Operational models of piecewise-smooth systems

Andrew Sogokon
Carnegie Mellon University
Pittsburgh, PA, USA
asogokon@cs.cmu.edu

Khalil Ghorbal
INRIA
Rennes, France
khalil.ghorbal@inria.fr

Taylor T. Johnson
Vanderbilt University
Nashville, TN, USA
taylor.johnson@vanderbilt.edu

ABSTRACT

This paper studies ways of constructing meaningful operational models of piecewise-smooth systems (PWS). The systems we consider are described by polynomial vector fields defined on non-overlapping semi-algebraic sets, which form a partition of the state space. Our approach is to give meaning to motion in systems of this type by automatically synthesizing operational models in the form of hybrid automata (HA). Despite appearances, it is in practice often difficult to arrive at satisfactory HA models of PWS. The different ways of building operational models that we explore in our approach can be thought of as defining different semantics for the underlying PWS. These differences have a number of interesting nuances related to phenomena such as chattering, non-determinism, so-called mythical modes and sliding behaviour.

Keywords

piecewise-smooth systems, hybrid automata, operational models, discontinuous differential equations.

1. INTRODUCTION

Many processes in which smooth continuous motion can be interrupted by discrete events can be represented by ordinary differential equations (ODEs) with discontinuities. As such, they are part of a broader class of dynamical systems, known as hybrid (also cyber-physical) systems, which combine discrete and continuous behaviour under a unified framework[1]. Hybrid systems are increasingly used in modelling and analyzing the behaviour of modern control systems employing embedded devices.

Systems described by discontinuous ODEs are sometimes referred to as *piecewise-smooth systems* (PWS). Their representation has proved popular in the control systems community because it provides a concise and convenient notation. However, a discontinuous system of ODEs explicitly only conveys information about the continuous dynamics of the system, along with a set of regions where state evolution is smooth; the discrete transition behaviour of the system between these regions is not explicitly elaborated.

There exist a number of specification formalisms, such as *hybrid automata* [1] and *hybrid programs* [34], whose semantics is clearly defined and which can serve as operational models for hybrid systems. Hybrid automata in particular have proved to be very popular in the verification community. In a hybrid automaton, the discrete transition behaviour of the hybrid system is specified explicitly, which can often make these automata large and unwieldy even when specifying hybrid systems of relatively modest size. As a specification formalism, discontinuous ODEs provide a much more concise and manageable description of piecewise-smooth systems, albeit leaving many important details about its behaviour implicit.

Researchers working in computer science and control systems tend to put different emphasis on the importance of formal modelling and tend to use significantly different methods to model and reason about systems. One particular aspect of these differences is manifest in the temptation to treat hybrid automata naïvely as being merely syntactic variants of discontinuous ODEs when modelling piecewise-smooth systems. Subscribing to this view is, however, rather dangerous and can lead to unintended behaviour in the resulting models.

In this paper we study the challenges presented by the problem of transforming concise descriptions of piecewise-smooth systems in the form of discontinuous ODEs into formal operational models in the form of hybrid automata. Transformations that result in satisfactory models are, as we shall see, far from trivial to both formulate and effect. We develop automatic procedures for transforming piecewise-smooth systems with polynomial dynamics and semi-algebraic constraints into hybrid automata.

A number of different interpretations of the operational meaning of piecewise-smooth systems are possible, creating a degree of ambiguity about their intended behaviour (i.e. their semantics); this gives rise to significant differences in the form and the behaviour of hybrid automata models that one can construct. A number of important choices can be exercised in transforming PWS to HA in order to ensure that the resulting operational models reflect the desired interpretation.

1.1 Contributions

In this paper we describe a method for automatically constructing hybrid automata from descriptions of piecewise-smooth polynomial systems and thus build their operational models. We discuss aspects of the semantics of transitions directly related to phenomena such as chattering, non-determinism and the presence of so-called *mythical modes* in the underlying systems. We illustrate how our technique can be applied to model systems with so-called *sliding modes*.

[1]Indeed, some of the earliest research in hybrid systems, e.g. in the work of Witsenhausen [44], began by considering precisely the systems where there are no "jumps" in the continuous state, but abrupt changes in the *dynamics* are possible.

We conclude with a discussion of related work and an outlook for future research.

2. MATHEMATICAL PRELIMINARIES

2.1 Continuous systems and vector fields

A general n-dimensional autonomous system of first-order ODEs has the form:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \ldots, x_n), \\ &\vdots \\ \dot{x}_n &= f_n(x_1, x_2, \ldots, x_n), \end{aligned}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ are real-valued functions (typically C^1) for each $i = 1, \ldots, n$ and $\dot{x}_i$ denotes the derivative of x_i with respect to time, i.e. $\frac{dx_i}{dt}$. Such a system defines a *vector field* $\mathfrak{f} : \mathbb{R}^n \to \mathbb{R}^n$, where $\mathfrak{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ for any $\boldsymbol{x} \in \mathbb{R}^n$. We will denote autonomous systems of ODEs using the more concise vector field notation, i.e. by writing $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$.

In applications, it is often the case that the state of the system is required to only evolve within some prescribed set of "legal" states $M \subseteq \mathbb{R}^n$, which is known as the *mode invariant*, or *evolution constraint*. We will express this requirement concisely by writing $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x}),\ \boldsymbol{x} \in M$. When no evolution constraint is specified, M is assumed to be $\mathbb{R}^n$.

A *solution* to the initial value problem for the system of ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$ with initial value $\boldsymbol{x}_0 \in \mathbb{R}^n$ is a differentiable function $\boldsymbol{x} : (a, b) \to \mathbb{R}^n$, where $\boldsymbol{x}(t)$ is defined for all t within some non-empty extended real interval including zero, i.e. $t \in (a, b) \subseteq \mathbb{R} \cup \{\infty, -\infty\}$ where $a < 0 < b$, and such that $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and $\frac{d}{dt}\boldsymbol{x}(t) = \mathfrak{f}(\boldsymbol{x}(t))$ for all $t \in (a, b)$. In what follows, we will denote the solution $\boldsymbol{x}(t)$ by writing $\varphi_t(\boldsymbol{x}_0)$, to emphasize the initial value. If the function $\varphi_t(\boldsymbol{x}_0)$ is available in closed-form[2], one can analyze the temporal behaviour of the system initialized in the state $\boldsymbol{x}_0$ by analyzing the closed-form expression. In practice, however, it has long been established that explicit closed-form solutions to non-linear ODEs are highly uncommon [20].

Systems of ODEs whose right-hand sides are *locally Lipschitz continuous* (e.g. polynomial functions fall under this class) guarantee existence of unique solutions on some non-trivial time interval (a, b) for any initial value $\boldsymbol{x}_0 \in \mathbb{R}^n$ (by the Cauchy-Lipschitz/Picard-Lindelöf theorem; see e.g. [40]).

2.2 Piecewise-smooth vector fields

Given a partition of some set $M \subseteq \mathbb{R}^n$ into finitely many non-overlapping subsets $M_1, \ldots, M_m$, we consider a finite family of vector fields $\mathfrak{f}_i : \mathbb{R}^n \to \mathbb{R}^n$, where $i \in \{1, \ldots, m\}$. By assigning the vector field $\mathfrak{f}_i$ from this family to the set M_i for each $i = 1, \ldots, m$, we arrive at a vector field $\mathfrak{F} : M \to \mathbb{R}^n$ which is defined piecewise on M, i.e.

$$\mathfrak{F}(\boldsymbol{x}) = \begin{cases} \mathfrak{f}_1(\boldsymbol{x}) & \boldsymbol{x} \in M_1\,, \\ \quad\vdots & \\ \mathfrak{f}_m(\boldsymbol{x}) & \boldsymbol{x} \in M_m\,. \end{cases} \tag{1}$$

[2]By this we understand a *finite* expression in terms of polynomials and elementary special functions such as $\sin, \cos, \exp, \ln$, etc.

