Semantics and computation of the evolution of hybrid systems with ariadne

SIAM J. CONTROL OPTIM.
Vol. 49, No. 2, pp. 890–925

SEMANTICS AND COMPUTABILITY OF THE EVOLUTION OF HYBRID SYSTEMS*

PIETER COLLINS†

Abstract. In this paper we consider the semantics for the evolution of hybrid systems, and the computability of the evolution with respect to these semantics. We show that with respect to *lower* semantics, the finite-time reachable sets are lower-semicomputable, and with respect to *upper* semantics, the finite-time reachable sets are upper-semicomputable. We use the framework of type-two Turing computability theory and computable analysis, which deal with obtaining approximation results with guaranteed error bounds from approximate data. We show that, in general, we cannot find a semantics for which the evolution is both lower- and upper-semicomputable, unless the system is free from *tangential* and *corner* contact with the guard sets. We highlight the main points of the theory with simple examples illustrating the subtleties involved.

Key words. computable analysis, hybrid automaton, reachable set

AMS subject classifications. 93B03, 93-04, 68Q17, 93B40

DOI. 10.1137/080716955

1. Introduction.

Hybrid systems are dynamic systems in which the evolution has both discrete-time (instantaneous) and continuous-time elements. Hybrid models are becoming increasingly prevalent in industry, and there is a need for tools which can perform reliable simulation and verification analysis of hybrid automata. The interplay between the continuous and discrete dynamics causes difficulties in the analysis of hybrid automata which do not occur in discrete- or continuous-time systems, and which lend hybrid automata a distinctive character.

Many questions about the behavior of a hybrid automaton can be framed in the context of reachable sets and the reachability relation. In particular, for systems where the input represents a control signal, the reachable set from a given point represents the set which can be attained by some open-loop control. It has long been known that the reachability relation for hybrid automata is undecidable [2], except for the class of timed automata (and slight generalizations), for which reachable sets can be computed exactly [44]. Rather than considering decidability of the reachability relation, we consider the related question of computability of the reachable set. For more complicated systems, symbolic computations are infeasible and approximate numerical computations are required. This motivates the study of what is possible to compute using approximations to the exact problem data if it is only necessary to compute the result approximately.

In this paper, we base our computability results on the theory of computable analysis of Weihrauch [49] and coworkers, which is equivalent to that of [28] based on Scott domain theory and of [36] based on oracle machines. All computations are performed using ordinary Turing machines, and hence can be implemented using existing digital computers. (This is unlike the real-RAM theory of [11], which cannot be effectively implemented.) In order to allow computation on uncountable sets, we

*Received by the editors February 28, 2008; accepted for publication (in revised form) February 15, 2011; published electronically April 27, 2011. This work was supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) Vidi grant 639.032.408.
http://www.siam.org/journals/sicon/49-2/71695.html
†Centrum voor Wiskunde en Informatica, Postbus 94079, 1090 GB Amsterdam, The Netherlands (pieter.collins@cwi.nl).

allow computations to run indefinitely, writing an output stream which represents successively more accurate approximations to the result. We say a quantity is *computable* if it can be computed to arbitrary (metric) accuracy and *semicomputable* if it is possible to compute convergent approximations from above or below. The theory is closely related to general topology (see [41] for a good introduction); in particular, only continuous operators can be computable. We note that uncomputability in the framework of computable analysis does not necessarily imply uncomputability in some algebraic framework in which the objects of interest can be specified exactly.

We will see that computability of the reachable set is strongly related to topological properties of the invariants and guard sets, and to continuity properties of the continuous and discrete dynamics. In order to separate technical issues relating to the solution of differential equations and differential inclusions from the intrinsic difficulties of hybrid automata, we first compute the continuous dynamics as a (multi-)flow from the defining differential relation. Upper-semicontinuity of solutions of hybrid automata has been considered in [4, 30]. Lower-semicontinuity of the solutions of Lipschitz differential inclusions and hybrid automata has been studied in [15, 16]. Viability in hybrid systems, which is related to lower-semicontinuity of the evolution, is studies in [37]. Existence and continuity of solutions has been studied in [40].

Unlike purely discrete- or continuous-time systems, for which there is a well-defined notion of solution, for hybrid automata we need to use different solution concepts for computing lower- and upper-approximations to the reachable set. The upper solution concept may necessarily impose nondeterministic (multivalued) solutions to an otherwise deterministic system, whereas lower solution concepts may impose blocking. Reliable simulation imposes the need to consider multiple possible evolutions, each of a qualitatively different nature.

Computability of the evolution is closely related with decidability of certain properties of the dynamics. It follows immediately from basic results of computable analysis that the evolution is upper-semicomputable if, and only if, every finite-time safety problem for an open set of safe states is verifiable. Conversely, the evolution being lower-semicomputable is equivalent to verifiability of every control-to-target problem for an open set of target states.

The results in this paper extend those of [23] and provide full proofs. Similar results on the computability of reachable sets for discrete-time systems were given in [19]. The computability of solutions of hybrid automata was also considered in [26] in the context of the Scott-domain approach. The results are weaker than those presented here, since the assumptions prohibit the discontinuities described in sections 2 and 4. An alternative approach [10] to semantics of hybrid systems uses nonstandard analysis to give a semantics for use in simulation. The emphasis is on providing a time-discretization for solving differential equations and scheduling crossings. The resulting semantics is not computable in the sense used here, since issues regarding the representation of points in the state space are not considered.

There are other tools available for computing reachable sets of hybrid automata, such as d/dt [1], Hy(per)tech [32], VeriShift [12], Checkmate [5], and Phaver [27]. However, these tools are mainly restricted to systems with affine dynamics and guard sets (apart from CheckMate, which allows nonlinear dynamics) and can only compute over-approximations to the reachable set. Computation of the solution of discrete-time hybrid systems using the software package GAIO [25] has been considered in [31], but the analysis does not straightforwardly extend to the continuous-time case. To remedy this situation, a software tool, Ariadne [6], is being developed to implement the computable operations of this paper.

The paper is structured as follows. In section 2 we give an introduction to hybrid automata theory, introducing the class of hybrid automata and standard solution concepts. In section 3 we give an introduction to computability theory as applied to topology and analysis. In section 4 we present the ways in which the evolution of a hybrid automaton may fail to be continuous. In section 5 we present the main theorems on semicomputability of the evolution. In section 6 we present some modeling frameworks for hybrid automata and discuss reliable simulation and implementation issues. Finally, in section 7 we state some conclusions and give directions for further research.

2. Hybrid systems. Hybrid systems are dynamic systems comprising both continuous- and discrete-time behavior. See [48] for an general introduction to hybrid systems and [42] for a discussion of hybrid models.

2.1. Continuous- and hybrid-time systems. One of the most important results in the theory of continuous-time systems is the existence and uniqueness result for Lipschitz differential equations $\dot{x} = f(x)$. Further, if $\Phi : X \times \mathbb{R} \to X$ is the solution flow of the differential equation, then Φ is continuous, and can be effectively approximated, in the sense that given the function f, the initial condition x_0, and the integration time t, we can compute $\Phi(x_0, t)$ arbitrarily accurately on a digital computer. In many situations, the data f, x_0 and t may not be known exactly. However, even in this case, given a sufficiently accurate description of f, x_0, and t, we can still compute the evolution Φ.

