between polynomials along with all their consequences. Ideals can be used to deduce facts about the set of points that satisfy their underlying equations. Some of the results in this section are stated in general terms without proofs. Details can be found in most standard texts on algebra and algebraic geometry [11, 21].

Let $\mathcal{R}$ be the set of reals and $\mathcal{C}$ be the set of complex numbers obtained as the *algebraic closure* of the reals. Let $V = \{x_1, \ldots, x_n\}$ be a set of variables. The set of polynomials on V, with coefficients from $\mathcal{R}$ ($\mathcal{C}$) is denoted by $\mathcal{R}[x_1, \ldots, x_n]$ ($\mathcal{C}[x_1, \ldots, x_n]$).

Definition 5 (Algebraic Assertion). An *algebraic assertion* ψ is a finite conjunction of polynomial equations denoted by

$$\bigwedge_i p_i(x_1, \ldots, x_n) = 0,$$

where each $p_i \in \mathcal{R}[x_1, \ldots, x_n]$. The *degree* of an assertion is the maximum among the degrees of the polynomials that make up the assertion.

Definition 6 (Algebraic Hybrid Systems). An *algebraic hybrid system* is a hybrid system $\Psi : \langle V, \mathcal{L}, \mathcal{T}, \Theta, \mathcal{D}, \mathbf{I}, \ell_0 \rangle$, where,

1. For each transition $\tau : \langle \ell_1, \ell_2, \rho_\tau \rangle \in \mathcal{T}$, the relation ρ_τ is an algebraic assertion,

2. The initial condition Θ, and the location conditions $\mathbf{I}(\ell)$ are also algebraic assertions, and

3. Each rule $\mathcal{D}(\ell)$ is of the form $\bigwedge_i \dot{x}_i = p_i(x_1, \ldots, x_n)$.

Definition 7 (Variety). A *variety* is the set of points in the complex plane that satisfy an algebraic assertion. Given an assertion $\psi : \bigwedge_i (p_i(\vec{x}) = 0)$, its corresponding variety is defined as

$$\mathbf{Variety}(\psi) = \{\vec{z} \in \mathcal{C}^n \mid (\forall i)\ p_i(\vec{z}) = 0\}.$$

Definition 8 (Ideal). A set $I \subseteq \mathcal{R}[x_1, \ldots, x_n]$ is an ideal, if

1. $0 \in I$,

2. If $p_1, p_2 \in I$ then $p_1 + p_2 \in I$,

3. If $p_1 \in I$ and $p_2 \in \mathcal{R}[x_1, \ldots, x_n]$ then $p_1 p_2 \in I$.

An ideal *generated by* a set of polynomials P, denoted by $((P))$, is the smallest ideal containing P. Equivalently,

$$((P)) = \left\{ g_1 p_1 + \cdots + g_m p_m \;\middle|\; \begin{array}{c} g_1, \ldots, g_m \in \mathcal{R}[x_1, \cdots, x_n], \\ p_1, \ldots, p_m \in P \end{array} \right\}.$$

An ideal I is *finitely generated* if there is a finite set P such that $I = ((P))$. A famous theorem due to Hilbert states that all ideals in $\mathcal{R}[x_1, \ldots, x_n]$ are finitely generated. As a result, algebraic assertions can be seen as the generators of an ideal and vice-versa. Any ideal defines a variety, which is the set of the common zeroes of all the polynomials it contains. This correspondence between ideals and varieties is one of the fundamental observations involving algebraic assertions called the *Hilbert's Nullstellensatz.* We state the relevant direction of this theorem below:

Theorem 1 (Hilbert's Nullstellensatz). *Consider an algebraic assertion ψ. Let $I = ((\psi))$ be the ideal generated by ψ in $\mathcal{R}[x_1, \ldots, x_n]$, and $f(x_1, \ldots, x_n) \in \mathcal{R}[x_1, \ldots, x_n]$. If $f \in I$, then*

$$(\forall \vec{z} \in \mathcal{C}^n)\ \psi(\vec{z}) \Rightarrow (f(\vec{z}) = 0)$$

i.e, $\psi(\vec{x}) \models (f(\vec{x}) = 0)$.

Proof. Let $\psi \equiv f_1 = 0 \ \wedge \ \cdots \wedge \ f_m = 0$ and $f \in I$. Hence, we can express f as

$$f = g_1 f_1 + \cdots + g_m f_m$$

for some $g_1, \ldots, g_m \in \mathcal{R}[x_1, \ldots, x_n]$. Let $\vec{z} \in \mathcal{C}^n$ be such that $\psi(\vec{z})$ holds. Then, $f(\vec{z}) = \sum_{i=1}^{m}(g_i(\vec{z}) f_i(\vec{z})) = 0$, since each $f_i(\vec{z}) = 0$, and therefore

$$(\forall \vec{z} \in \mathcal{C}^n)\ \psi(\vec{z}) \Rightarrow (f(\vec{z}) = 0).$$

□

The theorem shows that membership of a polynomial p in an ideal I leads to the semantic entailment of $p = 0$ by the variety induced by I. Hence, an ideal is (informally) a *Consequence-Closed* set of polynomials. While a similar statement can be made in the converse, it is not relevant to the soundness of our technique, and therefore we omit it from the discussion. Note that even though the ideals are defined in $\mathcal{R}[x_1, \ldots, x_n]$, the varieties along with the entailments are defined over the complex numbers, which subsume the reals. This is a technicality that arises from the fact that the reals are not algebraically closed.

Remark. Figure 2 shows an instructive comparison between the deduction of consequences of linear equalities and that of polynomial

Linear Equalities Hyperplanes	Polynomial Equalities Varieties
$$\begin{array}{r\|lcl} \lambda_1 & e_1 & = & 0 \\ \vdots & & \vdots & \\ \lambda_m & e_m & = & 0 \\ \hline & e & = & 0 \end{array}$$	$$\begin{array}{r\|lcl} g_1 & p_1 & = & 0 \\ \vdots & & \vdots & \\ g_m & p_m & = & 0 \\ \hline & p & = & 0 \end{array}$$
$e = \lambda_1 e_1 + \cdots + \lambda_m e_m$	$p = g_1 p_1 + \cdots + g_m p_m$
$e \in \textsc{linear}(e_1, \ldots, e_m)$	$p \in ((p_1, \ldots, p_m))$

Figure 2. Comparison between multivariate linear equations and multivariate polynomial equations

equalities, ignoring some technical details. A given set of linear equalities, $e_1 = 0, \ldots, e_m = 0$, entails a new linear equality $e = 0$ iff e can be written as a linear combination of $e_1, \ldots, e_m$. The set of all linear combinations of $e_1, \ldots, e_m$, forms a *linear subspace*. Similarly, for the polynomial equalities, the real multipliers $\lambda_1, \ldots, \lambda_m$, are replaced by arbitrary polynomials $g_1, \ldots, g_m$. The set of all such combinations of $p_1, \ldots, p_m$, is the ideal generated by $p_1, \ldots, p_m$.

Example 3. The assertion $\psi = (p_1 : \ x^2 + 2x + y^2 - 1 = 0 \ \wedge \ p_2 : x^2 - 2x + y^2 - 1 = 0)$ represents the intersection of two circles each with radius $\sqrt{2}$, centered at $(-1, 0)$ and $(1, 0)$, respectively. They intersect at two points $(0, 1)$ and $(0, -1)$. Furthermore, $x = \frac{p_1 - p_2}{4}$, and hence $x \in ((p_1, p_2))$. Geometrically, this states that $p_1 = 0 \wedge p_2 = 0 \models x = 0$, or in other words all the points of intersection of the two circles lie on the y-axis. Also, $x^2 + y^2 - 1 = \frac{p_1 + p_2}{2} \in ((p_1, p_2))$. Therefore, we can deduce that the points satisfying ψ also lie on the circle with radius 1 centered at the origin.

In the remainder of this subsection, we use the concept of ideals to derive a systematic algorithm for determining, given an assertion ψ and a polynomial f, whether $f \in ((\psi))$. We assume that all the polynomials are drawn from $\mathcal{R}[x_1, \ldots, x_n]$, and all consequence relations of the form $\psi_1 \models \psi_2$ are over the domain of complex numbers.

Let $V = \{x_1, \ldots, x_n\}$ be a set of variables. A *monomial* over V is of the form $x_1^{r_1} x_2^{r_2} \cdots x_n^{r_n}$, with $r_i \in N$. The set of monomials is denoted by M. A *term* is of the form $c \cdot p$ where $c \in \mathcal{R}$ and $p \in M$. The set of terms is denoted by *Term*.

Definition 9 (Monomial Orderings). A *monomial ordering* $<$ is a total and strict ordering on M that satisfies the following properties:

1. $(\forall t \in M)\ 1 \leq t$,
2. If $(t_1 \leq t_2)$ then $(\forall t \in M)\ t_1 t \leq t_2 t$.