At this point, let us remark that while the sets $M_1, \ldots, M_m$ need not be differentiable manifolds, the corresponding vector fields $\mathfrak{f}_1, \ldots, \mathfrak{f}_m$ are defined on $\mathbb{R}^n$. It is therefore meaningful to speak about motion occurring within the manifold $\mathbb{R}^n$ according to the systems of ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$, but confined to the states within M_i. With this intuition, the vector field $\mathfrak{F}$ can be interpreted as describing a system of ODEs $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$ with a piecewise-defined (and potentially discontinuous) right-hand side, i.e. explicitly given by

$$\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x}) \tag{2}$$

To precisely describe the motion taking place (within the set M) in such a system, in general one may no longer call upon the classical notion of solution developed for continuous ODEs.[3] Indeed, there is no single universally agreed-upon definition of solution for systems of ODEs with discontinuities. Extensions of the classical notion, such as Carathéodory solutions, among others [19], have been suggested, but these differ in the way they model certain dynamic behaviours and therefore give different meaning (i.e. semantics) to systems. An excellent accessible survey of discontinuous ODEs and the various *generalized solution* concepts developed for them was given by Cortés in [8]. Intuitively, one expects generalized solutions to piecewise-smooth systems to be continuous functions of time, because these systems do not allow for discontinuous jumps in their (continuous) state, but with the differentiability requirement for the solution (in some way) appropriately relaxed. Solutions for more general classes of hybrid systems (which may allow discontinuous jumps in the state) are trickier, and require generalized time domains, such as hybrid time domains explored in the work of Sanfelice, Goebel and Teel [37, 18]. In our approach, we will not directly make use of these notions, relying instead on the semantics of hybrid automata (after Lygeros et al. [27]), which we shall describe presently.

2.3 Hybrid automata as operational models

Hybrid automata were first introduced by Alur et al. [1] as a formal specification language for hybrid systems. They provide operational models for hybrid systems in the same way that transition systems provide models for discrete computer programs, making it possible to give a precise mathematical description of their *execution*. We will employ the term *evolution* when speaking about hybrid *systems* (just as with continuous systems) and use the term *execution* only in the context of operational models, such as hybrid automata.

As formal models, hybrid automata have been used extensively in both modelling [12] and verification of properties in hybrid systems. Below we reproduce a very convenient definition of hybrid automata and their execution, due to Lygeros et al. [27]; for alternative definitions the interested reader is invited to consult [1, 22, 42].

Definition 1. *Formally, a* hybrid automaton HA *is given by an* 8*-tuple*

$$\mathrm{HA} = (Q, X, F, \mathrm{Init}, \mathrm{Dom}, E, G, R),$$

where the elements are as follows:

- $Q = \{q_1, q_2, \ldots, q_m\}$ *is a finite set of* discrete states*,*
- $X = \mathbb{R}^n$ *is a set of* continuous states*,*

[3]E.g. it is continuity of the right-hand side that guarantees the existence of solutions (by Peano's theorem).

- $F : Q \times X \to \mathbb{R}^n$ *is a* vector field,
- $\text{Init} \subseteq Q \times X$ *is a set of* initial states,
- $\text{Dom} : Q \to 2^X$ *is a* mode domain *(also* invariant*),*
- $E \subseteq Q \times Q$ *is a set of* edges *(also* discrete transitions*),*
- $G : E \to 2^X$ *is a* guard condition,
- $R : E \times X \to 2^X$ *is a* reset map.

Standard assumptions with this definition are that guard conditions are non-empty whenever they are specified, i.e. for all $e \in E$ it is the case that $G(e) \neq \emptyset$ and also that reset maps can only take the system to a genuine continuous state, i.e. for all $x \in G(e)$, $R(e, x) \neq \emptyset$.

2.3.1 Semantics of hybrid automata

A *hybrid time trajectory* is a finite or infinite sequence of contiguous time intervals starting at 0, where the end points are interpreted as times at which a discrete *event*, such as a transition, occurs. More formally, following [27], a hybrid time trajectory is a sequence of intervals $\tau = \{I_i\}_{i=0}^{N}$, for which $I_i = [\tau_i, \tau_i']$ for all $i < N$, where $N \in \mathbb{N} \cup \{\infty\}$, and $\tau_i \leq \tau_i' = \tau_{i+1}$ for all i. If the sequence is finite, i.e. if $N < \infty$, then either $I_N = [\tau_N, \tau_N']$ or $I_N = [\tau_N, \tau_N')$. Intuitively, one may think of τ_i as the times at which discrete transitions occur.

An *execution* (or a *run*) of a hybrid automaton is defined to be the triple $(\tau, q, \varphi_t^i(\boldsymbol{x}))$, where τ is a hybrid time trajectory, $q : \langle\tau\rangle \to Q$, where $\langle\tau\rangle$ is defined to be the set $\{0, 1, \ldots, N\}$ if τ is finite and $\{0, 1, \ldots\}$ otherwise [27], is a map and $\varphi_t^i(\boldsymbol{x})$ is a collection of differentiable functions $\varphi_t^i(\boldsymbol{x}) : I_i \to \mathbb{R}^n$ such that $(q(0), \varphi_0^0(\boldsymbol{x})) \in \text{Init}$ and for all $t \in [\tau_i, \tau_i')$ it is the case that $\dot{\boldsymbol{x}} = F(q(i), \varphi_t^i(\boldsymbol{x}))$ and $\varphi_t^i(\boldsymbol{x}) \in \text{Dom}(q(i))$. It is also required that transitions respect the guards and the reset maps, i.e. $e = (q(i), q(i+1)) \in E$, $\varphi_{\tau_i'}^i(\boldsymbol{x}) \in G(e)$ and $(\varphi_{\tau_i'}^i(\boldsymbol{x}), \varphi_{\tau_{i+1}}^{i+1}(\boldsymbol{x})) \in R(e)$.

3. PROBLEM OVERVIEW

This section gives an overview of the challenges associated with modelling piecewise-smooth systems using the hybrid automaton formalism.

If one were to naïvely translate a system of the form shown in (2) into a hybrid automaton, as a first step one could simply take the sets $M_1, \ldots, M_m$ to be the mode invariants of the discrete states in the automaton (i.e. by letting Dom in Definition 1 be $q_i \mapsto M_i$ for each $i = 1, \ldots, m$) and set the continuous dynamics within these modes to be governed by the differential equation $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$, i.e. letting the vector field $F(q_i, \boldsymbol{x}) = \mathfrak{f}_i(\boldsymbol{x})$ for each $i = 1, \ldots, m$, respectively. The resulting hybrid automaton would have $|Q| = m$ discrete states and no discrete transitions between them. Clearly, this would not be an adequate model, since the original system will most likely evolve into and out of the sets $M_1, \ldots, M_m$. This fact raises an immediate problem: in order to have discrete transitions in the hybrid automaton one is required to specify their enabling guards, i.e. sets of states *within the mode invariant of the outgoing discrete state* in which a discrete transition is possible.

To appreciate the problem more fully, let us consider a simple 1-dimensional system defined on the partition of the real line $\mathbb{R}$ into three regions: $x < 0$, $x = 0$ and $x > 0$, and where the vector fields are respectively given by $\mathfrak{f}_1(x) = 1$, $\mathfrak{f}_2(x) = 2$ and $\mathfrak{f}_3(x) = 3$ (i.e. $\dot{x} = 1$, $\dot{x} = 2$, and $\dot{x} = 3$) inside each region. Clearly, one expects this system, when started inside $x < 0$, to transition into $x = 0$ and then to $x > 0$. In order for a hybrid automaton to faithfully model the behaviour of this system, we require two discrete transitions that take the state from $x < 0$ to $x = 0$ and from $x = 0$ to $x > 0$; however, in the former transition it is not possible to specify $x = 0$ to be the guard (as shown in Fig. 1), since this set lies outside of the mode invariant $x < 0$ of the outgoing discrete state. It is possible to declare the transition guard to be in some thin layer near the boundary, e.g. $\delta < x < 0$, where $-\frac{1}{\delta}$ is large, but any such choice of δ would be rather arbitrary in the general case. Furthermore, there would remain another important problem, this time with the latter transition from $x = 0$ to $x > 0$: in order to make such a transition without creating discontinuities (in this case "gaps") in the trajectory through reset maps, the state of the system needs to lie within the mode invariant of the destination discrete state when the transition guard is enabled. The guard $x = 0$ is thus also unsuitable in this case and there is no easy fix to this problem.

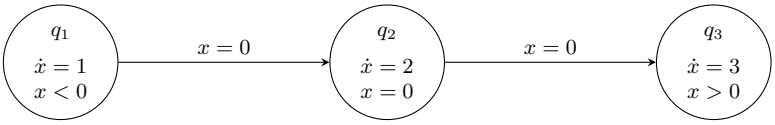


Figure 1: Naïve construction (mode transitions impossible).