Compare the situation for differential equations with that for deterministic hybrid automata. If we denote the solution of a hybrid automaton with initial condition x_0 at time t by $\psi(x_0, t)$, we see that there are a number of situations in which the ψ may vary discontinuously in x_0 and t (Figure 1).

Time discontinuity at discrete transitions. Whenever the state of the system is reset during a discrete transition, there is a discontinuity in the time evolution. If this occurs for the initial condition x_0 at time t, then there is a discontinuity in dependence of the current state on x_0 and t.

The standard way of handling time discontinuities at discrete transitions, introduced in [39], is to consider hybrid trajectories as continuous functions of a disconnected domain, such as a subset of $\mathbb{R}^+ \times \mathbb{Z}^+$. In this way, time discontinuities can be regularized, though there are still discontinuities in reachable sets.

Spatial discontinuity at tangencies and corner collisions. If the continuous evolution touches a guard set tangentially near x, then some points near x undergo a discrete transition, whereas other points undergo further continuous evolution. The same phenomenon may occur if the continuous evolution touches a corner of a guard set.

Spatial discontinuity at switching boundaries. If x lies on the boundary of two guard sets, then some points near x undergo one transition, and others undergo the other transition.

Spacial discontinuity at instantaneous transitions. Suppose that after a discrete transition, the state x lies on the boundary of the switching set. Then some points near x immediately undergo a second transition, whereas other points may flow away from the guard set and do not undergo the transition.

All these situations may occur generically, which means that they persist under small perturbations of the parameters defining the system. From the viewpoint of dynamics, it is these discontinuities which distinguish hybrid automata from purely discrete-time or continuous-time systems. In many cases, the spatial discontinuities

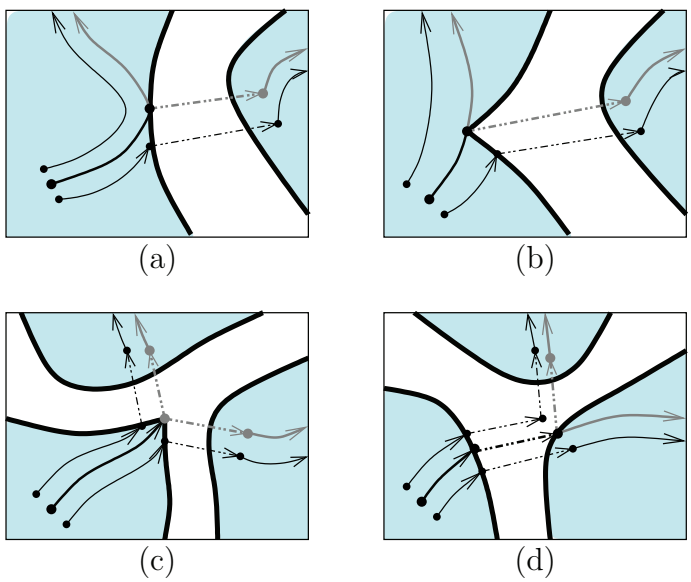

FIG. 1. *Spatial discontinuities.* (a) *At a tangency;* (b) *at a corner collision;* (c) *at a switching boundary; and* (d) *at an instantaneous transition.*

occur only for a "small" (measure zero) set of initial conditions, and therefore might not be considered of physical interest. However, spatial discontinuities may still occur on a (locally) dense set of initial conditions. If the exact solution passes very near a discontinuity point at time t, then the presence of even a small numerical error may cause the computed solution after time t to differ drastically from the exact solution. As we shall see, it is important to handle these situations correctly in the development of a sound numerical theory of hybrid automata.

2.2. Autonomous hybrid automata. The most commonly used framework for describing hybrid automaton models is that of *hybrid automata*. Informally, the hybrid automaton model is based on a discrete automaton with *discrete states* and *discrete events*, with continuous dynamics associated to each discrete state, with *guard* conditions determining the discrete events which occur, and with *reset maps* determining the way the continuous part of the state changes during a discrete event.

We now give a fairly standard formal definition of a hybrid automaton. Write $f : X \dashrightarrow Y$ to indicate that f is a partial function from X to Y, and write $F : X \rightrightarrows Y$ for a multivalued function.

DEFINITION 2.1 (hybrid automaton). *A* hybrid automaton *is a tuple*

$$H = (Q, E, \gamma, \{X_q, I_q, F_q, P_q \mid q \in Q\}, \{R_{q,e}, G_{q,e} \mid (q,e) \in \operatorname{dom}\gamma\}, E_U), \tag{1}$$

where

1. Q *is a finite set of* discrete states,
2. E *is a finite set of* discrete events,
3. $\gamma : Q \times E \dashrightarrow Q$ *is a partial* discrete transition function *with domain* $\operatorname{dom}\gamma$; *for each* $q \in Q$,
4. X_q *is a differentiable manifold giving the* continuous state space *for discrete state* q,

5. $I_q \subset X_q$ *is the* invariant *for the continuous state,*
6. $F_q : X_q \rightrightarrows TX_q$ *is a differential inclusion giving the* continuous dynamics*,*
7. $P_q \subset X_q$ *is the* progress *predicate for the continuous dynamics;*

for each $(q, e) \in \operatorname{dom} \gamma$*,*

7. $R_{q,e} : X_q \rightrightarrows X_{\gamma(q,e)}$ *is a multivalued* reset *map,*
8. $G_{q,e} \subset X_q$ *is a* guard *set;*

and

9. $E_U \subset E$ *is a set of* urgent *events.*

We use differential inclusions and allow multivalued resets to permit the modeling of control or disturbance inputs.

When considering the dynamical behavior of a hybrid automaton, the distinction between the discrete and continuous part of the state is often irrelevant and complicates the notation. We define the *state space* X of a hybrid automaton H by

$$X = \textstyle\bigcup_{q \in Q} \{q\} \times X_q. \tag{2}$$

Using this equation, we write

$$\begin{aligned} I &= \textstyle\bigcup_{q \in Q} \big(\{q\} \times I_q\big); \\ F(q, x) &= F_q(x); \\ P &= \textstyle\bigcup_{q \in Q} \big(\{q\} \times P_q\big); \\ R_e(q, x) &= (\gamma(q, e), R_{q,e}(x)); \\ G_e &= \textstyle\bigcup_{q \in Q} \big(\{q\} \times G_{q,e}\big). \end{aligned} \tag{3}$$

Note that during continuous evolution, the discrete state q is constant; implicitly $\dot{q} = 0$. The resulting system is described by the tuple

$$(E, X, I, F, P, \{R_e, G_e \mid e \in E\}, E_U), \tag{4}$$

or simply,

$$(E, X, I, F, P, R_e, G_e, E_U). \tag{5}$$

We henceforth use the form (4) when developing the theory and use (1) in examples.