These orderings can be extended to a non-total ordering over terms by ignoring the coefficients.

Monomial orderings are used to induce a reduction relation over polynomials. Many monomial orderings appear in the literature, the most common being the *lexicographic* and *total-degree lexicographic* orderings. Assuming a linear ordering $\prec$ on the variables in V, $x_1 \prec x_2 \prec \cdots \prec x_n$, the lexicographic extension $\prec_{lex}$ is defined as:

$$x_1^{r_1} \cdots x_n^{r_n} \prec_{lex} x_1^{q_1} \cdots x_n^{q_n} \text{ iff } (\exists i)\ r_i < q_i \ \wedge\ (\forall\ j < i)\ r_j = q_j.$$

The ordering is lexicographic on the tuple $\langle r_1, \ldots, r_n \rangle$ corresponding to a term $x_1^{r_1} \cdots x_n^{r_n}$. The *total-degree* lexicographic ordering is a variant of this order: it first compares monomials by their *total degree*, defined as the sum of the powers of all the variables, and then the lexicographic ordering is used to resolve the tie for terms with the same total degree. In general, the choice of ordering affects the complexity of the algorithms, but such an effect is not addressed in this paper.

Definition 10 (Lead term). Given a polynomial g, its *lead term* (denoted $\mathrm{LT}(g)$) is the largest term in g w.r.t. a given monomial ordering.

Definition 11 (Reduction). Let f, g be polynomials, and $<$ be a monomial ordering. The reduction relation over polynomials, $\xrightarrow{g}$ is defined as: $f \xrightarrow{g} f'$ iff there exists term t in f s.t. $\mathrm{LT}(g)$ divides t, and

$$f' = f - \frac{t}{\mathrm{LT}(g)} g.$$

The effect of the reduction is the cancellation of the term t that was selected. The reduction relation can be viewed as a *term rewriting system* over polynomials [2]. The reduction relation can be extended to a finite set P of polynomials as $f \xrightarrow{P} f'$ iff $(\exists g \in P)\ f \xrightarrow{g} f'$. The reduction relation $\xrightarrow{P}$ is terminating for any finite set of polynomials P, as a direct consequence of the definition of monomial orderings. A normal form f of the reduction relation is a polynomial such that no further reduction of f is possible. The reduction relation is *confluent* if every polynomial reduces to a unique normal form. By $\overset{P}{\twoheadrightarrow}$, we denote the reflexive transitive closure of the relation $\xrightarrow{P}$.

The reduction relation induced by P can be used to check membership of a given polynomial in the ideal generated by P:

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

Proof. The proof proceeds by induction on the length of the derivation. It is trivially true for zero length derivations, since $0 \in I$. Let $f \xrightarrow{P} f' \overset{P}{\twoheadrightarrow} 0$. It follows that $f' = f - tg$, for some suitable term t, and some $g \in P$. Since $f' \in I$ (the induction hypothesis), and $g \in P$, it follows that $f' + tg \in I$. Thus, $f \in I$. □

Example 4. Assume a set of variables x, y, z with a precedence ordering $x \succ y \succ z$. Consider the ideal $I = ((f : x^2 - y,\ g : y - z,\ h : x + z))$, and the polynomial $p : \ x^2 - y^2$. Using the total lexicographic ordering, we have $x^2 > y$, and thus the lead term in f is x^2, which divides the term $t : \ x^2$ in p. Therefore, $p \xrightarrow{f} p'$, 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 the ideal

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

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.

The reduction relation $\xrightarrow{P}$ may not be confluent, as illustrated by the example above. Therefore, Theorem 2 cannot be used to decide membership of a polynomial in the ideal generated by P. Fortunately, for any ideal $I = ((P))$, there exists a special set of generators G such that $I = ((G))$, and the reduction relation $\xrightarrow{G}$ induced by G is confluent. Such a basis for I is variously called the *Gröbner Basis*, or the *Standard Basis* of I.

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

Proof. A proof of this theorem can be found in any standard text or survey on this topic [11, 21]. □

Since the reduction relation is terminating, Theorem 3 provides a decision procedure for ideal membership. We shall use $\text{NF}_G(p)$ to denote the normal form of a polynomial p under $\overset{G}{\twoheadrightarrow}$. The subscript G in $\text{NF}_G(p)$ may be dropped if it is evident from the context. The standard algorithm for computing the Gröbner basis of an ideal is known as the *Buchberger algorithm*. There are numerous implementations of this algorithm available with standard computer-algebra packages and polynomial computation libraries. As an example, the library GROEBNER implements many improvements over the standard algorithm [35].

Example 5. Consider again the ideal from Example 4; $I = ((f : x^2 - y,\ g : y - z,\ h : x + z))$. The Gröbner basis for I is $G = \{f_1 : z^2 - z,\ g : y - z,\ h : x + z\}$. With this basis, every reduction of $p : x^2 - y^2$ will yield the normal form 0.

2.3. Templates

Our technique for invariant generation aims to find polynomials that satisfy certain properties. To represent these sets of polynomials we use templates — polynomials with coefficients that are linear expressions over some set of template variables. We show that the theory of ideals can be naturally extended to templates. In particular, we show that there exist confluent reduction relations on templates that allow the generation of constraints on the template variables, such that the resulting set of polynomials is precisely the set of polynomials that belong to the desired ideal.

Definition 12 (Templates). 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 $c_0 + c_1a_1 + \ldots + c_na_n$, where each c_i is a real-valued coefficient. A *template* over A, V is a polynomial over variables in V with coefficients from $\mathcal{L}(A)$.

Example 6. Let $A = \{a_1, a_2, a_3\}$, and hence

$$\mathcal{L}(A) = \{c_0 + c_1a_1 + c_2a_2 + c_3a_3 \ \mid\ c_0, \ldots, c_3 \in \mathcal{R}\}.$$

The set of templates is the ring $\mathcal{L}(A)[x_1, \ldots, x_n]$. For example,

$$(2a_2 + 3)x_1x_2^2 + (3a_3)x_2 + (4a_3 + a_1 + 10) \ \in\ \mathcal{L}(A)[x_1, \ldots, x_n].$$

Definition 13 (Semantics of Templates). Given a set of template variables A, an A-environment (if A is clear from the context, then simply an environment) is a map α that assigns real values to each variable in A. This map is naturally extended to map expressions in $\mathcal{L}(A)$ to their corresponding values in $\mathcal{R}$, and to map polynomials in $\mathcal{L}(A)[x_1, \ldots, x_n]$ to their corresponding polynomials in $\mathcal{R}[x_1, \ldots, x_n]$.

Example 7. The environment $\alpha \equiv \langle a_1 = 0, a_2 = 1, a_3 = 2\rangle$, maps the template

$$(2a_2 + 3)x_1x_2^2 + (3a_3)x_2 + (4a_3 + a_1 + 10)$$

from Example 6 to the polynomial

$$5x_1x_2^2 + 6x_2 + 18.$$

The reduction relation $\xrightarrow{g}$ for polynomials can be extended to a reduction relation for templates in a natural way.

Definition 14 (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 $c \cdot t$ in f with coefficient $c(a_0, \ldots, a_m)$ and

$$f' = f - \frac{c \cdot t}{\textsc{lt}(p)} p.$$

Remark. The reduction relation is a natural extension of the corresponding reduction relation over polynomials. The definition above defines reductions of templates by *polynomials* in $\mathcal{R}[x_1, \ldots, x_n]$. We shall *not* attempt to define reductions of templates by other templates.

The reduction relation can, in turn, be extended to sets of polynomials to define reduction relation $\xrightarrow{G}$ over templates for sets of polynomials G. Henceforth, we shall use the symbols f, g, with subscripts to denote templates and the symbols h, p, to denote polynomials.

Example 8. 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 $\textsc{lt}(p)$ divides the term ax^2 in f. Therefore, $f \xrightarrow{p} f'$, where

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

Given a template f and an ideal I, the objective is to find those environments α such that $\alpha(f) \in I$. This is achieved by obtaining constraints on the environment variables A such that any solution α satisfies $\alpha(f) \in I$.

The properties of the Gröbner basis reduction relations over polynomials can be extended smoothly to templates. Proofs of these results can be found in the appendix. We first show that the extension of reduction relation is consistent w.r.t. the semantics of templates under any A-environment. The confluence of Gröbner basis reduction relation over templates guarantees that a unique normal form exists for any template (Theorem 8). The template membership theorem (Theorem 10) proves that if $f_1 = \text{NF}_G(f)$ is the normal form of f, then for each environment α, $\alpha(f) \in ((G))$ iff $\alpha(f_1)$ is identically zero.

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

Proof. The proof is available from any standard text on algebra. □

Given a template f and an ideal I with Gröbner basis G, we first compute $\text{NF}_G(f)$, and then equate each coefficient of the normal form to zero to obtain a set of equations over the template variables A. Any solution to this set of equations yields an A-environment α such that $\alpha(f) \in I$ (and conversely).