Instead of using $M_1, \ldots, M_m$ as mode invariants in the automaton, one may instead opt to use their closures $\overline{M_1}, \ldots, \overline{M_m}$ with a view to enabling the transition guards on appropriate subsets of the boundaries $\partial M_1, \ldots, \partial M_m$, which would now lie inside the corresponding mode invariants. This approach, while conceptually simple, has a number of serious deficiencies and results in hybrid automata that exhibit *chattering runs*, i.e. can perform an arbitrary number of discrete transitions without advancing the continuous state or time.

The use of set closures additionally overlooks an important computational drawback, which is that closures are typically very difficult to compute exactly for important classes for sets, such as e.g. *semi-algebraic sets* (i.e. sets described by a finite Boolean combination of polynomial equalities and inequalities; see e.g. [28, Definition 8.6.1]).

REMARK 1. *In general for semi-algebraic sets,* $\overline{S}$ cannot *be obtained from* S *by syntactically replacing every instance of strict inequalities in its description by non-strict inequalities (e.g.* $x^3 - x^2 \geq 0$ *is* ***not*** *the closure of* $x^3 - x^2 > 0$*) [3, Remark 3.2]. The closure of a semi-algebraic set* S *is given by the set*

$$\overline{S} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \forall\, r > 0.\ \exists\, \boldsymbol{y} \in S.\ \|\boldsymbol{y} - \boldsymbol{x}\|^2 < r^2\},$$

where the norm $\|\|$ *is the standard Euclidean distance (see e.g. [3, Chapter 3]). Let the set* S *be described by a quantifier-free formula in the theory of real arithmetic with free variables* $x_1, \ldots, x_n$*. By performing a syntactic substitution of the free variables* x_i *by* y_i *(*$i = 1, \ldots, n$*) everywhere in the formula, one obtains a quantifier-free formula in the variables* $y_1, \ldots, y_n$*. The closure* $\overline{S}$ *can then be characterized by the formula*

$$\forall\, r > 0.\ \exists\, y_1, \ldots, y_n.\ S \wedge (y_1 - x_1)^2 + \cdots + (y_n - x_n)^2 < r^2,$$

where $x_1, \ldots, x_n$ *are treated as fresh free variables and* r *is*

a fresh bound variable. It is therefore possible to apply real quantifier elimination to reduce this formula to an equivalent one that is quantifier-free and features only the free variables $x_1, \ldots, x_n$.

Real quantifier elimination (QE) is computationally expensive, having complexity doubly-exponential in the number of quantifier alternations [10]. The popular CAD algorithm *for real QE, is doubly-exponential in the number of variables [4], which makes it impractical for problems with a large number of variables (using currently existing implementations).*

In this paper we pursue a very different approach to constructing HA operational models of PWS, which does not require computing set closures , but instead requires only the "relevant" subsets on their boundaries and relies fundamentally on the notion of "entry" and "exit" sets that will be the subject of the following section.

4. OPERATIONAL MODELS

This section will review some important definitions before presenting an algorithm for automatically generating HA operational models of PWS.

4.1 Fundamental Definitions

We start by defining an important set that will shortly become of interest:

Definition 2 (Inward Crossing Set).

$$\begin{aligned}\mathrm{Enter}_{\mathfrak{f}}(S) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \exists\ \varepsilon > 0.\\ \forall\ t \in (0,\varepsilon).\ \varphi_t(\boldsymbol{x}) \in S\ \wedge\ \forall\ t \in (-\varepsilon, 0).\ \varphi_t(\boldsymbol{x}) \notin S\}\end{aligned}$$

where $\varphi_t(\boldsymbol{x})$ *denotes the (unique) solution to the locally Lipschitz-continuous system of ODEs* $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$.

The intuition, as suggested by the name, is that $\mathrm{Enter}_{\mathfrak{f}}(S)$ describes the states at which the system is about to evolve inside S, after having only just evolved outside of S. Likewise, we define $\mathrm{Exit}_{\mathfrak{f}}(S)$ to be the set of states at which the system is about to evolve outside of S, after having only just evolved inside, i.e. $\mathrm{Exit}_{\mathfrak{f}}(S) \equiv \mathrm{Enter}_{\mathfrak{f}}(\neg S)$, where $\neg S := \mathbb{R}^n \setminus S$. Note that such states need not necessarily lie within S itself and may lie outside; however, they necessarily lie on the boundary of S. We observe that the crossing set, by its very definition, can be expressed by means of one fundamental building block.

Lemma 1 (Crossing Set Deconstruction).
$\mathrm{Enter}_{\mathfrak{f}}(S) \equiv \mathrm{In}_{\mathfrak{f}}(S)\ \cap\ \mathrm{In}_{-\mathfrak{f}}(\neg S)$, *where*

$$\mathrm{In}_{\mathfrak{f}}(S) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \exists\ \varepsilon > 0.\ \forall\ t \in (0,\varepsilon).\ \varphi_t(\boldsymbol{x}) \in S\}.$$

Intuitively, $\mathrm{In}_{\mathfrak{f}}(S)$ denotes the states in $\mathbb{R}^n$ from which the motion of the system takes place within the set S for some time segment immediately following 0 (i.e. in the immediate future). By analogy, when considering $-\mathfrak{f}$, the reverse of the vector field $\mathfrak{f}$, $\mathrm{In}_{-\mathfrak{f}}(S)$ denotes the states in $\mathbb{R}^n$ from which the motion of the system took place within the set S for some time segment immediately preceding 0 (i.e. in the immediate past).

In the special case when the system $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$ has polynomial right-hand sides and S is a semi-algebraic set, the sets $\mathrm{In}_{\mathfrak{f}}(S)$, and hence $\mathrm{In}_{-\mathfrak{f}}(S)$, are also semi-algebraic and can be computed exactly (a result due to Liu et al. [25]). As a consequence, the sets $\mathrm{Enter}_{\mathfrak{f}}(S)$ and $\mathrm{Exit}_{\mathfrak{f}}(S)$ are also computable and semi-algebraic under these assumptions.

We stress the fact that the boundary of S need not be included in $\mathrm{Enter}_{\mathfrak{f}}(S) \cup \mathrm{Exit}_{\mathfrak{f}}(S)$. In particular, the set

$$\mathrm{Bounce}_{\mathfrak{f}}(S) \equiv \mathrm{In}_{\mathfrak{f}}(S)\ \cap\ \mathrm{In}_{-\mathfrak{f}}(S)$$

describes those states that may leave S momentarily at a point while evolving within S before and after the "bounce" and can therefore lie outside of $\mathrm{Enter}_{\mathfrak{f}}(S) \cup \mathrm{Exit}_{\mathfrak{f}}(S)$. Appendix B provides an illustration to help develop some intuition about the meaning of these sets.

4.2 Generating Hybrid Automata

We now have at our disposal the machinery necessary for building operational models of piecewise-smooth systems $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$, i.e. systems of the form:

$$\dot{\boldsymbol{x}} = \begin{cases} \mathfrak{f}_1(\boldsymbol{x}) & \boldsymbol{x} \in M_1\,, \\ \quad\vdots & \\ \mathfrak{f}_m(\boldsymbol{x}) & \boldsymbol{x} \in M_m\,. \end{cases}$$

Given such a system, our aim is to synthesize a hybrid automaton that provides an adequate model of the behaviour of the system. To do this, our approach we will be to first *augment* the original invariant modes of the system M_i with additional states, before they can become mode invariants of a hybrid automaton. This step requires a definition.

Definition 3. *Given a semi-algebraic set* $S \subseteq \mathbb{R}^n$ *and a system of polynomial ODEs* $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$, *the* augmented set *of* S *with respect to this system,* $\mathrm{Aug}(S, \mathfrak{f})$, *is defined by*

$$\mathrm{Aug}(S, \mathfrak{f}) \equiv S \cup \mathrm{Enter}_{\mathfrak{f}}(S) \cup \mathrm{Exit}_{\mathfrak{f}}(S) \cup \mathrm{Bounce}_{\mathfrak{f}}(S)\,.$$

In the context of piecewise-smooth systems of the form $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$, *whenever we wish to augment the set* M_i *with respect to the system* $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$, *we shall adopt a more concise notation and simply write* $\mathfrak{M}_i$, *i.e.* $\mathfrak{M}_i \equiv \mathrm{Aug}(M_i, \mathfrak{f}_i)$.

The definition extends each invariant mode S with its "entry", "exit" and "bounce" sets. The main intuition being that if the system was to enter or exit the mode invariant S with respect to the dynamics $\mathfrak{f}$ then it will do so by necessarily crossing those sets. The set $\mathrm{Bounce}_{\mathfrak{f}}(S)$ allows the evolution to continue within S after "momentarily exiting" S. If in addition the mode invariants have to satisfy a global constraint M, then it should be accounted for by intersecting it with $\mathrm{Aug}(M_i, \mathfrak{f}_i)$.