We will sometimes restrict our attention to subclasses of hybrid automata. If there is a single event e which is not urgent, then the hybrid automaton is described by a tuple

$$H = (X, I, F, P, R, G). \tag{6}$$

If the invariant I is empty, we write

$$H = (X, F, P, R, G). \tag{7}$$

If the invariant I and progress predicate P equal the whole space X, and there is a single event e which is urgent, then the hybrid automaton is described by a tuple

$$H = (X, F, R, G) \tag{8}$$

and is an *impulse differential inclusion* as defined in [4]. If, additionally, the dynamic F is a Lipschitz differential equation $\dot{x} = f(x)$, the reset R is a single-valued map $x' = r(x)$, and the guard G is $\{x \in X \mid g(x) \geqslant 0\}$, then the resulting system

$$H = (X, f, r, g) \tag{9}$$

is a *deterministic hybrid automaton.*

2.3. Evolution of a hybrid automaton. To represent a trajectory of a hybrid automaton, we need to take into account the possibility that more than one discrete event occurs at a given time. To capture the intermediate states, we use the following definition of *hybrid trajectory* [18, 29], which is based on the work of [39].

DEFINITION 2.2 (hybrid trajectory). *Let $(t_i)_{i<\infty}$ be a (not necessarily strictly) increasing sequence in $\mathbb{R}^+ \cup \{\infty\}$ with $t_0 = 0$. Then the t_i define a* hybrid time domain *$\mathcal{T} \subset \mathbb{R}^+ \times \mathbb{Z}^+$ by*

$$\mathcal{T} = \{(t,n) \in \mathbb{R}^+ \times \mathbb{Z}^+ \mid t_n \leqslant t \leqslant t_{n+1}\} = \bigcup_{n=0}^{\infty} [t_n, t_{n+1}] \times \{n\}.$$

A hybrid trajectory *is a continuous function $\xi : \mathcal{T} \to X$ for some hybrid time domain $\mathcal{T} = \mathrm{dom}(\xi)$. The trajectory ξ is* Zeno *if $\lim_{n\to\infty} t_n < \infty$, and it has* finitely many events *if $t_n = \infty$ for some n. We let $\Xi(X)$ be the set of all hybrid trajectories in the state space X.*

Note that the continuity condition on ξ is equivalent to requiring that for all n, the function $t \mapsto \xi(t,n)$ is continuous on $[t_n, t_{n+1}]$.

The natural topology on the space of hybrid trajectories is a compact-open topology with an additional reparametrization to account for the fact that "nearby" trajectories may have slightly different event times. In other words, trajectories are considered "close" if, after a small reparametrization of the time domain, they are uniformly close on a large compact set. This idea can be seen in the Skorohod topology of [16] and is made explicit in the *compact-open hybrid Skorohod topology* of [18] and the three-parameter uniformity of [24]. It can be easily seen, though we shall not prove, that this is the finest topology under which the solutions of a switching system vary continuously with respect to the initial state, the system parameters, and the switching times.

In this article, we instead use the *graph topology* introduced in [29], which is equivalent, by results of [24], to the compact-open hybrid Skorohod topology but technically easier to handle.

DEFINITION 2.3 (hybrid trajectory space). *The* graph *of a hybrid trajectory $\xi : \mathcal{T} \to X$ is the set*

$$\mathrm{graph}(\xi) = \{(t,n,x) \mid (t,n) \in \mathrm{dom}(\xi) \text{ and } x = \xi(t,n)\}. \tag{10}$$

The topology on the space of hybrid trajectories is the Fell topology *(see* [9]*) on closed sets. This is the topology defined by the basic open sets $\{A \mid A \cap U \neq \emptyset\}$ and $\{A \mid A \cap C = \emptyset\}$, where U is open and C compact in $\mathbb{R}^+ \times \mathbb{Z}^+ \times X$.*

DEFINITION 2.4 (executions of a hybrid automaton). *An* execution *of the hybrid automaton $H = (E, X, I, F, P, R_e, G_e, E_U)$ with the* standard semantics *is a hybrid trajectory $\xi : \mathcal{T} \to X$ such that there exist events $e_1, e_2, \ldots$ with*

(SS1) $\xi(t,n) \in I$ whenever $t \in [t_n, t_{n+1}]$,
(SS2) $\dot{\xi}(t,n) \in F(\xi(t,x))$ for almost every $t \in [t_n, t_{n+1}]$,
(SS3) $\xi(t_n, n) \in R_{e_n}(\xi(t_n, n-1))$ for all n,
(SS4) $\xi(t,n) \in P$ whenever $t \in [t_n, t_{n+1}[$,
(SS5) $\xi(t_n, n-1) \in G_{e_n}$, and
(SS6) $\xi(t,n) \notin G_u$ whenever $t \in [t_n, t_{n+1}[$ and $u \in E_U$.

In other words, the evolution of the system proceeds via the continuous dynamics $\dot{x} \in F(x)$ until an event e occurs, at which point the state jumps to $x' \in R_e(x)$. The state must satisfy the invariant $x \in I$ at all times, including immediately before

and immediately after a transition. The event e may only occur if $x \in G_e$, and an event must occur at x if either $x \notin P$ or $x \in G_u$ for some urgent event u. Note that we do not insist that the urgent event u itself occur; this is unreasonable given the possibility that two urgent events u_1 and u_2 become active at exactly the same time.

The definitions of invariant and progress predicate used here are equivalent to those used in the Compositional Interchange Format (CIF) for hybrid automata [47]. The difference is that an invariant must hold at all states along a trajectory, whereas a progress predicate need not hold if a discrete transition occurs immediately. In terms of the standard semantics, there is no difference between an urgent action with guard G_u, and a nonurgent action with guard G_e and a progress condition $X \setminus G_e$. However, the difference is important in the definition of crossing semantics given in section 5.3.

If the event e is urgent, then the guard set G_e should always be taken as a closed set. If G_e were open, then there would be no possibility for e to be activated during a continuous evolution, for at the activation time t_{n+1}, we would already have $\xi(t,n) \in G_e$ for some $t < t_{n+1}$, contradicting condition (SS6).

Notice that we have two types of restrictions on the continuous dynamics, namely, those given by the invariants and those given by the guards of urgent events. We also have two types of discrete dynamics, namely, those given by the urgent events and those given by the nonurgent events. When computing upper-approximations to the evolution, we will be able to effectively consider an urgent action e with guard set G_e as a nonurgent action, but with an extra invariant $\mathrm{cl}(X \setminus G_e)$. Computing lower-approximations to the evolution is more challenging, since we have to treat urgent transitions in a special way.

We have the following simplifications in the description of a hybrid automaton. The proofs are straightforward from the definitions.