Example 9. Let I be the ideal $((x^2 - y,\ y - z,\ z + x))$ of Example 4 with Gröbner basis $G = \{x + z,\ y - z, z^2 - z\}$. We are interested in polynomials represented by the template

$$f:\ ax^2 + by^2 + cz^2 + dz + e$$

that are members of the ideal. The normal form of this template is

$$\text{NF}_G(f):\ (a + b + c + d)z + e\ \ .$$

Equating each coefficient in the normal form to zero, we obtain

$$\begin{aligned} &a + b + c + d = 0 \\ &e = 0 \end{aligned}$$

From this, we can generate all instances of the template that belong to I. Some examples are $x^2 - y^2,\ y^2 - z,\ 2x^2 - z^2 - z$. On the other hand, $x^2 + y^2 - z \notin I$, as the coefficients do not satisfy the constraints.

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;
4. solve the constraints to obtain the invariants of the form specified by the template map.

In this section we describe and illustrate 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 10. 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 $a_1 \ldots a_{10}$ 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 involves the encoding of the conditions for invariance as an ideal membership statement. Given a template $\eta(\ell)$, we recast the invariance conditions as an ideal membership problem of the form

$$\eta(\ell) \in ((p_1, \ldots, p_k))$$

where $p_1, \ldots, p_k$, are polynomials over the reals. This is equivalent to $\mathrm{NF}_G(\eta(\ell)) \equiv 0$, where G is the Gröbner basis of $((p_1, \ldots, p_k))$.

Initiation

The initiation condition, $\Theta \models (\eta(\ell_0) = 0)$, is encoded by $\eta(\ell_0) \in ((\Theta))$. This is in turn encoded by computing the Gröbner basis of Θ, and encoding $\mathrm{NF}_\Theta(\eta(\ell_0)) \equiv 0$.

Example 11. The initial condition for the bouncing ball example (Example 1) is $(y = 0,\ v_y = 16,\ \delta = 0)$. Taking the template from Example 10, 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, i.e, 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 [28], 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. However, in doing so we lose the invariants that satisfy the general condition of consecution but not the stronger conditions.

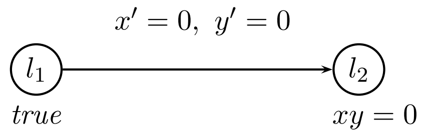
We describe four options for stronger consecution conditions, starting with the strongest. A summary of these conditions, together with their encodings, is shown in Figure 3.

Local Consecution (LC) The local consecution condition states that the transition simply establishes the invariant at the postlocation, without any assumptions on the precondition. Invariants that can be established in this way are also known as *local* invariants or *reaffirmed* invariants.

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

Figure 3. Consecution Conditions for Algebraic Templates

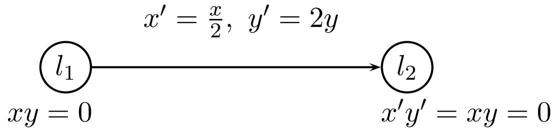
As an example, consider the following discrete transition from l_1 to l_2,



The assertion $xy = 0$ holds immediately after taking the transition regardless of what held before the transition.

Constant Value (CV) The constant-value condition states that the value of the polynomial at the prelocation $(\eta(\ell_1))$ and postlocation $(\eta(\ell_2))$ is not 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.

For example, in the transition



the assertion $xy = 0$ holds immediately after taking the transition if it held before the transition, because the value of xy does not change when the transition is taken.

Constant Scale (CS) The constant-scale condition states that the transition may change the value of the polynomial only by a constant factor, λ. As before, if the value of the polynomial is zero before the transition is taken, it will be zero after the transition is taken.

In the transition

$$\underset{xy = 0}{\textcircled{l_1}} \xrightarrow{\quad x' = y,\ y' = 2x \quad} \underset{x'y' = 2xy = 0}{\textcircled{l_2}}$$

the assertion $xy = 0$ holds immediately after taking the transition if it held before the transition, because the value of xy is doubled every time the transition is taken.

Polynomial Scale (PS) The polynomial scale condition states that the transition may change the value of the polynomial by a polynomial factor, exemplified by the transition

$$\underset{xy = 0}{\textcircled{l_1}} \xrightarrow{\quad x' = y,\ y' = x^3 + x \quad} \underset{x'y' = (x^2 + 1)xy = 0}{\textcircled{l_2}}$$

The assertion $xy = 0$ holds immediately after taking the transition if it held before the transition, because the value of xy is multiplied by $x^2 + 1$ every time the transition is taken.

In all these cases, if the value of the polynomial is zero before the transition is taken, it will be zero afterwards. 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.

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 $V' \succ V$. The notion of CS consecution includes both LC and CV consecutions by substituting $0, 1$ for the parameter λ, while PS consecution includes CS consecution by requiring the multiplier polynomial to be of degree 0.

Lemma 1 (Soundness). *Let η be an inductive assertion map that satisfies any of $\{LC, CV, CS, PS\}$ consecutions for a transition τ, then η satisfies consecution for τ.*

Example 12. 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 10 according to local consecution yields a normal form

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

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

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

Continuous Consecution

The continuous consecution condition states that if the invariant holds at some state $\langle \ell, \vec{x_1}\rangle$ then it must hold at any state $\langle \ell, \vec{x_2}\rangle$ where $\vec{x_2}$ can be reached from $\vec{x_1}$ according to the differential rule $\mathcal{D}(\ell)$, while satisfying $\mathbf{I}(\ell)$. That is,

$$\mathbf{I}(\ell) \;\wedge\; \eta(\ell) = 0 \models \dot{\eta(\ell)} = 0\;\;.$$

As in the discrete case, encoding this exactly is not practical, and therefore we impose stronger conditions, summarized in Figure 4.

Constant Value (CV) The constant value 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:

$$\dot{\eta}(\ell) \;=\; 0\;\;.$$

Clearly, this condition guarantees that the assertion is preserved. Noting that the system variables are functions of time only, 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 $\dot{\eta}(\ell)$:

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

This is also known as the *Lie-Derivative* of $\eta(\ell)$ w.r.t. the vector field $\vec{\dot{x}} = \langle p_1, \ldots, p_n\rangle$ associated with the location ℓ.

Name	Condition	Encoding
CV	$\mathbf{I}(\ell) \models \dot\eta(\ell) = 0$	$\text{NF}_{\mathbf{I}(\ell)}(\dot\eta(\ell)) \equiv 0$
CS	$(\exists\ \lambda)\ \mathbf{I}(\ell) \models \dot\eta(\ell) - \lambda\eta(\ell) = 0$	$(\exists\ \lambda)\ \text{NF}_{\mathbf{I}(\ell)}(\dot\eta(\ell) - \lambda\eta(\ell)) \equiv 0$
PS	$(\exists\ g)\ \mathbf{I}(\ell) \models \dot\eta(\ell) - g\eta(\ell) = 0$	$(\exists\ g)\ \text{NF}_{\mathbf{I}(\ell)}(\dot\eta(\ell) - g\eta(\ell)) \equiv 0$

Figure 4. Continuous consecution conditions

Constant Scale (CS) The constant scale condition 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:

$$(\exists\ \lambda)\ \dot\eta(\ell) - \lambda\eta(\ell) = 0\ .$$

Polynomial Scale (PS) The polynomial scale condition relaxes the condition further by requiring that there exists a polynomial factor such that the difference between the derivative and this factor times the invariant be zero:

$$(\exists\ g)\ \dot\eta(\ell) - g\eta(\ell) = 0\ .$$

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 13. Returning to the bouncing-ball system from Example 1, and the template from Example 10, the derivative of the template is

$$\dot\eta(l) = \left(\begin{array}{c} (2a_1y + a_4v_y + a_6\delta + a_7)\ \dot y + \\ (2a_2v_y + a_4y + a_5\delta + a_8)\ \dot{v_y} + \\ (2a_3\delta + a_5v_y + a_6y + a_9)\ \dot\delta \end{array}\right)\ ,$$

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

$$\dot\eta(l)\ =\ \left(\begin{array}{c} a_4v_y^2\ +\ 2a_1yv_y\ +\ a_6\delta v_y\ +\ (-20a_2 + a_5 + a_7)v_y + \\ (2a_3 - 10a_5)\delta\ +\ (-10a_4 + a_6)y\ +\ (a_9 - 10a_8) \end{array}\right)\ .$$

For this system the location condition $\mathbf{I}(l)$ does not have any algebraic conjuncts and therefore the CV encoding for continuous consecution for location l is

$$\text{NF}_{\mathbf{I}(l)}(\dot\eta(l)) = \dot\eta(l)$$

Similarly, the CS encoding is

$$(\exists\ \lambda)\ \text{NF}_{\mathbf{I}(l)}(\dot\eta(l) - \lambda\eta(l)) = (\exists\ \lambda)\ (\dot\eta(l) - \lambda\eta(l))\ ,$$

which equals

$$(\exists\ \lambda)\left(\begin{array}{c} \lambda a_1 y^2\ +\ (a_4-\lambda a_2)v_y^2\ -\ \lambda a_3\delta^2 + \\ (2a_1-\lambda a_4)yv_y\ +\ (a_6-\lambda a_5)v_y\delta\ -\ \lambda a_6 y\delta + \\ (-10a_4+a_6-\lambda a_7)v_y\ +\ (-20a_2+a_5+a_7-\lambda a_8)v_y + \\ (2a_3-10a_5-\lambda a_9)\delta\ +\ (a_9-10a_8-\lambda a_{10}) \end{array}\right)\ .$$

We shall establish the soundness of PS consecution for continuous flows. The soundness of CS and CV consecutions follows as a corollary.