Algorithm 1 gives a pseudocode procedure for generating a hybrid automaton $\mathrm{HA}_{\mathfrak{F}}$. The procedure begins constructing the automaton by first creating m distinct discrete states $Q = \{q_1, \ldots, q_m\}$ (line 1), defining X to be $\mathbb{R}^n$ (line 2) and creating a set of edges (i.e. transitions) E by computing the Cartesian product $Q \times Q$ and removing all edges of the form (q_i, q_i), i.e. removing all stuttering/self-looping transitions (line 3). It then proceeds to initially assign the empty set to all the remaining variables on line 4. The algorithm then proceeds to create the modes of the hybrid automaton $\mathrm{HA}_{\mathfrak{F}}$ in its first loop (lines 5–12), where it builds the extensional definition of F by assigning the vector field $\mathfrak{f}_i$ to the discrete state q_i (line 6), augmenting each set M_i with its "entry", "exit" and "bounce" sets (line 7) and using this to build an extensional definition of the Dom mapping (line 8) which provides the mode invariant $\mathfrak{M}_i$ for each state q_i of the hybrid automaton. Lines 9–11 are responsible for converting

the initial set of states for the PWS into one for the hybrid automaton (this step can in fact be factored out of the algorithm and performed separately).

The second loop of the algorithm (lines 13–16) constructs the discrete transitions and is responsible for defining the discrete transition behaviour of the resulting automaton. The loop iterates through all the transitions constructed on line 3 and defines the guards (line 14) and reset maps (line 15) associated with each transition. The reset map is chosen to be the identity and therefore does not affect the state of the system upon taking any transition. Different possible choices for the guard condition $\mathrm{GC}(i,j)$ (line 14) are discussed in the next section.

Data: $M \subseteq \mathbb{R}^n, M_1, \ldots, M_m \subseteq M, \mathfrak{f}_1, \ldots, \mathfrak{f}_m : \mathbb{R}^n \to \mathbb{R}^n, X_0 \subseteq M$
Result: Hybrid automaton $\mathrm{HA}_{\mathfrak{F}}$
1 $Q \leftarrow \{q_1, \ldots, q_m\}$;
2 $X \leftarrow \mathbb{R}^n$;
3 $E \leftarrow Q \times Q \setminus \{(q_1,q_1),(q_2,q_2),\ldots(q_m,q_m)\}$;
4 $\mathrm{Init}, \mathrm{Dom}, F, G, R \leftarrow \emptyset$;
5 **foreach** $i \in \{1,\ldots,m\}$ **do**
6 $\quad F \leftarrow F \cup \{((q_i,\boldsymbol{x}),\mathfrak{f}_i(\boldsymbol{x}))\}$;
7 $\quad \mathfrak{M}_i \leftarrow \mathrm{Aug}(M_i,\mathfrak{f}_i) \cap M$;
8 $\quad \mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{q_i \mapsto \mathfrak{M}_i\}$;
9 $\quad$ **if** $X_0 \cap M_i \neq \emptyset$ **then**
10 $\quad\quad \mathrm{Init} \leftarrow \mathrm{Init} \cup \{(q_i,\boldsymbol{x}) \mid \boldsymbol{x} \in M_i \cap X_0\}$
11 $\quad$ **end**
12 **end**
13 **foreach** $e = (q_i,q_j) \in E$ **do**
14 $\quad G \leftarrow G \cup \{(e, \mathrm{GC}(i,j))\}$;
15 $\quad R \leftarrow R \cup \{((q_i,q_j), \boldsymbol{x} \mapsto \{\boldsymbol{x}\})\}$
16 **end**
17 **return** $(Q, X, F, \mathrm{Init}, \mathrm{Dom}, E, G, R)$

Algorithm 1: Procedure for synthesizing a HA from PWS.

4.3 Discrete Transition Behaviour

The transition guard $G(e) = \mathrm{GC}(i,j)$, $i \neq j$, for the transition $q_i \to q_j$ entirely determines the discrete transition behaviour of the automaton. In what follows, we will consider three choices for this formula.

REMARK 2. *We stress the fact that these are by no means the only possible semantics; they are primarily meant to exemplify how the method works and how one can adapt Algorithm 1 to generate operational models exhibiting qualitatively different behaviours.*

Recall that mode q_i (resp. q_j) has mode invariant $\mathfrak{M}_i$ (resp. $\mathfrak{M}_j$).

$$\begin{aligned} \mathrm{I} &\equiv \mathfrak{M}_i \wedge \mathfrak{M}_j \wedge \mathrm{In}_{\mathfrak{f}_j}(M_j) \\ \mathrm{II} &\equiv \mathrm{I} \wedge \neg\mathrm{Enter}_{\mathfrak{f}_i}(M_i) \wedge \neg\mathrm{Bounce}_{\mathfrak{f}_j}(M_j) \\ \mathrm{III} &\equiv \mathrm{I} \wedge \neg\mathrm{In}_{\mathfrak{f}_i}(M_i) \end{aligned}$$

Informally, these formulas are characterizing the sets of states where (1) the augmented mode invariants $\mathfrak{M}_i$ and $\mathfrak{M}_j$ intersect to allow for continuous transitions and (2) where the trajectory of the system in mode q_j can evolve within that mode for some time, hence the intersection with $\mathrm{In}_{\mathfrak{f}_j}(M_j)$. Formulas II and III impose additional constraints on the guard. Namely, formula II additionally requires that the guard does not feature states in the intersection of the "entering" set of the outgoing state and the "bounce" set of the incoming state. As will be seen in later sections, this is primarily done to eliminate so-called *chattering* in the model. Formula III is different in that it only enables a transition guard if no further continuous motion is possible within the mode. This has the effect that transitions *must* be taken precisely when they are enabled.

Replacing $\mathrm{GC}(i,j)$ in line 14 of Algorithm 1 by formula I, II or III will generally result in a different operational model which can exhibit very different behaviour. In what follows, we will refer to these formulas as respectively defining guard conditions of *type* I, II and III.

4.4 Computability

An operational model of a PWS in the form of a hybrid automaton is computable using Algorithm 1 whenever the vector fields $\mathfrak{f}_1, \ldots, \mathfrak{f}_m$ are polynomial and the sets $M_1, \ldots, M_m, M$ and X_0 are semi-algebraic.

We recall that a set is semi-algebraic if it is characterized by a finite Boolean combination of polynomial equations and inequalities. Thus, the formula $x_1 > 0 \wedge x_2 = 0 \vee x_2^3 - x_1 \leq 0$, where the symbols x_1, x_2 are interpreted over the real numbers, characterizes the semi-algebraic set $\{(x_1,x_2) \in \mathbb{R}^2 \mid x_1 > 0 \wedge x_2 = 0 \vee x_2^3 - x_1 \leq 0\}$. It suffices to consider formulas without quantifiers, e.g. $\forall$ and $\exists$, since the theory of real arithmetic admits quantifier elimination [39] and therefore any formula featuring quantifiers may be reduced to an equivalent quantifier-free formula using a terminating procedure.[4]

It was shown in [25] that the set $\mathrm{In}_{\mathfrak{f}}(S)$ can be computed exactly by employing higher-order *Lie derivatives* and the *ascending chain property* of Noetherian rings. A Lie derivative of a polynomial $p : \mathbb{R}^n \to \mathbb{R}$ with respect to the polynomial vector field $\mathfrak{f} : \mathbb{R}^n \to \mathbb{R}^n$ is also a polynomial denoted $\mathfrak{L}_{\mathfrak{f}}(p)$ and defined as $\mathfrak{L}_{\mathfrak{f}}(p) \equiv \nabla p \cdot \mathfrak{f} = \sum_{i=1}^n \frac{\partial p}{\partial x_i} f_i$. It gives the total derivative of the p with respect to time, i.e. the rate of change of p along the solutions to the corresponding system of ODEs. Higher-order Lie derivatives are defined inductively as $\mathfrak{L}_{\mathfrak{f}}^k(p) = \mathfrak{L}_{\mathfrak{f}}(\mathfrak{L}_{\mathfrak{f}}^{k-1}(p))$, with $\mathfrak{L}_{\mathfrak{f}}^0(p) = p$.