LEMMA 2.5. *Let $H = (E, X, I, F, P, R_e, G_e, E_U)$ be a hybrid automaton.*

1. *The executions of H are the same as those of the hybrid automaton $(X, I, F, P \setminus \bigcup_{u \in U} G_u, R_e, G_e, \emptyset)$.*
2. *If $E_U = \emptyset$, then the executions of H are the same as those of the single-event hybrid automaton (X, I, F, P, R, G) with $G = \bigcup_{e \in E} G_e$ and $R(x) = \bigcup\{R_e(x) \mid x \in G_e\}$.*
3. *If H is a single-event hybrid automaton, then executions of H with initial state $x_0 \in I$ are the same as those of the single-event hybrid automaton $(X, \emptyset, F, P \cap I, R|^I, G)$, where $R|^I$ is defined by $R|^I(x) = R(x) \cap I$.*

We are often not concerned with computing the full set of trajectories, except in the points which can be reached by evolution up to a certain time.

DEFINITION 2.6. *Given a hybrid automaton H, the* evolution operator $\Psi^H : X \times \mathbb{R}^+ \rightrightarrows X$ *is defined by*

$$\Psi^H(x,t) = \{y \in X \mid \exists \text{ execution } \xi \text{ of } H,\ n \in \mathbb{Z}^+ \text{ s.t. } \xi(0,0) = x \text{ and } \xi(t,n) = y\}. \tag{11}$$

In other words, the evolution operator Ψ^H of a hybrid automaton H is the function taking a point x and time t to the set of all points which can be reached from x by an execution of duration t. The *reachability operator* is defined to be the map

$$(X_0, T) \mapsto \Psi^H(X_0, T) = \bigcup\{\Psi^H(x_0, t) \mid x_0 \in X_0 \wedge t \in T\}, \tag{12}$$

where $X_0 \subset X$ and $T \subset \mathbb{R}^+$.

In this paper, we will consider the following main problem.

PROBLEM 2.7. *Compute the evolution $\Psi^H(x,t)$ of a hybrid automaton H.*

Remark 2.8. It would also be possible to consider the *solution operator* Θ^H defined by

$$\Theta^H(x) = \{\xi \mid \xi \text{ is an execution of } H \text{ and } \xi(0,0) = x\}, \tag{13}$$

though we do not do so to avoid additional technical complications of dealing with the hybrid trajectory space.

3. Computable analysis.

3.1. Overview of computability theory. We have seen that hybrid automata may exhibit discontinuities in the evolution, and intuitively we expect that the presence of discontinuities will cause difficulties in computing the system evolution, even to the extent that it is impossible to compute the evolution to arbitrary accuracy. However, to actually prove that a certain computational task is impossible, we need a formal theory of computation, which requires specifying a computational model, and also the input and output data that the computational model works with. We compare this motivation with Turing's motivation for introducing his computing machines, which was to prove the impossibility of an algorithmic solution of Hilbert's Entscheidungsproblem. Since we are interested in algorithmic solutions to problems concerning hybrid automata, if the general form of our problem turns out to be unsolvable, we want to know which problem instances are solvable, or find related problems which are completely solvable.

In this paper, we use the theory of *computable analysis* as developed by Weihrauch [49] and coworkers. In this theory, computation is performed by ordinary Turing machines acting on streams of data. The data stream encodes a sequence of approximations to some quantity, such as a subset of the state space, or a function describing a system. A function or operator is computable if, given a data stream encoding a sequence of approximations converging to the input, it is possible to calculate a data stream encoding a sequence of approximations converging to the output. In practice, finite computations can be obtained by terminating whenever a given accuracy criterion is met. However, it is theoretically very convenient to consider the computations to be infinite, since we can talk about computing the mathematical objects themselves. Two encodings or *representations* of the same class of mathematical object are equivalent if each can be transformed into the other by a Turing machine; this makes it possible to relate results on representations which are easy to work with theoretically to representations which are efficient to work with in implementations.

The representations used in computable analysis are related to a topology on the set of objects under consideration, and so give a clean link between approximability, continuity, and formal computability. The fundamental theorem is that only continuous functions with respect to a given topology can be computable with respect to representations based on that topology. Hence if we can prove that a function is discontinuous, then it is uncomputable. For "naturally" defined functions the converse is typically also true, that is, continuous functions are computable. It is worth emphasizing that a function which is uncomputable with respect to one representation may be computable with respect to a representation based on a different topology. This corresponds to giving more information in the input or requiring less information in the output. We shall see later that the use of the correct topology/representation is vital when considering computability for hybrid automata.

Since objects are described by sequences of symbols, we can represent sets of continuum cardinality. This includes points in Euclidean space; open, closed, and

compact subsets; continuous functions; and semicontinuous multivalued functions, but not arbitrary subsets of space or arbitrary discontinuous functions. It is also possible to represent Borel probability measures and measurable functions, though in this article, we consider only computations involving points and sets. In particular, we will require the data describing our systems to be in terms of open/closed sets and (semi-)continuous functions.

The representations used must allow information about the objects they describe to be obtained from a finite amount of data. Consider a computation whose result is some real number x. In traditional numerical analysis, it is usual to compute a floating-point or rational approximation x_n to x based on some accuracy parameter(s). Often some order of convergence is given, such as $|x - x_n| = O(1/n^k)$ for some integer k. Unfortunately, in this model, knowing some particular approximation x_n gives in theory *no information* on the value of x. To gain information about x, we also need to know an error bound ϵ_n for the approximation, such that $|x - x_n| < \epsilon_n$. If $\epsilon_n \to 0$, then we can compute an approximation to x with arbitrary *known* accuracy. We say that x_n *converges effectively* to x. In some problems, especially optimization problems, we merely seek a sequence of approximations x_n converging to x from above (or below). In this case, we cannot give metric bounds on x but can still deduce properties of x, such as $x > x_n$.

In theoretical work, especially when making a link between computation and topology, it is more convenient to work with *properties* of objects. For example, if $]a, b[$ is an open interval, then $x \in]a, b[$ is a property of x. Further, such properties should be *robust*, in the sense that if some property holds for x, then it holds for all y near x. Topologically, this means that a property corresponds to membership of an open set.

To describe arbitrary objects in some space, we first choose a countable collection $\sigma = \{I_1, I_2, \ldots\}$ of *basic* open sets (properties) such that x is determined uniquely by its properties. For example, if we take σ to be the collection of all open intervals $]a, b[$ with rational endpoints, then determining whether $x \in]a, b[$ for all $]a, b[\in \sigma$ is sufficient to determine the real number x. Usually, we need only know a subset of properties to determine x and all of its properties uniquely. For example, if we can enumerate a sequence of open rational intervals $]a_n, b_n[$ such that $x \in]a_n, b_n[$ for all n and $\lim_{n\to\infty} b_n - a_n = 0$, then we can determine all other intervals $]a, b[$ such that $x \in]a, b[$. Notice that the information given by approximations is equivalent to the information given by properties. For if we know $x \in]a, b[$, then $(a + b)/2$ is an approximation to x with error $\epsilon = (b - a)/2$.