Lemma 2 (Soundness). *Let* $\bigwedge_{i=1}^{n}\ \dot{x}_i = p_i(x_1,\ldots,x_n)$ *be a differential rule in some location* l *. Let* p *be any polynomial in* $\mathcal{R}[x_1,\ldots,x_n]$*, such that the derivative of* p *according to the flow in* l*, satisfies* $\dot{p} = gp$ *for some polynomial* g*. Then for any continuous flow* $\vec{u}(t)$*, such that* $t \in [0,\delta]$*, if* $p(\vec{u}(0)) = 0$ *then* $p(\vec{u}(\delta))$*=0.*

Proof. Let f be a real-valued function of time, defined as $f(t) = p(\vec{u}(t))$ such that f is smooth (i.e, continuous and differentiable to any order) for $t \in [0,\delta]$. We first show that at $t = 0$, all the orders of derivatives of f vanish. If u_i is the i^{th} component of $\vec{u}$, then by definition, $\dot{u}_i = p_i(\vec{u})$.

$$\dot{f}(t) = \dot{p}(\vec{u}(t)) = g(\vec{u}(t)) \cdot p(\vec{u}(t)) = f_1 \cdot f\ \ .$$

We prove by induction that $f^{(m)} = \frac{d^m f}{dt^m} = f \cdot f_m$ for $m > 0$, where f_m is a smooth function of time. The base case for $n = 0, 1$ have been established above. Note that

$$f^{(m+1)} = \frac{df^{(m)}}{dt} = \frac{df_m f}{dt} = \dot{f}_m f + f_m \dot{f} = (\dot{f}_m + f_m f_1) f = f_{m+1} f$$

Since $p(\vec{u}(0)) = f(0) = 0$, we have that $f^{(n)}(0) = 0$ for all $n \geq 0$. Using the Taylor series expansion for f in the interval $t \in [0,\delta]$, we obtain $f(t) = 0$. In particular, $f(\delta) = p(\vec{u}(\delta)) = 0$. □

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.

Example 14. Consider again the system presented in Example 1 and the template of Example 10. The encodings, derived in Section 3.2, of the initial condition, discrete consecution (LC) of τ and continuous consecution (CV) at location l are

$$\begin{array}{lcl}
\mathrm{NF}(\eta(l)) & = & 256a_2 + 16a_8 + a_{10} \\[2ex]
\mathrm{NF}_\rho(\eta'(l)) & = & \frac{a_2}{4}v_y^2 - \frac{a_8}{2}v_y + a_{10} \\[2ex]
\mathrm{NF}_{\mathbf{I}(l)}(\dot\eta(l)) & = & \left(\begin{array}{c} a_4v_y^2 + 2a_1yv_y + a_6\delta v_y + (-20a_2 + a_5 + a_7)v_y + \\ (2a_3 - 10a_5)\delta + (-10a_4 + a_6)y + (a_9 - 10a_8) \end{array}\right)
\end{array}$$

yielding the set of (linear) constraints

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

and

$$a_2 \;=\; a_8 \;=\; a_{10} \;= a_4 \;=\; a_1 \;=\; a_6 \;=\; 0$$

Example 15. If we use the Constant Scale (CS) condition for both discrete and continuous consecution, then, as presented in Examples 12 and 13, we obtain the following set of (parametric) constraints:

$$256a_2 \;+\; 16a_8 \;+\; a_{10} \;=\; 0$$

$$\wedge$$

$$\exists\lambda_1 \left(\begin{array}{c} a_2(4\lambda_1 - 1) = 0 \;\wedge\; \lambda_1 a_5 = 0 \;\wedge\; a_9(2\lambda_1 + 1) = 0 \wedge \\ \lambda_1 a_3 = 0 \;\wedge\; \lambda_1 a_9 = 0 \;\wedge\; a_{10}(\lambda_1 - 1) = 0 \end{array}\right)$$

$$\wedge$$

$$\exists\lambda_2 \left(\begin{array}{c} \lambda_2 a_1 = 0 \;\wedge\; a_4 - \lambda_2 a_2 = 0 \;\wedge\; \lambda_2 a_3 = 0 \wedge \\ 2a_1 - \lambda_2 a_4 = 0 \;\wedge\; a_6 - \lambda_2 a_5 = 0 \;\wedge\; \lambda_2 a_6 = 0 \wedge \\ -10a_4 + a_6 - \lambda a_7 = 0 \;\wedge\; -20a_2 + a_5 + a_7 - \lambda_2 a_8 = 0 \wedge \\ 2a_3 - 10a_5 - \lambda_2 a_9 = 0 \;\wedge\; a_9 - 10a_8 - \lambda_2 a_{10} = 0 \end{array}\right)$$

A solution to these sets of constraints is presented in the next section.

Complexity of Constraint Generation
An assessment of the complexity of the constraint generation process requires the quantification of the time complexity of the Gröbner basis

and the normal form computations. We shall assume an upper bound T_{gb} on these operations. The template size is assumed to be polynomial in the dimensions, and exponential in the degree of the template. This can be remedied to some extent by the *template shrinking* strategies presented in the next section.

The number of Gröbner basis conversions is one per location, and per discrete transition. More often than not, the location invariants, the initial conditions and the discrete transitions are simple in structure, containing only non-factorizable polynomials. This speeds up the Gröbner basis computation in practice. Computing the initiation and discrete consecution constraints involves one normal form computation per condition. Each normal form computation is assumed linear in the template size. Furthermore, computing the Lie derivative involves n polynomial multiplications, and is at most quadratic in the template size (algorithms using fast fourier transforms (FFT) multiply two polynomials of length m is $O(m\ log(m))$ time). Therefore, modulo the cost of computing Gröbner bases and normal forms, the constraint generation process is polynomial in the program size (number of dimensions + number of locations + number of transitions), and exponential in the desired degree if generic templates are used. The number of Gröbner basis computations is linear in the system description. The number of normal form conversions is also linear in the system description, with each normal form conversion operating on a template polynomial.

4. Solving Constraints

The encodings presented in the previous section generate several types of constraints. Initiation always produces linear equalities. The type of constraints generated by the consecution condition vary from linear equalities for the most restricted, Local Consecution condition, to general non-linear equalities for the Polynomial Scale condition. Figure 5 shows the different types obtained for the various conditions and encodings.

Solution techniques exist for all of these types of constraints, but they differ greatly in their complexity. Solution techniques for linear equalities are well understood and tend to be computationally inexpensive. For example, Gaussian elimination can be done efficiently in polynomial time (assuming constant time arithmetic operations). 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 [5].

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

Figure 5. Constraints obtained from different conditions for inductive assertions

Two types of non-linear constraints can be distinguished. For the case of Constant Scale consecution, non-linearities are introduced in the constraints by the scale parameter λ. For this case, the resulting constraints are generalized eigenproblems of the form $A\vec{x} = \lambda B\vec{x}$. Both numerical and symbolic solutions exist for these problems. Numerical solution techniques are faster and more easily implemented, but may be sensitive to round-off errors. Using inaccurate values for λ results in finding only the trivial solution. Symbolic solution techniques, on the other hand, can only be used for special cases, because they depend on polynomial root solving. From our experience, numerical techniques are adequate, provided the values obtained for λ are rational or algebraic of low degree.

The general non-linear constraints generated by the Polynomial Scale consecution condition are much harder to solve. For small to medium size systems, these constraints can be handled using a combination of simplification and quantifier elimination. For higher-degree polynomials, generic quantifier elimination techniques over the reals are required, which work well for small sized problems [6], but do not scale.

4.1. Linear Equalities

Linear equalities arise from the encodings of the initial condition, LC and CV consecution for the discrete case, and the CV consecution for the continuous case.