In addition to [25], a more accessible illustrated description of the main idea behind the procedure may be found in [17, Section 5.4]. Similar ideas employing higher-order Lie derivatives and ascending chains of ideals have also appeared elsewhere, e.g. [33, 16]. Appendix A describes the construction of $\mathrm{In}_{\mathfrak{f}}(S)$ in more detail. As a consequence, the sets $\mathrm{Enter}_{\mathfrak{f}}(S)$, $\mathrm{Exit}_{\mathfrak{f}}(S)$ and $\mathrm{Bounce}_{\mathfrak{f}}(S)$ are also semi-algebraic and may be computed exactly using a terminating algorithm.

5. DYNAMIC PROPERTIES OF OPERATIONAL MODELS

This section will illustrate some of the dynamic phenomena observed in the operational models that we can compute using Algorithm 1 and will discuss some of the differences in their behaviour when different types of guard conditions are employed.

5.1 Non-determinism

[4] A number of algorithms have been developed since Tarski's [39] and Seidenberg's [38] seminal papers, e.g. the CAD algorithm due to Collins [6].

Non-determinism occurs when the piecewise-smooth system may evolve inside more than one of its modes. At first sight, this may look surprising because in a PWS any state $\boldsymbol{x} \in M$ belongs to exactly one region M_i, $i \in \{1, \ldots, m\}$, if one indeed has a mathematical partition of M into these regions, and therefore there cannot be any ambiguity in the choice of the ODEs that should govern the continuous state evolution at $\boldsymbol{x}$. However, generalized solutions to the system at $\boldsymbol{x}$ may not be unique even when the ODEs inside each mode all have unique solutions when considered separately. This is mirrored in our operational models, where we augment the regions M_i with their respective "entry" and "exit" sets to obtain the augmented mode invariants $\mathfrak{M}_i$ in the hybrid automaton. One may face a scenario where $\boldsymbol{x} \in \mathfrak{M}_i$ and $\boldsymbol{x} \in \mathfrak{M}_j$, with $i \neq j$, and both transition guards between the two states q_i and q_j are enabled. For instance, $\boldsymbol{x}$ may lie in a region where both $\mathfrak{M}_i \wedge \mathfrak{M}_j \wedge \mathrm{In}_{\mathfrak{f}_i}(M_i)$ and $\mathfrak{M}_i \wedge \mathfrak{M}_j \wedge \mathrm{In}_{\mathfrak{f}_j}(M_j)$ hold true.

The standard semantics of transition guards of hybrid automata is that they *enable* transitions, but do not force them (this is known as *non-urgent*, or *may* semantics [13]). Thus, while there is no ambiguity about the initial discrete state of the hybrid automaton for any given $\boldsymbol{x} \in M$, the system is free to take an enabled transition immediately after it starts evolving. This non-determinism can be informally understood as capturing the "instability" that arbitrarily small perturbations in the initial state can cause in the mode switching behaviour of the piecewise-smooth system.

5.2 Chattering Runs

A phenomenon known as *chatter* is traditionally associated with so-called *Zeno behaviour* that can occur in mathematical models of hybrid systems and can present a problem for their simulation and verification. This behaviour is non-physical and manifests itself in the possibility of performing an infinite number of transition in a finite amount of time.

For example, a hybrid automaton will admit *chattering runs* whenever for two distinct states q_i and q_j there are transitions in both directions such that their respective transition guards have non-empty intersection. Any state $\boldsymbol{x}$ within this intersection can shuttle back and forth between the states q_i and q_j an arbitrary (though perhaps not infinite) number of times.

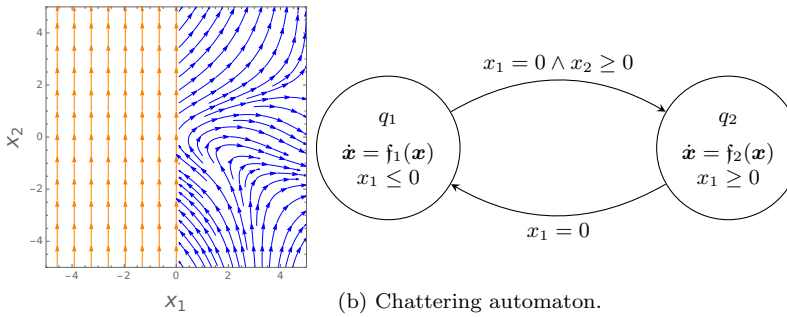
As an example, let us consider a PWS with two modes:

$$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = 0\,, \\ \dot{x}_2 = x_2^2 + 2\,, \end{cases} \qquad x_1 \leq 0\,,$$

$$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = x_1 + 4x_2 - x_1 x_2\,, \\ \dot{x}_2 = x_2^2 - x_1 + 2\,, \end{cases} \qquad x_1 > 0.$$

By running Algorithm 1 with guard conditions of type I, one obtains a hybrid automaton shown in Fig. 2b. This automaton admits chattering runs because on the set characterized by $x_1 = 0 \wedge x_2 \geq 0$ the guards for transitions between both modes are enabled simultaneously and the system may thus shuttle back and forth arbitrarily may times without advancing in (continuous) time. However, if one were to employ guard conditions of type II, the resulting automaton (Fig. 3) would be chatter-free.

Since infinite Zeno executions cannot in practice be realized, it is common to consider only the *non-Zeno executions* when modelling systems using hybrid automata [22, 11] (this is also the case with hybrid programs [35]).



(a) Phase portrait.

(b) Chattering automaton.

Figure 2: Chattering in the presence of non-determinism.

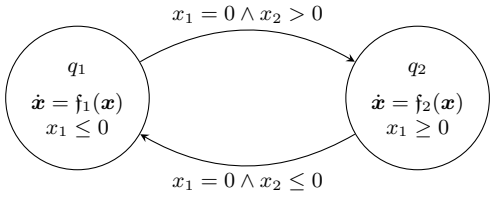


Figure 3: Chatter-free automaton.

We should note that infinite chattering runs are a special kind of Zeno behaviour, which some authors distinguish from the more involved *genuine Zeno* behaviour (see e.g. [2]). Chatter-free automata may still suffer from this latter type of Zeno behaviour. Detecting and eliminating genuine Zeno behaviour in hybrid automata is highly non-trivial and is the focus of ongoing research.

5.3 Mythical Modes

A piecewise-smooth system may feature a mode M_i inside which it is altogether impossible to evolve continuously according to its respective system of ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$. More precisely, it is possible that $M_i \cap \mathrm{In}_{\mathfrak{f}_i}(M_i) = \emptyset$. Inside such a mode, the (continuous) state of the system remains invariant and may only change by switching into a different mode; such a mode is sometimes called *mythical* [30, 31]. For example, in a system where the state space is the real line $\mathbb{R}$ that is partitioned into 3 modes $x < 0$, $x = 0$ and $x > 0$ where the dynamics is respectively $\dot{x} = 1$, $\dot{x} = 2$ and $\dot{x} = 3$, the mode $x = 0$ is mythical.

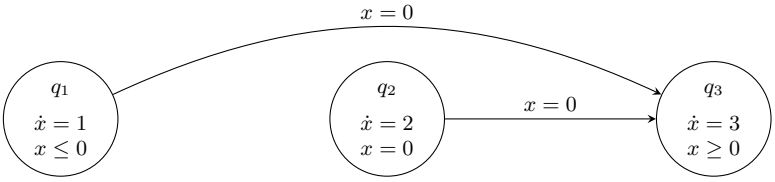


Figure 4: Mythical mode q_2.

Following our approach, the mode invariants for the hybrid automaton are augmented to be $x \leq 0$, $x = 0$ and $x \geq 0$ respectively, and a transition from $x \leq 0$ into $x \geq 0$ is possible without ever visiting the mythical mode. In general, in hybrid automata constructed using our method (e.g. Fig. 4 where only possible transitions are depicted with their guards) it is impossible to transition into mythical modes

with any of the three types of guard conditions.

5.4 Sliding Modes

In control systems literature, it is not uncommon to encounter systems of the form

$$\dot{\boldsymbol{x}} = \begin{cases} \mathfrak{f}_1(\boldsymbol{x}) & s(\boldsymbol{x}) > 0\,, \\ \mathfrak{f}_2(\boldsymbol{x}) & s(\boldsymbol{x}) < 0\,, \end{cases}$$

where $s : \mathbb{R}^n \to \mathbb{R}$ is some differentiable (often polynomial) function. These and similar systems are sometimes termed *variable structure systems* (VSS) and have been applied in discontinuous non-linear control strategies, known as *variable structure control* (VSC). A phenomenon known as *sliding motion* lies at the heart of an important class of variable structure control, known as *sliding mode control* (SMC), which, broadly speaking, achieves system order reduction by steering the trajectories of an n-dimensional system onto an $n-1$ dimensional switching hyper-surface in the system's state space, defined by $s = 0$. The so-called *sliding motion* taking place on the hyper-surface corresponds to the infinitely-fast switching between the modes governing the evolution on either side of the surface [45], i.e. inside regions where $s > 0$ and $s < 0$.