In practice, we cannot determine all properties of x or compute an infinite sequence of approximations to x. Instead, we are usually content to compute sufficient information about x to be able to approximate x to some desired accuracy (which can be checked a posteriori). However, it is useful to know that x can be approximated to *any* desired accuracy. Further, by describing x by a list of *all* its properties, we can often conceptually work with the object x itself rather than with approximations to x, a considerable simplification.

While the model of computation, being based on Turing machines, subsumes ordinary, finite computation, the main purpose of computable analysis theory is to deal with approximations. In particular, the data describing the systems is interpreted as being approximate. This can drastically change the computability properties. Consider the following simple example.

Example 3.1. Consider the differential equation $\dot{x} = x^2 + \epsilon$ with initial condition

$x(0) = -1$. We wish to determine whether the solution remains bounded. If ϵ is taken to be a rational number which is described exactly, the problem is always solvable; the solution is bounded if and only if $\epsilon \leqslant 0$. However, if the *only* information we have about ϵ is approximate (possibly ϵ is an experimental parameter), then if $\epsilon = 0$, then no matter how accurate the approximation to ϵ, we cannot eliminate the possibility that $\epsilon > 0$ and that the solution is unbounded.

To summarize, boundedness of the solution in the case $\epsilon = 0$ is undecidable when using approximate data but decidable using exact data. Further, if ϵ is very close to 0, we need a very accurate approximation to ϵ in order to determine boundedness. Even in the exact model, if $\epsilon = 0$, then a very small amount of noise in the system will destroy boundedness.

The interested reader is strongly advised to read [49] for more details.

3.2. Machine computability. We now outline how to describe objects such as points, sets, and functions in the framework of computable analysis. The material in this section can be found in [49, 19].

We let Σ be a finite alphabet (such as the binary digits $\{0, 1\}$; the exact choice is unimportant). In "ordinary" computability theory, we consider computation by Turing machines working on finite words, i.e., elements of Σ^*. We consider computation by *type-two Turing machines* which work with inputs which are infinite *streams* of data, represented by elements of Σ^ω. We have natural *tupling* operations on Σ^ω, given by $p = \langle p_1, \ldots, p_k \rangle$ with $p_{jk+i} = (p_i)_j$ for $i = 0, \ldots, k-1$ and $j = 0, 1, \ldots$. We can also construct an infinite tuple $p = \langle p_1, p_2, \ldots \rangle$ by $p_{g(i,j)} = (p_i)_j$, where $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is the bijective Cantor pairing $g(i, j) = (i + j)(i + j + 1)/2 + j$.

We say that a *stream function* is a partial function $\Sigma^\omega \times \cdots \times \Sigma^\omega \to \Sigma^\omega$. A type-two Turing machine M computes a stream function η_M. The *domain* of η_M is the set of inputs for which the machine runs forever and writes infinitely many symbols to the output tape. It can be shown that the domain of η_M is always a G_δ-set (a countable intersection of open sets), and that the function η_M is continuous with respect to the product topology on Σ^ω. We therefore have a set of *machine-computable* stream functions, which is a subset of the set of all continuous stream functions with G_δ domain. Since there are uncountably many continuous stream functions and countably many Turing machines, not all continuous stream functions are machine-computable.

In this article we do not need a precise characterization of machine-computable functions, but will need the following properties:

1. If $\eta : (\Sigma^\omega)^m \dashrightarrow \Sigma^\omega$ is computable, and each $\zeta_i : (\Sigma^\omega)^n \dashrightarrow \Sigma^\omega$ is computable for $i = 1, \ldots, m$, then so is the function $\eta \circ (\zeta_1, \ldots, \zeta_m)$.
2. There is a surjective function $\varphi : \Sigma^\omega \to C_p(\Sigma^\omega; \Sigma^\omega)$ and a machine-computable function $\varepsilon : \Sigma^\omega \times \Sigma^\omega \dashrightarrow \Sigma^\omega$ such that $\varepsilon(p, q) = \varphi(p)(q)$.
3. For every computable function $f : \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$, there is a total computable function $\hat{f} : \Sigma^\omega \to \Sigma^\omega$ such that $f(p, q) = \varepsilon(\hat{f}(p), q)$, where ε is as in (2).

Part 1 asserts that computable functions are closed under composition. Part 2 asserts the existence of a *universal Turing machine*, which, given an *infinite* program p, computes the function determined by p. The function ε is the *evaluation* function, and ε is a Gödel-type numbering of partial continuous stream functions, denoted $C_p(\Sigma^\omega; \Sigma^\omega)$, which for technical reasons must be restricted to having G_δ-domain. Part 3 is a type-two version of Kleene's s_{mn} theorem on partial binding. It implies that to compute the result of an operator returning a function, is suffices to show that we can evaluate the function at every point on its domain.

3.3. Notation and representations. We relate stream functions to functions on arbitrary spaces by means of *notation* and *representations*.

A *notation* of a (necessarily countable) set W is a partial surjective function $\nu : \Sigma^* \dashrightarrow W$. Note that if W has a notation, then W must be denumerable. We say $v \in \Sigma^*$ is a *ν-name* of $w \in W$ if $\nu(v) = w$.

We say a notation is *prefix-free* if no element of $\mathrm{dom}(\nu)$ is a prefix of another. If ν is a prefix-free notation, then concatenation of words $(w_n)_{n\in\mathbb{N}}$ in $\mathrm{dom}\,\nu$ to obtain a sequence $p = w_0 w_1 w_2 \cdots$ is an injective function $(\mathrm{dom}\,\nu)^\omega \to \Sigma^\omega$, and so the elements $w_0, w_1, \ldots$ can be recovered from p.

A *representation* of a set X is a partial surjective function $\delta : \Sigma^\omega \dashrightarrow X$. If X has a representation, then it must have at most continuum cardinality. We say $p \in \Sigma^\omega$ is a *δ-name* of $x \in X$ if $\delta(p) = x$.

Let $X_0, X_1, \ldots, X_k$ be sets, and let δ_i be a representation of X_i for $i = 0, \ldots, k$. We say that a function $f : X_1 \times \cdots \times X_k \to X_0$ is $(\delta_1, \ldots, \delta_k; \delta_0)$*-computable* if there is a machine-computable partial function $\eta_M : \Sigma^\omega \times \cdots \times \Sigma^\omega \dashrightarrow \Sigma^\omega$ such that $\delta_0(\eta_M(p_1, \ldots, p_k)) = f(\delta_1(p_1), \ldots, \delta_k(p_k))$ whenever the right-hand side is defined. The function η is said to be a $(\delta_1, \ldots, \delta_k; \delta_0)$*-realizer* of f. If the representations $\delta_0, \delta_1, \ldots, \delta_k$ are clear from the context, we will simply say that f is *computable*. If f is computable, $x_0 = f(x_1, \ldots, x_k)$, and δ_i names of the x_i, $i = 1, \ldots, k$ are available, we will sometimes say that x_0 is δ_0-computable.