The theory of linear equalities over the reals is relatively simple with fast solution techniques. Representing the system of constraints in matrix form

$$A\vec{x} + \vec{b} = 0$$

with A an $m \times n$ matrix and $\vec{b}$ an $m \times 1$ column vector, we are interested in all solutions $\vec{x}$. In general, the system is under-determined, and hence, the set of solutions forms a linear subspace of dimension greater than zero. Assume that the solution is given as $\vec{x}_d = C\vec{x}_i$, where $\vec{x}_i$ are the independent variables and $\vec{x}_d$ are dependent. The generators are computed by expanding the orthonormal basis for $\vec{x}_i$ with $\vec{x}_d = C\vec{x}_i$. Each generator yields an independent invariant when substituted in place of the unknown coefficients of the template.

The system of constraints may not always lead to a (useful) invariant. If the system is infeasible, which is the case iff the row reduced echelon form contains the equation $1 = 0$, then there are no solutions, signifying that no invariant of the form of the template could be found by our technique. In case the system is homogeneous, that is $\vec{b} = 0$, it always has the trivial solution $\vec{x} = 0$, which produces the trivial invariant *true*. In this case non-trivial invariants are obtained if the constraint system exhibits non-trivial solutions.

Example 16. Recall the constraints from Example 14. After simplification, this system can be written as

$$\begin{array}{rcrcl} a_5 & + & a_7 & = & 0 \\ a_3 & - & 5a_5 & = & 0 \end{array}$$

or in matrix notation,

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & -5 & 0 \end{pmatrix} \begin{pmatrix} a_3 \\ a_5 \\ a_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

which can be converted to the equivalent row reduced echelon form

$$\begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_3 \\ a_5 \\ a_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

with dependent variables a_3 and a_5 and independent variable a_7. Expanding the basis vector for a_7 to include a_3 and a_5 yields the single generator

$$\begin{pmatrix} -5 \\ -1 \\ 1 \end{pmatrix}$$

for the solution subspace. Assigning these values to the template of Example 10 produces the non-trivial, inductive invariant

$$-5\delta^2 - v_y\delta + y = 0 \ .$$

Template Shrinking

Linear equalities are useful even when some constraints are non-linear. In the example above, we considered the case where all constraints were linear equalities, and therefore linear constraint solving techniques were able to produce the invariant. More commonly, only some of the constraints are linear equalities and non-linear constraint solving techniques must be used to obtain the invariant. In this case the linear equalities may be used to reduce the size of the template, thereby simplifying the constraint generation and the constraint solving processes.

Let f be a template:

$$f : c_1 p_1 + c_2 p_2 + \ldots + c_n p_n \quad \text{or} \quad \vec{c} \cdot \vec{p}$$

where $\vec{c}$ are the coefficients (or template variables) and $\vec{p}$ are the monomials. Let $A\vec{c} = 0$ be a system of constraints generated on the template variables, where the column rank of A is m. Then the number of independent variables is $n - m$. If A is full column rank ($m = n$) all template variables are determined, and hence it just remains to check whether the resulting invariant satisfies the remaining (non-linear) constraints. If the system of constraints is empty ($m = 0$), all template variables are independent and no simplification can be made to the template. For any intermediate value m of the rank, rewriting $A\vec{c} = 0$ in row echelon form produces the m dependent variables as linear combinations of the $m - n$ independent variables. Hence the dependent variables can be removed from the template, resulting in a template with fewer unknowns.

Example 17. Recall the template from example 10:

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

and the constraints for initiation and CV (continuous) consecution from example 14:

$$\begin{aligned} a_{\{1,4,6\}} &= 0 \\ a_{10} + 256 a_2 + 16 a_8 &= 0 \\ a_5 + a_7 - 20 a_2 &= 0 \\ a_3 - 5 a_5 &= 0 \\ a_9 - 10 a_8 &= 0 \end{aligned}$$

This system of constraints can be rewritten in row echelon form (leaving out a_1, a_4, and a_6) as follows:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -5 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -20 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -10 \\ 0 & 0 & 0 & 1 & 0 & 256 & 16 \end{pmatrix} \begin{pmatrix} a_3 \\ a_7 \\ a_9 \\ a_{10} \\ a_5 \\ a_2 \\ a_8 \end{pmatrix} = 0$$

producing as dependent variables a_3, a_7, a_9, and a_{10}, and the relationships

$$\begin{array}{lcl} a_{\{1,4,6\}} & = & 0 \\ a_3 & = & 5a_5 \\ a_7 & = & 20a_2 - a_5 \\ a_9 & = & 10a_8 \\ a_{10} & = & -256a_2 - 16a_8 \end{array}$$

Substituting the linear combinations for the dependent variables yields the template:

$$\eta_s(l) = a_2(v_y^2 + 20y - 256) + a_5(v_y\delta + 5\delta^2 - y) + a_8(v_y + 10\delta - 16)$$

with only the dependent variables a_2, a_5, and a_8 as unknowns. This makes the encoding and the solving of the remaining consecution condition easier.

In practice, if the CV condition is used for continuous consecution at a particular location, the generated linear equalities can automatically be used to reduce the template for that location. This makes the generation of the discrete consecution constraints and their solution faster, providing a dramatic reduction in the complexity of the constraints to be solved.

4.2. Parametric-Linear Equalities

Parametric linear equalities arise from the encoding of the constant scale (CS) consecution constraints. They encompass both the CV and LC conditions for the discrete case, and the CV condition for the continuous case.

Definition 15 (Parametric Linear Constraint). A parametric linear constraint is an equality of the form

$$(\exists \lambda_1, \ldots, \lambda_k)\ (e_0 + \sum_{i=1}^{k} \lambda_i e_i = 0)$$

where $e_0, \ldots, e_k$ are homogeneous linear expressions over the template variables.

Assuming that a CS encoding is used for all consecution conditions, the constraint system generated is a conjunction of linear equalities and parametric linear equalities with one parameter λ_i for each consecution condition. That is, the system of constraints is of the following form:

$$(\exists \lambda_1, \ldots, \lambda_k)\ (A\vec{x}\ = 0\ \wedge\ \bigwedge_{i=1}^{m}((A_i + \lambda_i B_i)\vec{x} = 0))$$

where $\vec{x}$ are the template variables and A, A_i, B_i are constant matrices. We first show that checking whether such a system has a non-trivial solution is NP-hard in general, and then provide a strategy that solves these systems efficiently in most cases we have encountered so far.

Lemma 3. *Consider a constraint system of the form given above. Checking whether there exist values for $\lambda_1, \ldots, \lambda_k$ such that the instantiated system has a non-trivial solution for $\vec{x}$ is NP-hard.*

Proof. The reduction is from the SUBSET-SUM problem, which is shown to be NP -complete (see [13], page 223 for further details).

SUBSET SUM: Given an n element set S, with elements having real-valued weights $w_1, \ldots, w_n$, is there a subset whose elements' weights add up to $W \neq 0$?

We introduce $n+1$ variables $x_1, \ldots, x_n, \epsilon$. The system of constraints is shown below:

$$\begin{array}{lcl} \sum_{i=1}^{n} w_i x_i - W\epsilon = 0 & \longrightarrow & \text{weights must add up} \\ \bigwedge_{i=1}^{n} \left\{ \begin{array}{l} \lambda_i x_i = 0 \\ (\lambda_i - 1)(x_i - \epsilon) = 0 \end{array} \right. & \longrightarrow & x_i = 0 \ \mathit{or}\ x_i = \epsilon \ \mathit{or}\ x_i = \epsilon = 0 \end{array}$$

The equations have been set up so that each x_i is either 0 or ϵ. A solution is non-trivial iff $\epsilon \neq 0$. Let $\vec{x}, \epsilon$ be a non-trivial solution. Therefore $\frac{1}{\epsilon}\vec{w}^{\mathrm{T}}\vec{x} = W$. Given that each x_i is either 0 or ϵ, $\frac{x_i}{\epsilon}$ is either 0 or 1. Therefore, a non-trivial solution to the feasibility problem yields one for the subset-sum problem. Similarly, a (necessarily non-trivial) solution for the subset-sum problem along with $\epsilon = 1$ yields a non-trivial solution to the feasibility problem, completing the reduction. □

We do not attempt a proof of NP-completeness since membership in NP requires proving that the solution size is polynomial in the problem size. This seems non-trivial, and outside the scope of our discussion. In practice, such a system of constraints can often be solved much more efficiently by means of factorization followed by case splitting. In a previous paper we showed that this strategy handles all parametric linear constraints generated in the invariant generation of Petri Nets by exploiting the structure of the transitions in Petri Nets [30]. In the present case we use the strategy as a heuristic that has proven successful for most constraints we have encountered so far.