Remark 3. *Note however, that the description of the system may not explicitly prescribe any dynamics on the* switching surface $s = 0$ *itself.*

In practice, sliding motions are often modelled by introducing a so-called *equivalent control* on the switching surface; this is achieved by letting

$$\dot{\boldsymbol{x}} = \mathfrak{f}_s(\boldsymbol{x}) = \frac{\mathfrak{f}_1(\boldsymbol{x}) + \mathfrak{f}_2(\boldsymbol{x})}{2} + u_{eq}\frac{\mathfrak{f}_1(\boldsymbol{x}) - \mathfrak{f}_2(\boldsymbol{x})}{2}\,,$$

where $u_{eq} = \dfrac{\mathfrak{L}_{\mathfrak{f}_1}(s) \;+\; \mathfrak{L}_{\mathfrak{f}_2}(s)}{\mathfrak{L}_{\mathfrak{f}_2}(s) \;-\; \mathfrak{L}_{\mathfrak{f}_1}(s)}$, be the sliding dynamics on the regions of surface $s = 0$ where there is sliding [32, 41].

Let us consider a 2-dimensional non-linear system with a 1-dimensional sliding mode that was obtained by applying an equivalent control. The system is given by:

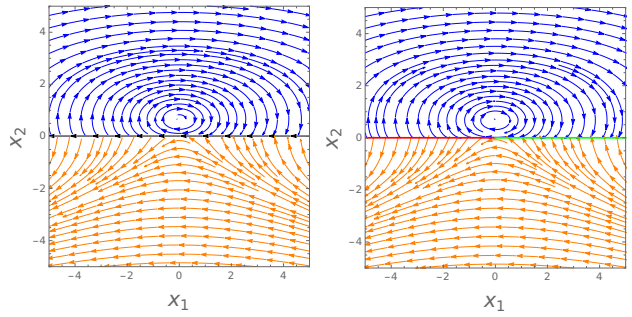
$$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = x_2^3 + \frac{3x_2^2}{8} + \frac{3x_2}{64} - \frac{255}{512}\,, \\ \dot{x}_2 = -\frac{x_1}{8} - x_1 x_2\,, \end{cases} \qquad x_2 > 0\,,$$

$$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = -2x_2^3 + \frac{9x_2^2}{8} + \frac{123x_2}{320} - \frac{303}{640}\,, \\ \dot{x}_2 = 0\,, \end{cases} \qquad x_2 = 0\,,$$

$$\dot{\boldsymbol{x}} = \mathfrak{f}_3(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = x_2^3 - \frac{3x_2^2}{2} + \frac{3x_2}{4} - \frac{3}{8}\,, \\ \dot{x}_2 = \frac{x_1}{2} - x_1 x_2\,, \end{cases} \qquad x_2 < 0\,.$$

Sliding occurs on the set characterized by $x_2 = 0$ and $\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x})$ is the equivalent control dynamics which steers the system along the surface $x_2 = 0$ (Fig. 5a). The system exhibits both stable and unstable sliding behaviour, which can be observed in the phase portrait, as shown in Fig. 5b. Roughly speaking, in the neighbourhoods of states where the sliding mode is stable the vector fields are "pointing towards" the sliding set, whereas in the neighbourhood of states where it is unstable the vector fields are "pointing outwards" away from the set.

For this system, different types of guard conditions lead to radically different operational models. The resulting hybrid automata employing guard conditions of type I, II and III are respectively shown in Fig. 6, Fig. 7 and Fig. 8.



(a) Phase portrait. (b) Stable and unstable sliding.

Figure 5: Piecewise-smooth system $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$ with a sliding mode at $x_2 = 0$ that is unstable when $x_1 > 0$ (shown in red) and a stable when $x_1 < 0$ (in green).

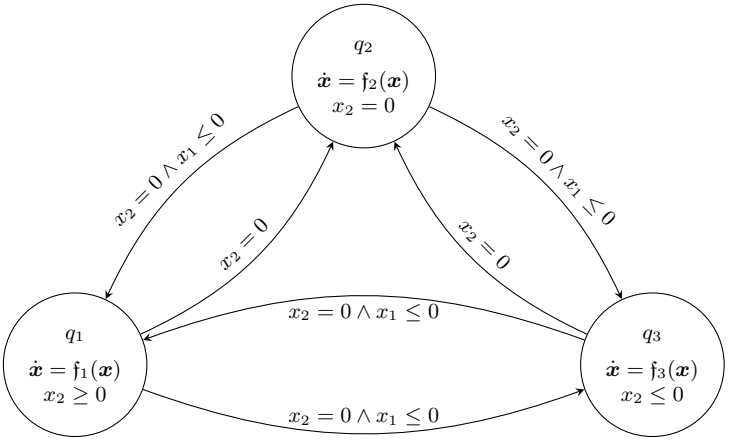


Figure 6: Hybrid automaton model with guard conditions of type I.

The three automata differ in the way they model non-determinism in the system. In particular guard conditions of type III result in the automaton in Fig. 8, which is completely deterministic and only models the stable sliding taking place in the system; there is no non-determinism corresponding to unstable sliding in this operational model. In practice, this behaviour is un-physical because unstable motions can leave the unstable sliding mode under arbitrarily small perturbations in the state or the vector field. As such, this operational model represents a mathematical idealization which is of little use when modelling physical systems. However, if physical considerations are unimportant, the model is interesting because it has the property that discrete transitions are taken precisely when they are enabled, in a way that is analogous to some non-standard *urgent/must* semantics for transition guards of hybrid automata.

The hybrid automaton in Fig. 7 models both stable and unstable sliding and is additionally chatter-free, whereas the automaton in Fig. 6 admits chattering runs when the continuous state is at the origin. Of all these operational models, the one employing guard conditions of type II (in Fig. 7) is perhaps the most physically meaningful and faithful to the intended behaviour of the system.

6. OUTLOOK AND RELATED WORK

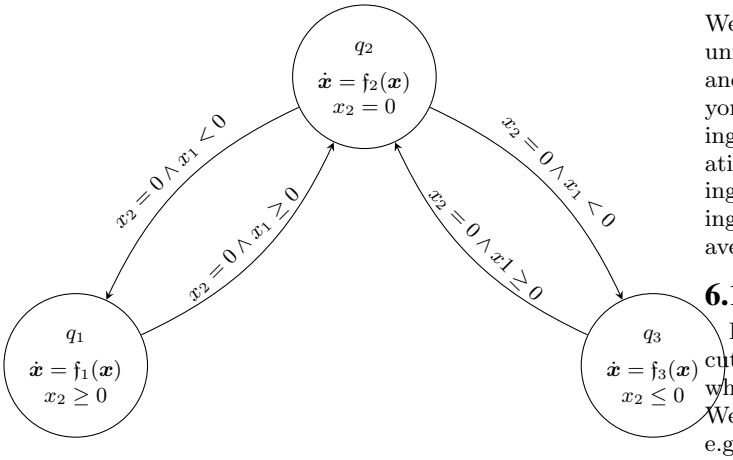


Figure 7: Hybrid automaton model with guard conditions of type II.

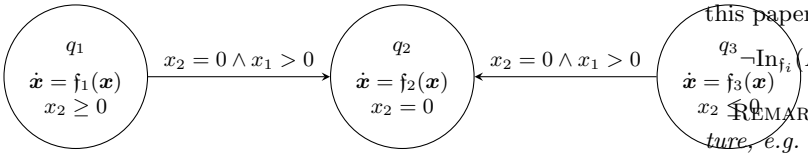


Figure 8: Hybrid automaton model with guard conditions of type III.