We say a representation δ_1 of a set X *reduces to* δ_2 if there is a machine-computable function $\eta : \Sigma^\omega \dashrightarrow \sigma^\omega$ such that $\delta_2 \circ \eta|_{\mathrm{dom}(\delta_1)} = \delta_1$. This means that the information about an element x of X provided by a δ_1-name is enough to compute a δ_2-name. We say δ_1 and δ_2 are *equivalent*, denoted $\delta_1 \equiv \delta_2$, if δ_1 reduces to δ_2 and δ_2 reduces to δ_1. Equivalent representations induce the same computability theory on a set X.

Given representations δ_1 and δ_2 for sets X_1 and X_2, we can define a representation $\delta_1 \times \delta_2$ for $X_1 \times X_2$ by interleaving names of $x_1 \in X_1$ and $x_2 \in X_2$. Similarly, given a representation of δ of X, we can define a representation δ^ω of X^ω via a numbering of $\mathbb{N} \times \mathbb{N}$.

Given representations δ_X and δ_Y for X and Y, we can define a representation $\delta_{X\to Y}$ of the space of functions $X \to Y$ by saying $\delta_{X\to Y}(p) = f$ if $\delta_Y(\varphi(p)(q)) = f(\delta_X(q))$ for all $q \in \mathrm{dom}(\varepsilon(p, \cdot))$, where φ and ε are as defined in section 3.2. In other words, $\delta_{X\to Y}(p) = f$ if, and only if, $\varphi(p)$ is a $(\delta_X; \delta_Y)$-realizer of f. This implies that the *evaluation* operator $(f, x) \mapsto f(x)$ is $(\delta_{X\to Y}, \delta_X; \delta_Y)$-computable. If $f : A \times X \to Y$ is $(\delta_A, \delta_X; \delta_Y)$-computable, then the function $\hat{f} : A \to (X \to Y)$ given by $\hat{f}(a)(x)$ is $(\delta_A; \delta_{X\to Y})$-computable by the s_{mn} theorem.

Using the above remarks, we obtain the following useful general principles for proving computability.

THEOREM 3.2.

1. *In order to compute a $\delta_{(X\to Y)}$-name of f, we need only show how to compute a δ_Y-name of $f(x)$ from a δ_X-name of x.*
2. *In order to compute a $\delta_{(X\to Y)}$-name of a parametrized function f_a from a δ_A-name of a, we need only show how to compute a δ_Y-name of $f_a(x)$ from a δ_X-name of x and the δ_A-name of a.*

3.4. Computable topological spaces. Let X be a topological space whose topology τ is generated by a countable collection of open sets $\sigma = \{I_0, I_1, \ldots\}$, and let $\nu : \Sigma^* \dashrightarrow \sigma$ be a notation of σ with prefix-free domain. Then δ is the *standard representation* of (X, τ, σ, ν) if a δ-name p of $x \in X$ has the form $p = w_0 w_1 w_2 \ldots$ with

each $w_i \in \mathrm{dom}(\nu)$ such that $\{w_0, w_1, w_2, \ldots\} = \{w \in \mathrm{dom}(\nu) \mid x \in \nu(w)\}$. Informally, we say that a δ-name of x encodes a list of all $I \in \sigma$ such that $x \in I$. We say a representation is *admissible* if it is equivalent to the standard representation.

In general, a given topological space has many equivalence classes of admissible representation, corresponding to different choice of sub-basic sets σ and notation ν. However, by a result of Bauer [7, Theorem 5.5.18], there is a unique equivalence class of representations of $\mathbb{R}$ compatible with the topology, making 1 a computable number, arithmetic operations $+, -, \times, \div$ computable, and strict comparison $x > 0$ verifiable. We can therefore speak of a canonical computable real number type. From this canonical representation of $\mathbb{R}$, we can build canonical representations of $\mathbb{R}^n$ by tupling, and build canonical representations of functions and sets.

A concrete way of building a standard representation of X with sub-base σ using binary digits is to take $\mathrm{dom}(\nu) = \mathtt{1}^*\mathtt{0}$, with $\nu(\mathtt{1}^n\mathtt{0}) = I_n$. The sequence $\mathtt{1}^{n_0}\mathtt{0}\mathtt{1}^{n_1}\mathtt{0}\mathtt{1}^{n_2}\mathtt{0}\cdots$ encodes the sequence of generators $I_{n_0}, I_{n_1}, I_{n_2}, \ldots$, which (if ν is injective) is a name for x if $\{I_{n_j} \mid j \in \mathbb{N}\} = \{I \in \sigma \mid x \in I\}$.

We will restrict our attention to hybrid automata such that the state space X is a locally compact countably based Hausdorff space. We take $\sigma = \beta$, where β is a countable base for X consisting of sets with compact closure. For convenience, we will sometimes write $\overline{\beta}$ for $\{\overline{I} \mid I \in \beta\}$, where $\overline{I}$ denotes the closure of I. We denote the standard representation of (X, τ, β, ν) by ρ_X, which we abbreviate to ρ when the space X is clear from the context.

As in Brattka and Presser [13] we use the following *effectivity properties* of the notation ν of β. These are necessary in order to perform certain computations on sets and functions.

1. (Disjointness) The set of pairs $(I, J) \in \beta^2$ such that $\overline{I} \cap \overline{J} = \emptyset$ is recursively enumerable.
2. (Overlap) The set of pairs $(I, J) \in \beta^2$ such that $I \cap J \neq \emptyset$ is recursively enumerable.
3. (Subset) The set of pairs $(I, J) \in \beta^2$ such that $\overline{I} \subset J$ is recursively enumerable.
4. (Cover) The set of tuples $(I, J_1, \ldots, J_k) \in \beta^*$ such that $\overline{I} \subset \bigcup_{i=1}^k J_i$ is recursively enumerable.

For Euclidean space $X = \mathbb{R}^n$, we take β to be the collection of all open bounded boxes with rational endpoints; $]a_1, b_1[\times]a_2, b_2[\times \cdots \times]a_n, b_n[$ with $a_i, b_i \in \mathbb{Q}$ for $i = 1, 2, \ldots, n$. Using any "reasonable" notation ν of β gives the canonical representation of $\mathbb{R}^n$.

3.5. Computability for continuous functions. We now wish to give concrete representations for spaces of continuous functions $C(X; Y)$ and continuous partial functions $C_p(X; Y)$. If the domain of the function is a locally compact Hausdorff space, then the standard way of doing this is via the *compact-open* representation. For if $f : X \to Y$ is continuous, and $J \subset Y$ is open, then $f^{-1}(Y)$ is open. Hence if $\overline{I}$ is compact, then the property $\overline{I} \subset f^{-1}(J)$ is robust. Alternatively, if $\overline{I}$ is compact, then $f(\overline{I})$ is compact, so the property $f(\overline{I}) \subset J$ is robust and equivalent to $\overline{I} \subset f^{-1}(J)$.

Definition 3.3.

1. *A $\gamma_{(X;Y)}$-name of continuous $f : X \to Y$ encodes a list of all $(I, J) \in \beta_X \times \beta_Y$ such that $\overline{I} \subset f^{-1}(J)$.*

If the domain and codomain of the functions are clear from the context, we simply write γ instead of $\gamma_{(X;Y)}$.