A parametric constraint is factorisable if it can be written in the form

$$e(\lambda - \alpha) \ = \ 0$$

for some $\alpha \in \mathcal{R}$. A system of constraints $\exists\vec{\lambda}\ \varphi$ that contains such a factorisable constraint can be split into two cases as follows:

$$\exists\lambda\ ((\varphi\ \wedge\ \lambda = \alpha)\ \vee\ (\varphi\ \wedge\ \lambda \neq \alpha))$$

which can then be solved separately. In the first disjunct all occurrences of λ can be substituted with α, effectively eliminating λ. The presence of $\lambda \neq \alpha$ in the second disjunct entails the equality $e = 0$.

Our strategy is as follows. To solve a system of linear parametric constraints

$$(\exists\lambda_1, \ldots, \lambda_k)\ (A\vec{x}\ = 0\ \wedge\ \bigwedge_{i=1}^{m}((A_i + \lambda_i B_i)\vec{x} = 0))$$

repeatedly apply the following two steps:

1. use $A\vec{x}\ = 0$ to rewrite the dependent variables in $\vec{x}$ in terms of the independent variables in $\bigwedge_{i=1}^{m}((A_i + \lambda_i B_i)\vec{x} = 0)$ (as was illustrated in Example 17 for template reduction);

2. select a factorisable equality $a_i + \lambda_i b_i = 0$ and split the system in two cases as shown above and solve each case separately;

until neither step is applicable. The unsolved cases may be discarded (without loss of soundness), or we may revert to the simpler conditions by instantiating the remaining parameters to 0 (for LC encoding in the discrete case or CV encoding in the continuous case), or 1 (for CV encoding in the discrete case). Note that each disjunct, in particular each instantiation of the parameters, potentially leads to an invariant. At any stage we may discard disjuncts, or instantiate parameters. By doing so we may fail to find some invariants but we do not compromise soundness. In practice further expansion of a disjunct can often be avoided if the linear equalities in it entail an another disjunct that has already been resolved. Testing this subsumption can be done efficiently, and saves a lot of effort in practice.

Example 18. Reconsider the constraints derived in Example 15 for the Initiation condition and the CS consecution conditions. Let this system of constraints be denoted by

$$\exists\lambda_1\ \lambda_2\ \varphi\ .$$

We will eliminate the quantified variables by the factorization strategy outlined above. Noticing that λ_2 is a factor in several equations, we first split the system on $\lambda_2 = 0$, yielding

$$\exists\lambda_1\ \lambda_2\ (\varphi\ \wedge\ \lambda_2 = 0)\ \vee\ (\varphi\ \wedge\ \lambda_2 \neq 0)\ .$$

Simplification of the second disjunct produces the trivial solution

$$a_1 = a_2 = \ldots = a_{10} = 0$$

which corresponds to the trivial invariant *true*.

Substituting 0 for λ_2 in φ and simplifying the result leads to

$$\exists \lambda_1 \left(\begin{array}{c} 256a_2 + 16a_8 + a_{10} = 0 \wedge \\ a_2(4\lambda_1 - 1) = 0 \ \wedge \ \lambda_1 a_5 = 0 \ \wedge \ a_9(2\lambda_1 + 1) = 0 \wedge \\ \lambda_1 a_3 = 0 \ \wedge \ \lambda_1 a_9 = 0 \ \wedge \ a_{10}(\lambda_1 - 1) = 0 \wedge \\ a_4 = 0 \ \wedge \ a_1 = 0 \ \wedge \ a_6 = 0 \wedge \\ -20a_2 + a_5 + a_7 = 0 \ \wedge \ 2a_3 - 10a_5 = 0 \ \wedge \ a_9 - 10a_8 = 0 \end{array} \right)$$

We denote the system above as $\exists \lambda_1 \varphi_1$. Splitting this system with respect to the eigenvalues for λ_1 results in

$$\exists \lambda_1 \left(\begin{array}{c} (\varphi_1 \ \wedge \ \lambda_1 = 0) \vee \\ (\varphi_1 \ \wedge \ \lambda_1 = 1) \vee \\ (\varphi_1 \ \wedge \ \lambda_1 = \frac{1}{4}) \vee \\ (\varphi_1 \ \wedge \ \lambda_1 = -\frac{1}{2}) \vee \\ (\varphi_1 \ \wedge \ \lambda_1 \neq \{0, \frac{1}{4}, -\frac{1}{2}, 1\}) \end{array} \right)$$

The first disjunct, in which $\lambda_1 = 0$, results in the same linear system solved in Example 16, and thus produces the same invariant

$$-5\delta^2 - v_y \delta + y \ = \ 0 \ .$$

All other disjuncts produce the trivial solution

$$a_1 = a_2 = \ldots = a_{10} = 0$$

and thus do not lead to other invariants. More accurate conditions for consecution did not result in stronger invariants for this example. Only the choice $\lambda_1 = \lambda_2 = 0$, which coincides with the CV encoding for the continuous consecution and the LC encoding for the discrete consecution, produced a non-trivial invariant.

5. Applications

We demonstrate the results of our technique on some application examples. All the examples were run on our prototype implementation in Mathematica linked with a C++ implementation of the constraint-solving strategies discussed in the previous section, and elsewhere [30]. The example descriptions and the code will be available from our webpages.

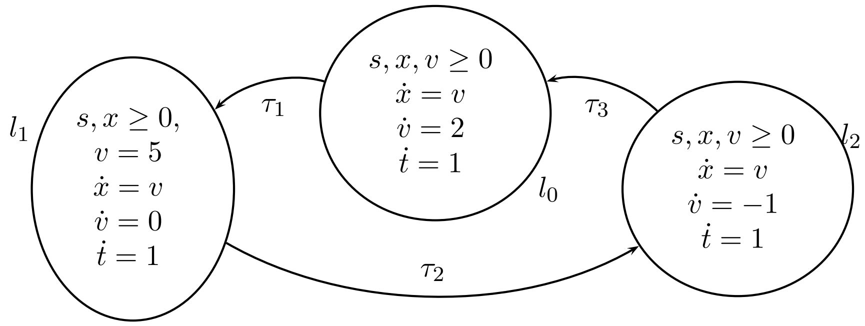


Figure 6. Hybrid Automaton for the Train System

TRAIN SYSTEM:

Figure 6 shows a hybrid automaton modeling a train accelerating (location l_0), traversing 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 master clock. The system has one discrete variable 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

$$\begin{array}{ll}
\rho_{\tau_1} & : \ v = 5 \ \wedge \ \mathbf{id}(s, x, v, t) \\
\rho_{\tau_2} & : \ \mathbf{id}(s, x, v, t) \\
\rho_{\tau_3} & : \ v = 0 \ \wedge \ s' = s + 1 \ \wedge \ t' = t + 2 \ \wedge \ \mathbf{id}(x, v)
\end{array}$$

wherein $\mathbf{id}(X)$ denotes $\bigwedge_{x \in X}(x' = x)$. Application of our technique resulted in the following assertion map:

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

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 traveled 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 modeling (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.

Particle in a Magnetic Field

Figure 7 shows a charged particle on a 2D-plane with a reflecting barrier at $x = 0$, and a magnetic field at $x \geq d \geq 0$. The eight system variables are the particle's position and velocity, x, y, v_x, v_y, a *bounce counter* b that is incremented every time the particle collides against the barrier at $x = 0$, along with the parameters a, d, and time t. The dynamics of the particle at $0 \leq x \leq d$ are those of constant velocity motion given by $\dot{\vec{x}} = \vec{v}$ and $\dot{\vec{v}} = 0$. In the magnetic field region, a force perpendicular to the particle's velocity vector causes it to move in a circular orbit, with dynamics given by $\dot{\vec{x}} = \vec{v}$, $\dot{v_x} = -av_y$ and $\dot{v_y} = av_x$. The particle's behavior is modeled by the three mode hybrid automaton shown in Figure 7. Initially, $v_x = 2$, $v_y = -2$, and $x = y = t = b = 0$. The parameter d was set to 2, and a left unspecified. This makes the simulation and the propagation-based exploration of the system hard. Transition τ_1 models the transition from the right moving particle into the magnetic field. It is guarded by $x = 2$, and leaves the variables unchanged. The transition τ_2 models the transition back from *magnetic* into the *left* region. It is guarded by $x = 2$, and leaves the variables unchanged. The bounce on the barrier at $x = 0$ is modeled by τ_3, which negates v_x, and increments b by 1. The mode invariants $v_x \geq 0$ for left, and $v_x \leq 0$ for right were ignored. This however leads to unwanted behaviors like the particle in mode *right* making a transition into *magnetic* only to make another transition into *left* with no time elapse. The following