Having automatic means of computing operational models of systems which can be concisely specified (but whose operational models require an unreasonable amount of effort and care to explicitly write down manually) is a significant enabling factor. In general, computing adequate hybrid automaton models of systems is highly non-trivial [29]. The examples used in this paper are very simple and are intended to highlight differences between the different models; more interesting examples of PWS lead to automata that are indeed quite formidable. We have implemented our HA synthesis algorithm in Mathematica and are able to generate automata in the format of the verification tool SpaceEx [14].[5]

The hybrid automata we are able to generate can provide suitable models for addressing the problem of verification (e.g. of safety and liveness properties) and benefit from a large and growing number of software tools developed to verify or simulate hybrid systems [14, 24, 5, 15, 43, 36]. Verification technology for hybrid systems has improved tremendously in the last two decades; however, in much of existing work there are significant restrictions on the form of hybrid automata, such as e.g. only allowing linear ODEs to govern continuous evolution, or only allowing a specific class of sets (e.g. polytopes) to act as mode invariants for the states of the automaton. We should note that in this sense the class of systems considered in this paper is very broad because it allows for non-linear continuous dynamics and for arbitrary semi-algebraic sets to act as mode invariants and transition guards.

It is our hope our techniques will in future be applied to modelling and verification of properties in systems with engineering applications that employ variable structure control.

[5]The implementation is available from goo.gl/8ReQxB.

We stress, however, that many important questions remain unresolved. For instance, the difficult task of categorizing and classifying the possible kinds of operational models (beyond the three presented) remains to be addressed. Interesting questions as to which of the many possible types of operational semantics for PWS that can be obtained through using techniques described in this paper are "physically meaningful" (and for what phenomena) present many intriguing avenues for future research.

6.1 Related Work

Lygeros et al. studied existence and uniqueness of executions of hybrid automata in [26], giving conditions under which hybrid automata are deterministic and non-blocking. We note that there are important differences in definitions, e.g. the use of semi-open time intervals in [26], such as in

$$\mathrm{Out}(q_i) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \forall\, \varepsilon.\ \exists\, t \in [0, \varepsilon).\ \varphi_t(\boldsymbol{x}) \notin M_i\},$$

where $M_i = \mathrm{Dom}(q_i)$. This differs from definitions used in this paper, e.g.

$$\neg\mathrm{In}_{\mathfrak{f}_i}(M_i) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \forall\, \varepsilon.\ \exists\, t \in (0, \varepsilon).\ \varphi_t(\boldsymbol{x}) \notin M_i\}.$$

REMARK 4. *Similar notions also exist in the ODE literature, e.g. "ingress" and "egress" sets used to state and prove the Ważewski principle ([21, p. 282],[7]).*

The work in [26] is also similar in using Lie derivatives of functions to reason about the transition behaviour; however, the authors consider a special class of hybrid automata in which mode invariants can be characterized by sub-level sets of analytic functions, i.e. $\sigma(\boldsymbol{x}) \geq 0$. The same restriction was used in the work of Johansson et al. [23] and already rules out systems in which mode invariants are given by polytopes. We work under much more general assumptions where the mode invariants are semi-algebraic sets and work with their representations directly. Further investigations of existence and uniqueness of executions of hybrid automata were reported in later work by Lygeros et al. [27].

7. CONCLUSION

In this paper we presented a methodology for automatically synthesizing hybrid automata from descriptions of piecewise-smooth polynomial systems, i.e. systems of discontinuous ODEs that are polynomial on disjoint semi-algebraic sets forming a partition of the state space. The hybrid automata thus obtained provide operational models of piecewise-smooth systems, which can behave in different ways, depending on certain choices in formulating the conditions on the transition guards. We have described in Sections 4.2, 4.3 three alternative choices that can be exercised in this regard, and which can be thought of as giving different operational meaning (i.e. semantics) to the piecewise-smooth systems. Many more choices are possible and the task of studying and classifying these possibilities presents a very interesting direction for further research.

One of our main aims in this paper was to present a case as to why it is not meaningful to speak of "a hybrid automaton model" of a given piecewise-smooth system without a precise description of how the said hybrid automaton model was created. We argue that a synthesis algorithm, such as that presented in Section 4.2, is needed in order to provide this description.

We believe that correct modelling of piecewise-smooth systems is a problem that is of more than just theoretical interest, since systems of this type occur frequently in control engineering (often in the context of autonomous switching or sliding mode controllers). Their representation as differential equations active inside certain designated regions is deceptively simple and great care needs to be taken when extracting operational models from these simple representations. Our work addressed some of the fundamental difficulties inherent in this task.

8. REFERENCES

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.

[2] A. D. Ames, A. Abate, and S. Sastry. Sufficient conditions for the existence of zeno behavior. In *CDC-ECC'05*, pages 696–701. IEEE, 2005.

[3] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, second edition, 2006.

[4] B. F. Caviness and J. R. Johnson. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer, 1998.

[5] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *CAV*, pages 258–263, 2013.

[6] G. E. Collins. Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

[7] C. C. Conley. *Isolated invariant sets and the Morse index*. Conference Board of the Mathematical Sciences, 1978.

[8] J. Cortés. Discontinuous dynamical systems: A tutorial on solutions, non-smooth analysis and stability. *IEEE Control Systems*, 28(3):36–73, 2008.

[9] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2010.

[10] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1):29–35, 1988.

[11] J. M. Davoren and A. Nerode. Logics for hybrid systems. *Proc. IEEE*, 88(7):985–1010, 2000.

[12] M. Egerstedt. Behavior based robotics using hybrid automata. In *HSCC*, pages 103–116. Springer, 2000.

[13] G. Frehse. An introduction to hybrid automata, numerical simulation and reachability analysis. In *Formal Modeling and Verification of Cyber-Physical Systems*, pages 50–81. Springer, 2015.

[14] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *CAV*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.

[15] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In *CADE*, LNCS. Springer, 2015.

[16] K. Ghorbal and A. Platzer. Characterizing algebraic invariants by differential radical invariants. In *TACAS*, volume 8413, pages 279–294. Springer, 2014.

[17] K. Ghorbal, A. Sogokon, and A. Platzer. A hierarchy of proof rules for checking positive invariance of algebraic and semi-algebraic sets. *Computer Languages, Systems & Structures*, 2015.

[18] R. Goebel, R. G. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems*, 29(2):28–93, 2009.

[19] O. Hájek. Discontinuous differential equations, I. *J. Diff. Eq.*, 32(2):149 – 170, 1979.

[20] J. K. Hale and J. P. LaSalle. Differential equations: Linearity vs. nonlinearity. *SIAM Review*, 5(3):249–272, July 1963.

[21] P. Hartman. *Ordinary Differential Equations*. John Wiley & Sons, Inc., New York, 1964.

[22] T. Henzinger. The theory of hybrid automata. In *Proc. IEEE Symposium on Logic in Computer Science*, page 278. IEEE Computer Society, 1996.

[23] K. H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of zeno hybrid automata. *Systems & Control Letters*, 38(3):141–150, 1999.

[24] S. Kong, S. Gao, W. Chen, and E. M. Clarke. dreach: δ-reachability analysis for hybrid systems. In *TACAS*, volume 9035 of *LNCS*, pages 200–205. Springer, 2015.

[25] J. Liu, N. Zhan, and H. Zhao. Computing semi-algebraic invariants for polynomial dynamical systems. In *EMSOFT*, pages 97–106. ACM, 2011.

[26] J. Lygeros, K. H. Johansson, S. Sastry, and M. Egerstedt. On the existence of executions of hybrid automata. In *CDC*, pages 2249–2254. IEEE, 1999.

[27] J. Lygeros, K. H. Johansson, S. N. Simić, J. Zhang, and S. S. Sastry. Dynamical properties of hybrid automata. *IEEE TAC*, 48(1):2–17, 2003.

[28] B. Mishra. *Algorithmic Algebra*. Texts and Monographs in Computer Science. Springer, 1993.

[29] P. J. Mosterman. Mode transition behavior in hybrid dynamc systems. In S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, editors, *Proc. of the 2003 Winter Simulation Conference*, pages 623–631, Dec. 2003.

[30] P. J. Mosterman and G. Biswas. A theory of discontinuities in physical system models. *Journal of the Franklin Institute*, 335(3):401–439, 1998.

[31] P. J. Mosterman, F. Zhao, G. Biswas, et al. An ontology for transitions in physical dynamic systems. In *AAAI/IAAI*, pages 219–224, 1998.

[32] E. Navarro-López and R. Carter. Hybrid automata: An insight into the discrete abstraction of discontinuous systems. *International Journal of Systems Science*, 42(11):1883–1898, 2011.

[33] D. Novikov and S. Yakovenko. Trajectories of polynomial vector fields and ascending chains of polynomial ideals. In *Annales de l'institut Fourier*, volume 49, pages 563–609, 1999.

[34] A. Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reasoning*, 41(2):143–189, 2008.

[35] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.