We can extend the compact-open names to partial functions with open or closed domains.

2. *A $\gamma_<$-name of a continuous partial function $f : X \dashrightarrow Y$ with open domain U encodes a list of all pairs $(I, J) \in \beta_X \times \beta_Y$ such that $\bar{I} \subset U$ and $f(\bar{I}) \subset J$.*
3. *A $\gamma_>$-name of a continuous partial function $f : X \dashrightarrow Y$ with closed domain A encodes a list of all pairs $(I, J) \in \beta_X \times \beta_Y$ such that $f(\bar{I} \cap A) \subset J$.*

For the case of functions defined on intervals in $\mathbb{R}^+$, we will also need to consider half-open intervals. Hence a $\gamma_{\lessgtr}$-name of a continuous function $f : [0, t[\to Y$ encodes a list of all pairs $(I, J) \in \beta_{\mathbb{R}} \times \beta_Y$ such that $\bar{I} \subset (-\infty, t)$ and $f(\bar{I} \cap [0, t[) \subset J$.

Note that in 2 a name of f implicitly contains a description of the domain U, since $U = \bigcup\{\bar{I} \in \overline{\beta} \mid \bar{I} \subset U \text{ and } \exists\, J \in \beta_Y \text{ such that } f(\bar{I}) \subset J\}$. In 3, a name of f implicitly contains a description of the domain A since $X \setminus A = \bigcup\{\bar{I} \in \overline{\beta} \mid f(\bar{I} \cap A) \subset \emptyset\}$.

The following result is part of [49, Theorem 6.1.7].

THEOREM 3.4. *If X is a locally compact Hausdorff space, then the compact-open representation $\gamma_{(X;Y)}$ is equivalent to the representation $\delta_{X \to Y}$. In particular,*

1. *function evaluation $C(X; Y) \times X \to Y$ is $(\gamma_{(X;Y)}, \rho_X; \rho_Y)$-computable;*
2. *if $f : X \to Y$, and it is possible to compute a ρ_Y-name of $f(x)$ from a ρ_X-name of x, then it is possible to compute a $\gamma_{(X;Y)}$ name of $\hat{f}$.*

In other words, the information provided by a $\gamma_{(X;Y)}$-name of f is *precisely* the information needed to evaluate f.

3.6. Computability theory for sets. When considering reachability analysis, we will need to have a way of describing the reachable sets themselves and to also guard sets and set-valued operators used in the system description. The following representations of open, closed, and compact sets, which are given in [49, section 5.1] are most useful.

DEFINITION 3.5. *Let X be a locally compact Hausdorff space with a countable base β of open sets with compact closures.*

1. *A $\theta_<$-name of open $U \subset X$ encodes a list of all $I \in \beta$ such that $\bar{I} \subset U$.*
2. *A $\psi_>$-name of closed $A \subset X$ encodes a list of all $I \in \beta$ such that $\bar{I} \cap A = \emptyset$.*
3. *A $\psi_<$-name of closed $A \subset X$ encodes a list of all $I \in \beta$ such that $I \cap A \neq \emptyset$.*
4. *A $\kappa_>$-name of compact $C \subset X$ encodes a list of all tuples $(I_1, \ldots, I_k) \in \beta^*$ such that $C \subset \bigcup_{i=1}^k I_j$.*

Informally we say that a set-valued operator is lower-semicomputable *if it is possible to compute a lower ($\theta_<$ or $\psi_<$) name of the result, and* upper-semicomputable *if it is possible to compute an upper ($\psi_>$ or $\kappa_>$) name.*

It is easy to see that each of the properties encoded is robust with respect to a small change in the set being described. For example, if $\bar{I}$ is a subset of some open set U, then $\bar{I}$ is also a subset of V for any sufficiently small perturbation V of U. The information given by $\theta_<$ is sufficient to compute a sequence of sets (described as finite unions of boxes) converging to U from inside, and the information given by $\kappa_>$ is sufficient to compute a convergent sequence of outer-approximations to C. Notice that the information provided by a $\theta_<$-name of U is exactly the same as that provided by a $\psi_>$-name of $X \setminus U$.

In general, we use $\theta_<$ and $\psi_>$-names for sets used as invariant and guard sets in the system description. In practice, a set D used as a guard for an urgent transition will usually be *regular*, which means that $\mathrm{cl}(D) = \mathrm{cl}(\mathrm{int}(D))$ and $\mathrm{int}(D) = \mathrm{int}(\mathrm{cl}(D))$. In this case, we can give both a $\theta_<$-name of $\mathrm{int}(D)$ and a $\psi_>$-name of $\mathrm{cl}(D)$. We use $\psi_<$ and $\kappa_>$ as names for initial and reachable sets.

The information about an open set U provided by $\theta_<$ allows U to be constructed as a countable union of basic sets. In particular, if β consists of successively finer *pavings* of Euclidean space (see [33, Chapter 3]), then it is possible to extract from a $\theta_<$-name of U a list of nonoverlapping boxes J_i such that $U = \bigcup_{i=1}^{\infty} \overline{J}_i$. The information given by $\kappa_>$ is sufficient to give a sequence of successively finer outer-approximations to a compact set C within a bounded domain. The information provided by $\psi_<$ is not sufficient to give an under-approximation to a closed set A, but this should not be seen as surprising, since a compact set may have empty interior. Instead, for a fixed $\epsilon > 0$, we can provide an under-approximation A_ϵ to the ϵ-neighborhood $N_\epsilon(A)$, and for a sequence $\epsilon_i \to 0$, we can give under-approximations A_i to $N_{\epsilon_i}(A)$ such that A_i converges to A as $i \to \infty$. The information given by both $\psi_<$ and $\kappa_>$ is sufficient to compute a compact set C to arbitrary accuracy in the Hausdorff metric.

Using these representations, the first natural question to ask is which geometric operations (union, intersection) are computable. We will need the following results.

THEOREM 3.6 (computable set-based operators).