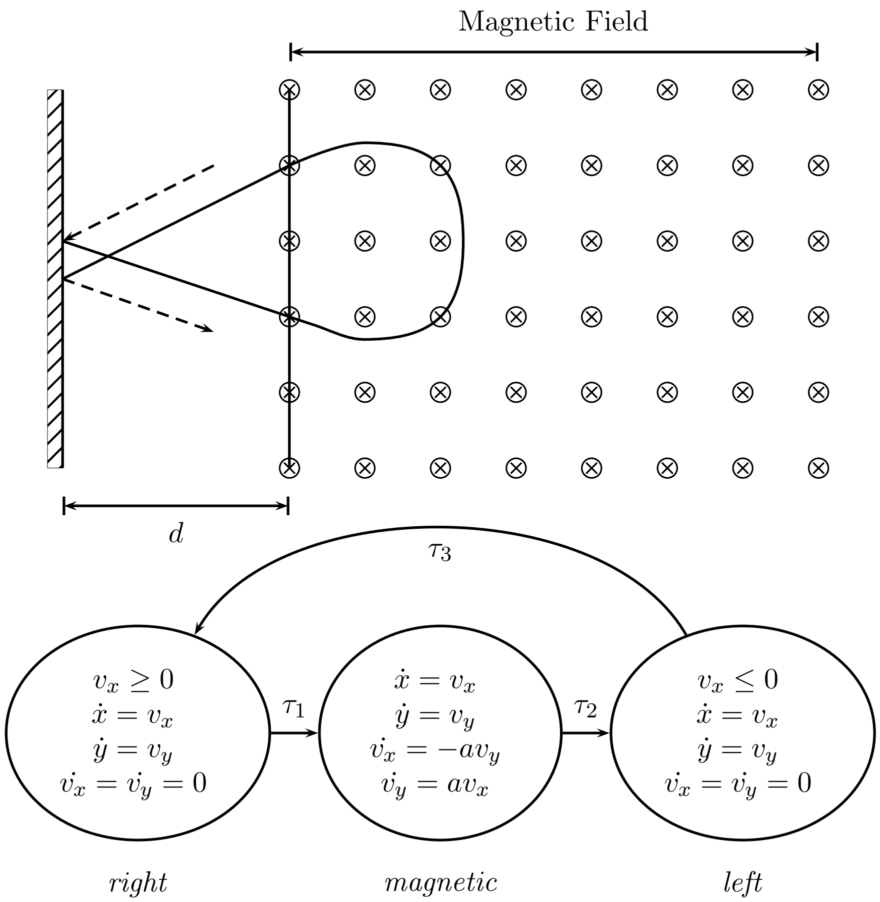


Figure 7. Particle in a magnetic field and its non-linear hybrid model. The dynamics of a, t, b are not shown.

invariants were obtained:

Location	Invariant
left	$v_y + 2 = 0 \ \wedge \ v_x^2 - 4 = 0$
magnetic	$v_y = a(x-2) - 2 \ \wedge \ v_x^2 + v_y^2 = 8$
right	$v_y + 2 = 0 \ \wedge \ v_x^2 - 4 = 0$

However, at location *right*, the invariant $v_x \geq 0$ and $v_x^2 - 4 = 0$ lets us conclude $v_x = 2$. This is easily mechanized by computing the roots of the univariate polynomials, and discarding extraneous roots disallowed by the inequalities in the invariants. Similarly, at location *left*, we conclude $v_x + 2 = 0$. With the newly computed invariants added to the modes, another run of the procedure yields the following invariants

Location	Invariant
left	$v_y + 2 = 0 \ \wedge \ v_x - 2 = 0 \ \wedge \ a(x+y) = 4b - 4ab$
magnetic	$\begin{bmatrix} v_y + 2 = a(x-2) \\ v_x^2 + v_y^2 = 8 \\ v_x - 2 = -a(y+2) + 4b(1-a) \end{bmatrix}$
right	$v_y + 2 = 0 \ \wedge \ v_x + 2 = 0 \ \wedge \ a(x-y) = -4(b+1)(1-a)$

These invariants let us deduce many properties about the system.

For instance, when $a = 0$, the invariant $b = 0, \ v_x = 2, v_y = -2$ can be deduced at *right*. The invariant at *magnetic* is the same as that in *right*. The invariant at *left* is $b = -1, \ v_x = v_y = -2$. Also for $a = 0$, these invariants suffice to show the unreachability of *left*. For the case $a \neq 0$, the invariants let us conclude that the collision point at the barrier for the bounce number b is $y = 4b\frac{1-a}{a}$, where $a \neq 0$. For $a = 1$, the particle returns to the origin for every bounce. For $a > 1$, the particle's path crosses itself as shown in Figure 7. The radius of the arc described by the particle tends to infinity as a approaches 0. If the sign of a is reversed, the particle's circular motion changes from clockwise to anticlockwise.

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. We are investigating strategies for extending the results in this paper to generate inequality invariants of hybrid systems.

Acknowledgement
We would like to thank the anonymous referees for their careful reading of the paper and the many suggestions for improvement.

References

1. Asarin, E., Dang, T., and Maler, O. The d/dt tool for verification of hybrid systems. In *Proc.* 14^{th} *Intl. Conference on Computer Aided Verification* (2002), vol. 2404 of *LNCS*, Springer Verlag, pp. 365–370.
2. Baader, F., and Nipkow, T. *Term Rewriting and* All That. Cambridge University Press, 1998.
3. Bengtsson, J., Larsen, K. G., Larsson, F., Pettersson, P., and Yi, W. Uppaal — a Tool Suite for Automatic Verification of Real–Time Systems. In *Proc. of Workshop on Verification and Control of Hybrid Systems III* (Oct. 1995), no. 1066 in Lecture Notes in Computer Science, Springer–Verlag, pp. 232–243.
4. 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.
5. 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.
6. Collins, G. E., and Hong, H. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation 12*, 3 (sep 1991), 299–328.
7. Colón, M. Approximating the algebraic relational semantics of imperative programs. In 11^{th} *Static Analysis Symposium (SAS'2004)* (2004), vol. 3148 of *LNCS*, Springer-Verlag.
8. 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.
9. 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.
10. 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.
11. Cox, D., little, J., and O'Shea, D. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Springer, 1991.

12. FORSMAN, K. Construction of Lyapunov functions using Gröbner bases. In *Proc.* 30^{th} *IEEE CDC* (1991).
13. GAREY, M., AND JOHNSON, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman & Co., New York, 1999.
14. 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.
15. HENZINGER, T., AND HO, P.-H. Algorithmic analysis of nonlinear hybrid systems. In *Computer-Aided Verification*, vol. 939 of *LNCS*. Springer-Verlag, 1995, pp. 225–238.
16. HENZINGER, T. A. The theory of hybrid automata. In *Logic In Computer Science (LICS 1996)* (1996), IEEE Computer Society Press, pp. 278–292.
17. HENZINGER, T. A., AND HO, P. HYTECH: The Cornell hybrid technology tool. In *Hybrid Systems II* (1995), vol. 999 of *LNCS*, Springer-Verlag, pp. 265–293.
18. KARR, M. Affine relationships among variables of a program. *Acta Inf. 6* (1976), 133–151.
19. LAFFERRIERE, G., PAPPAS, G., AND YOVINE, S. Symbolic reachability computation for families of linear vector fields. *J. Symbolic Computation 32* (2001), 231–253.
20. MANNA, Z., AND PNUELI, A. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.
21. MISHRA, B., AND YAP, C. Notes on Gröbner bases. *Information Sciences 48* (1989), 219–252.
22. MÜLLER-OLM, M., AND SEIDL, H. Polynomial constants are decidable. In *Static Analysis Symposium (SAS 2002)* (2002), vol. 2477 of *LNCS*, Springer-Verlag, pp. 4–19.
23. MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE 77*, 4 (Apr. 1989), 541–580.
24. PARILLO, P. A. Semidefinite programming relaxation for semialgebraic problems. *Mathematical Programming Ser. B 96*, 2 (2003), 293–320.
25. PRAJNA, S., AND JADBABAIE, A. Safety verification using barrier certificates. In *Hybrid Systems: Computation and Control* (2004), vol. 2993 of *LNCS*, Springer-Verlag, pp. 477–492.
26. RODRIGUEZ-CARBONELL, E., AND KAPUR, D. An abstract interpretation approach for automatic generation of polynomial invariants. In 11^{th} *Static Analysis Symposium (SAS'2004)* (2004), vol. 3148 of *LNCS*, Springer-Verlag.
27. RODRIGUEZ-CARBONELL, E., AND KAPUR, D. Automatic generation of polynomial loop invariants: Algebraic foundations. In *Proc. Intl. Symp on Symbolic and Algebraic Computation (ISSAC-2004)* (Spain, 2004).
28. SANKARANARAYANAN, S., SIPMA, H., AND MANNA, Z. Non-linear loop invariant generation using Gröbner bases. In *ACM Principles of Programming Languages (POPL)* (2004), ACM Press, pp. 318–330.
29. SANKARANARAYANAN, S., SIPMA, H. B., AND MANNA, Z. Petri net analysis using invariant generation. In *Verification: Theory and Practice* (2003), vol. 2772 of *LNCS*, Springer-Verlag, pp. 682–701.
30. SANKARANARAYANAN, S., SIPMA, H. B., AND MANNA, Z. Constraint-based linear relations analysis. In 11^{th} *Static Analysis Symposium (SAS'2004)* (2004), vol. 3148 of *LNCS*, Springer-Verlag, pp. 53–68.
31. SILVA, B., RICHESON, K., KROGH, B., AND CHUTINAN, A. Modeling and verifying hybrid dynamic systems using CheckMate. In *Proc. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems* (2000), pp. 323–328.