[36] R. Sanfelice, D. Copp, and P. Nanez. A toolbox for simulation of hybrid systems in Matlab/Simulink: Hybrid Equations (HyEQ) Toolbox. In *HSCC*, pages 101–106. ACM, 2013.

[37] R. G. Sanfelice, R. Goebel, and A. R. Teel. Generalized solutions to hybrid dynamical systems. *ESAIM: Control, Optimisation and Calculus of Variations*, 14:699–724, 10 2008.

[38] A. Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, pages 365–374, 1954.

[39] A. Tarski. A decision method for elementary algebra and geometry. Technical Report R-109, RAND, 1951.

[40] G. Teschl. *Ordinary Differential Equations and Dynamical Systems*, volume 140 of *Graduate Studies in Mathematics*. American Mathematical Society, 2012.

[41] V. I. Utkin. Sliding modes in control and optimization, 2013.

[42] A. J. Van Der Schaft and J. M. Schumacher. *An introduction to hybrid dynamical systems*, volume 251. Springer, 2000.

[43] S. Wang, N. Zhan, and L. Zou. An improved HHL prover: An interactive theorem prover for hybrid systems. In *ICFEM*, volume 9407 of *Lecture Notes in Computer Science*, pages 382–399. Springer, 2015.

[44] H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, 11(2):161–167, Apr 1966.

[45] F. Zhao and V. I. Utkin. Adaptive simulation and control of variable-structure control systems in sliding regimes. *Automatica*, 32(7):1037 – 1042, 1996.

APPENDIX

A. COMPUTING "IN SETS" EXACTLY

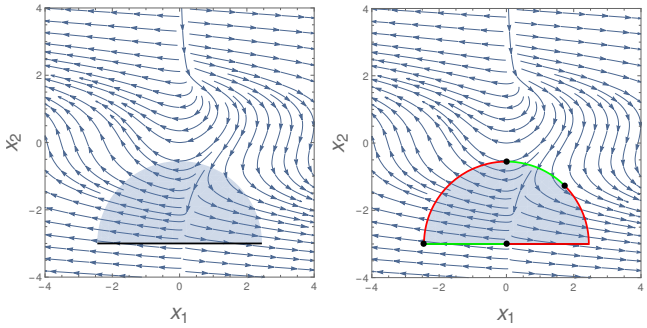
To give an idea of how $\mathrm{In}_{\mathfrak{f}}(S)$ is computed exactly, consider a set S which is given by $p \leq 0$, where p is some polynomial function in the state variables $x_1, \ldots, x_n$ with real coefficients. Firstly, note that each point $\boldsymbol{x}$ in the interior of S, i.e. satisfying $p < 0$ necessarily lies inside $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ because motion within the interior is always possible within some open neighbourhood. The set $p < 0$ thus provides the first under-approximation of the set $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$. We now refine this under-approximation by adding some of the states satisfying $p = 0$, for which a *sufficient* (but not necessary) condition for membership in $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ is that of satisfying the inequality $\mathfrak{L}_{\mathfrak{f}}(p) < 0$. This is intuitive because the rate of change of p at such a state is negative and therefore the system will immediately evolve into the set satisfying $p < 0$. However, for states satisfying $p = 0$ and $\mathfrak{L}_{\mathfrak{f}}(p) = 0$, one needs to check that the second-order Lie derivative is negative (i.e. $\mathfrak{L}^2_{\mathfrak{f}}(p) < 0$) in order to conclude their membership in $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$, and so on for higher-order Lie derivatives. Intuitively, these cases correspond to situations where "the velocity is zero, but the acceleration is negative", etc., which likewise ensures that the system cannot evolve into a state satisfying $p > 0$ (i.e. the complement of $p \leq 0$) immediately afterwards. The set $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ is then constructed as follows:

$$\begin{aligned}\mathrm{In}_{\mathfrak{f}}(p \leq 0) \equiv\ & p < 0 \\ & \vee\ (p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) < 0) \\ & \vee\ (p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) = 0 \wedge \mathfrak{L}^2_{\mathfrak{f}}(p) < 0) \\ & \vdots \\ & \vee\ (p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) = 0 \wedge \cdots \wedge \mathfrak{L}^k_{\mathfrak{f}}(p) \leq 0)\end{aligned}$$

The fact that the number k is finite and can be computed is a consequence of Hilbert's basis theorem and the ascending chain property of Noetherian rings (see e.g. [28, Sec. 2.3.2]). These fundamental results guarantee that one is always able to find a $k \in \mathbb{N}$ such that the ideal membership $\mathfrak{L}^K_{\mathfrak{f}}(p) \in \langle p, \mathfrak{L}_{\mathfrak{f}}(p), \ldots, \mathfrak{L}^k_{\mathfrak{f}}(p)\rangle$ [6] holds for all $K \geq k$. This property is equivalent to the statement that for each $K \geq k$ the following equality holds: $\mathfrak{L}^K_{\mathfrak{f}}(p) = \alpha_0 p + \alpha_1 \mathfrak{L}_{\mathfrak{f}}(p) + \cdots + \alpha_k \mathfrak{L}^k_{\mathfrak{f}}(p)$, where the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_k$ are some polynomials in the ring $\mathbb{R}[x_1, \ldots, x_n]$. Thus, whenever $p = \mathfrak{L}_{\mathfrak{f}}(p) = \cdots = \mathfrak{L}^k_{\mathfrak{f}}(p) = 0$ holds, one necessarily has $\mathfrak{L}^K_{\mathfrak{f}}(p) = 0$ for all $K \geq 0$, and thus it is impossible to grow the under-approximation of $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ by adding any more disjuncts of the form $p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) = 0 \wedge \cdots \wedge \mathfrak{L}^k_{\mathfrak{f}}(p) = 0 \wedge \cdots \wedge \mathfrak{L}^K_{\mathfrak{f}}(p) < 0$ for any $K > k$ and the construction is therefore complete. In practice, the number k is computed using Gröbner bases (e.g. see [9, Chap. 2]).

B. ENTER, EXIT AND BOUNCE SETS

Consider a semi-algebraic set described by the formula $S \equiv x_1^2 + (x_2 + 3)^2 < 6 \wedge -3 \leq x_2$ and let the dynamics of the system, $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$, be given by the system of polynomial ODEs: $\dot{x}_1 = x_1 x_2^2 - 1, \dot{x}_2 = -x_1$. Fig. 9a shows the set S



(a) Semi-algebraic set $S \subset \mathbb{R}^2$ (b) $\mathrm{Enter}_{\mathfrak{f}}(S)$ and $\mathrm{Exit}_{\mathfrak{f}}(S)$

Figure 9: Semi-algebraic set, along with its "entry" states (in green) and "exit" states (in red).

along with some of the trajectories of the system. The set of "entering states", given by

$$\begin{aligned}&\mathrm{Enter}_{\mathfrak{f}}(S) = \big(x_2 + 3 = 0 \wedge x_1 < 0 \wedge x_1^2 + x_2^2 + 6x_2 + 3 < 0\big) \\ &\vee\ (x_2 + 3 > 0 \wedge x_1^2 + x_2^2 + 6x_2 + 3 = 0 \wedge x_1^2 x_2^2 < x_1\,(x_2 + 4)),\end{aligned}$$

[6] i.e. $\mathfrak{L}^K_{\mathfrak{f}}(p)$ is in the ideal generated by the finite set of polynomials $\{p, \mathfrak{L}_{\mathfrak{f}}(p), \ldots, \mathfrak{L}^k_{\mathfrak{f}}(p)\}$

is shown in green in Fig. 9b, and

$$\mathrm{Exit}_{\mathfrak{f}}(S) = \Big(0 < x_1 \leq \sqrt{6} \wedge x_2 + 3 = 0\Big) \vee \big(x_2 + 3 > 0 \\ \wedge\, x_1^2 + x_2\,(x_2 + 6) + 3 = 0 \wedge x_1^2 x_2^2 > x_1\,(x_2 + 4)\,\big)$$

is shown in red. Note that these two sets need not necessarily include all the points on the boundary of S. The black points in Fig. 9b represent states on the boundary which are neither in $\mathrm{Enter}_{\mathfrak{f}}(S)$ nor $\mathrm{Exit}_{\mathfrak{f}}(S)$. In particular, $\mathrm{Bounce}_{\mathfrak{f}}(S)$ includes the point at the centre of the semi-circle, i.e. $x_1 = 0 \wedge x_2 = -3$, whereas the remaining three points in the figure belong to $\mathrm{Bounce}_{\mathfrak{f}}(\neg S)$.