32. Tiwari, A. Approximate reachability for linear systems. In *Hybrid Systems: Computation and Control HSCC* (2003), vol. 2623 of *LNCS*, Springer-Verlag, pp. 514–525.
33. Tiwari, A., and Khanna, G. Non-linear systems: Approximating reach sets. In *Hybrid Systems: Computation and Control* (2004), vol. 2993 of *LNCS*, Springer-Verlag, pp. 477–492.
34. Tiwari, A., Ruess, H., Saïdi, H., and Shankar, N. A technique for invariant generation. In *TACAS 2001* (2001), vol. 2031 of *LNCS*, Springer-Verlag, pp. 113–127.
35. Windsteiger, W., and Buchberger, B. Gröbner: A library for computing Gröbner bases based on saclib. Tech. rep., RISC-Linz, 1993.
36. Yovine, S. Kronos: A verification tool for real-time systems. *Springer International Journal of Software Tools for Technology Transfer 1*, 1/2 (October 1997).

Appendix

In this appendix, we shall prove a few useful theorems mentioned in other results along with the confluence of the Gröbner basis reduction on templates. The key idea is to reduce any failure of confluence on templates to a failure of confluence on the instantiation of the template under an appropriate environment.

Claim 1. *Let f, f' be templates and α be any environment.*

1. $\alpha(f+g) = \alpha(f) + \alpha(g)$,
2. $c \cdot \alpha(f) = \alpha(c \cdot f)$, *for any* $c \in \mathcal{R}$.

Theorem 5 (Consistency). *Let $f \xrightarrow{p} f'$ for templates f, f', over template variables in A. Then, for an arbitrary A-environment α, $\alpha(f) \xrightarrow{p} \alpha(f')$ or $\alpha(f) = \alpha(f')$. Conversely, if for some α, $\alpha(f) \xrightarrow{p} h$ then there is a f' such that $h = \alpha(f')$ and $f \xrightarrow{p} f'$.*

Proof. We have that $f' = f - \frac{c \cdot t}{\mathrm{LT}(p)} p$ for some term $c \cdot t$ in f wherein $c(a_0, \ldots, a_m)$ is a linear expression. If $\alpha(c) = 0$ then $\alpha(f) = \alpha(f')$, or else, if $\alpha(c) \neq 0$ then we have that $\alpha(f') = \alpha(f) - \frac{\alpha(c \cdot t)}{\mathrm{LT}(p)} p$. In this case, $\alpha(f) \xrightarrow{p} \alpha(f')$.

On the other hand, let $\alpha(f) \xrightarrow{p} h$, for some template f and polynomial p. Let $\alpha(c) \cdot t$ be the term in $\alpha(f)$ that $\textsc{lt}(p)$ divides. Hence, the result of the reduction is,

$$h = \alpha(f) - \frac{\alpha(c)t}{\mathrm{LT}(p)} p = \alpha(f - \frac{ct}{\mathrm{LT}(p)} p)$$

Therefore, setting $f' = f - \frac{ct}{\mathrm{LT}(p)} p$, we have $\alpha(f') = h$ and $f \xrightarrow{p} f'$. □

Theorem 6 (Template Identity). *Two templates f_1, f_2 over template variables A are not identical iff there is an environment α such that $\alpha(f_1) \not\equiv \alpha(f_2)$.*

Proof. Since $f_1 \not\equiv f_2$, we have that $f_1 - f_2$ is a non-zero template. Let t be a term in $f_1 - f_2$, with a non-zero coefficient expression c. Let α be an environment s.t. $\alpha(c) \neq 0$. Hence $\alpha(f_1 - f_2) \not\equiv 0$. Thus we have that $\alpha(f_1) \not\equiv \alpha(f_2)$.

The other direction is immediate from the definition of identical templates. □

Theorem 7 (Normal Form Theorem). *A template f is a normal form under $\xrightarrow{G}$ iff for each environment α, $\alpha(f)$ is a normal form under $\xrightarrow{G}$.*

Proof. To prove the forward implication, assume that f is a normal form under $\xrightarrow{G}$. However assume that $\alpha(f) \xrightarrow{G} h$ for some α. By the reverse direction of the consistency theorem (Theorem 5), we have that there exists f' such that $f \xrightarrow{G} f'$ and $\alpha(f') = h$. This contradicts our assumption that f is in normal form.

To prove the reverse implication, let us assume that f is not in normal form. Then there is a reduction $f \xrightarrow{G} f'$. Let t be the term in f that is replaced by the reduction and c be its non-zero coefficient. By Theorem 5, we have that for each environment α, $\alpha(f) = \alpha(f')$ or $\alpha(f) \xrightarrow{G} \alpha(f')$. We find an environment α such that $\alpha(c) \neq 0$. For such an environment $\alpha(f) \xrightarrow{G} \alpha(f')$, since $\alpha(c) \neq 0$. Thus $\alpha(f)$ is not in normal form w.r.t. $\xrightarrow{G}$. □

Theorem 8 (Confluence of Templates). *Let G be a Gröbner basis and f be a template. Let $f \overset{G}{\twoheadrightarrow} f_1$ and $f \overset{G}{\twoheadrightarrow} f_2$, where f_1, f_2 are normal forms. We have that $f_1 \equiv f_2$ and hence $\xrightarrow{G}$ is confluent for templates.*

Proof. Assuming otherwise, i.e., $f_1 \not\equiv f_2$, we have by Theorem 6 that $\alpha(f_1) \not\equiv \alpha(f_2)$ for some α. Furthermore, Theorem 7 implies that $\alpha(f_1)$ and $\alpha(f_2)$ are in normal forms. By the forward direction of Theorem 5, we have that $\alpha(f) \overset{G}{\twoheadrightarrow} \alpha(f_i)$, $i = 1, 2$. Hence, by the confluence of the Gröbner basis reduction relation over real polynomials, we have that $\alpha(f_1) = \alpha(f_2)$, thus leading to a contradiction. □

Theorem 9 (Normal Forms for Templates). *Let f be a template over variables A. Let G be the Gröbner basis of $I = (G)$. Then, for any A-environment α,*

$$\alpha(\mathrm{NF}_G(f)) = \mathrm{NF}_G(\alpha(f))$$

Proof. The proof is by induction over the length of the minimal sequence of reductions from f to $f' = \textsc{nf}(f)$. For zero length derivations, $f = \textsc{nf}(f)$, we have that $\alpha(f)$ is a normal form. Hence, $\alpha(\textsc{nf}(f)) = \alpha(f) = \textsc{nf}(\alpha(f))$.

Assuming that the theorem holds for templates which have a length n or less derivation to their normal forms, let $f \xrightarrow{G} f_1 \overset{G}{\twoheadrightarrow}_n \textsc{nf}(f)$. Hence, by Theorem 5, we have that $\alpha(f) = \alpha(f_1)$ or $\alpha(f) \xrightarrow{G} \alpha(f_1)$. In either case $\textsc{nf}(\alpha(f_1)) = \textsc{nf}(\alpha(f))$. Applying the induction to f_1, we have that $\alpha(\textsc{nf}(f_1)) = \textsc{nf}(\alpha(f_1))$. Hence

$$\alpha(\textsc{nf}(f)) = \alpha(\textsc{nf}(f_1)) = \textsc{nf}(\alpha(f_1)) = \textsc{nf}(\alpha(f))$$

□

Theorem 10 (Template Membership). *Let f be a template and G be a Gröbner basis, such that $I = ((G))$. Let $f' = \textsc{nf}_G(f)$. For each environment α, $\alpha(f) \in I$ iff $\alpha(f')$ is* identically zero*.*

Proof. Given f, G such that $f' = \textsc{nf}_G(f)$, for any environment α, we have that $\alpha(f') = \alpha(\textsc{nf}(f)) = \textsc{nf}(\alpha(f))$. Hence, if $\alpha(f')$ is identically zero, then $\textsc{nf}(\alpha(f))$ is identically zero, and therefore, $\alpha(f) \in (G)$.

Let $\alpha(f) \in (G)$, hence $\textsc{nf}_G(\alpha(f))$ is identically zero. However, $\alpha(f') = \alpha(\textsc{nf}(f)) = \textsc{nf}(\alpha(f)) \equiv 0$, and hence, $\alpha(\textsc{nf}(f)) = \alpha(f')$ is identically zero. □