1. *Finite union is computable, i.e.,* $(\theta_<, \theta_<; \theta_<)$*-computable,* $(\psi_<, \psi_<; \psi_<)$*-computable,* $(\psi_>, \psi_>; \psi_>)$*-computable, and* $(\kappa_>, \kappa_>; \kappa_>)$*-computable.*
2. *Finite intersection of open sets is lower-semicomputable, i.e.,* $(\theta_<, \theta_<; \theta_<)$*-computable, and finite intersection of closed sets is upper-semicomputable, i.e.,* $(\psi_>, \psi_>; \psi_>)$*-computable.*
3. *The closure of the intersection of an open and a closed set is lower-semicomputable, i.e.,* $(U, A) \mapsto \mathrm{cl}(U \cap A)$ *is* $(\theta_<, \psi_<; \psi_<)$*-computable.*
4. *The intersection of a closed and a compact set is upper-semicomputable, i.e.,* $(A, C) \mapsto A \cap C$ *is* $(\psi_>, \kappa_>; \kappa_>)$*-computable.*
5. *Countable union of open sets, and countable closed union of closed sets are lower-semicomputable, i.e.,* $(U_1, U_2, \ldots) \mapsto \bigcup_{n=1}^{\infty} U_i$ *is* $(\theta_<^\omega; \theta_<)$*-computable and* $(A_1, A_2, \ldots) \mapsto \mathrm{cl}(\bigcup_{n=1}^{\infty} A_n)$ *is* $(\psi_<^\omega; \psi_<)$*-computable.*
6. *The countable intersection of closed sets is upper-semicomputable i.e.,* $(A_1, A_2, \ldots) \mapsto \mathrm{cl}(\bigcup_{n=1}^{\infty} A_n)$ *is* $(\psi_>^\omega; \psi_>)$*-computable.*
7. *The singleton set operator* $x \mapsto \{x\}$ *is* $(\rho; \psi_<)$*-computable and* $(\rho; \kappa_>)$*-computable.*
8. *The closed set-image operator* $(f, A) \mapsto \mathrm{cl}(f(A))$ *is* $(\gamma, \psi_<; \psi_<)$*-computable and the set-image operator* $(f, C) \mapsto f(C)$ *is* $(\gamma, \kappa_>; \kappa_>)$*-computable.*
9. *The set-preimage operator* $(f, U) \mapsto f^{-1}(U)$ *is* $(\gamma, \theta_<; \theta_<)$*-computable.*

The following operations are uncomputable in general:

1. The intersection of two closed sets $(A, B) \mapsto A \cap B$ is not lower-semicomputable, i.e., not $(\psi_<, \psi_<; \psi_<)$-computable.
2. The interior of a closed set is not $(\psi_<; \theta_<)$-computable.
3. The interior of the image of an open set under a continuous function $(f, U) \mapsto \mathrm{int}(f(U))$ is not lower-semicomputable, i.e., not $(\gamma, \theta_<; \theta_<)$-computable.
4. The closure of the image of a closed set under a continuous function $(f, A) \mapsto \mathrm{cl}(f(A))$ is not upper-semicomputable, i.e., not $(\gamma, \psi_>; \psi_>)$-computable.

The most important restriction is that on lower-semicomputability of the intersection of two closed sets. This states that it is not possible to enumerate all basic open sets $I \in \beta$ such that $(A \cap B) \cap I \neq \emptyset$ from similar enumerations for A and B. However, if U is open, then we *can* lower-compute $\mathrm{cl}(A \cap U)$, and this is usually sufficient in practice.

3.7. Computability theory for multivalued maps. In many applications, it is convenient to represent systems by nondeterministic models defined by multivalued

functions. Further, as we shall see, nondeterminism is unavoidable if we are to give a framework for hybrid automata under which we can compute the evolution.

We say F is a *multivalued function* from X to Y, denoted $F : X \rightrightarrows Y$, if F associates to each $x \in X$ a subset $F(x)$ of Y. If $A \subset X$, we define $F(A) = \bigcup_{x \in A} F(x)$. If $F : X \rightrightarrows Y$ and $G : Y \rightrightarrows Z$, we define $G \circ F : X \rightrightarrows Z$ by $G \circ F(x) := G(F(x)) = \bigcup_{y \in F(x)} G(y)$. If $F_1, F_2 : X \rightrightarrows Y$, we define $F_1 \cup F_2$ by $(F_1 \cup F_2)(x) = F_1(x) \cup F_2(x)$. If $A \subset X$, we define $F|_A$ by $F_A(x) = F(\{x\} \cap A)$, and if $B \subset Y$, define $F|^B$ by $F|^B(x) = F(x) \cap B$. The (weak) *preimage* $F^{-1} : Y \rightrightarrows X$ of a multivalued function $F : X \rightrightarrows Y$ is defined by $F^{-1}(B) = \{x \in X \mid F(x) \cap B \neq \emptyset\}$. The *strong preimage* $F^{\Leftarrow} : Y \rightrightarrows X$ is defined by $F^{\Leftarrow}(B) = \{x \in X \mid F(x) \subset B\}$. Note that $F^{\Leftarrow}(B) = X \setminus (F^{-1}(Y \setminus B))$. We say F is *lower-semicontinuous* if $F^{-1}(V)$ is open whenever V is open, and *upper-semicontinuous* if $F^{-1}(B)$ is closed whenever B is closed. F is *continuous* if it is both lower- and upper-semicontinuous.

Note that if $F : X \rightrightarrows Y$ is closed-valued lower-semicontinuous and C is compact, then $F(C)$ need not be closed, but for any set A, we have $F(\mathrm{cl}(A)) \subset \mathrm{cl}(F(A))$.

Remark 3.7. The definitions of upper- and lower-semicontinuity used here are standard [35]. The notion of *outer-semicontinuity* was used in [30], based on definitions in [46]. A closed-valued function is outer-semicontinuous if $\limsup_{n\to\infty} F(x_n) = F(x_\infty)$ for all convergent sequences $x_n \to x_\infty$. Equivalently, F is outer-semicontinuous if $F^{-1}(C)$ is closed whenever C is compact, or if $\mathrm{graph}(F)$ is a closed set. A function is upper-semicontinuous if, and only if, it is outer-semicontinuous and locally bounded.

For dynamical systems, upper-semicontinuity is more appropriate than outer-semicontinuity. For if F is compact-valued upper-semicontinuous and C is compact, then $F(C)$ is also compact, whereas if F is compact-valued but merely outer-semicontinuous, then $A = F(C)$ is closed, and then $F(A)$ need not be closed and cannot be computed.

If $f : X \times U \to X$ is continuous, we can define $F(x) = f(x, U)$, so multivalued functions can be used to model systems with control or disturbance inputs. The function $x \mapsto \mathrm{cl}(F(x))$ is closed-valued lower-semicontinuous, and F is compact-valued upper-semicontinuous if U is compact. In control theory, lower-semicontinuous functions are appropriate for modeling control systems with input, and upper-semicontinuous functions are appropriate to model systems with disturbances. Indeed, lower-semicontinuity is required to prove that a trajectory with some property *exists*, whereas upper-semicontinuity is required to prove that *all* trajectories have some property.

We have the following representations of multivalued maps.

Definition 3.8.

1. *A $\mu_<$ name of a lower-semicontinuous map $F : X \rightrightarrows Y$ with closed values encodes a list of all pairs $(I, J) \in \beta_X \times \beta_Y$ such that $\bar{I} \subset F^{-1}(J)$.*
2. *A $\mu_>$ name of an upper-semicontinuous map $F : X \rightrightarrows Y$ with compact values encodes a list of all tuples $(I, J_1, \ldots, J_k) \in \beta_X \times \beta_Y^*$ such that $F(\bar{I}) \subset \bigcup_{i=1}^k J_i$.*

Theorem 3.9 (computability for multivalued maps).

1. *The representation $\mu_<$ of closed-valued lower-semicontinuous $F : X \rightrightarrows Y$ is equivalent to the $\delta_{(X \to \mathcal{A}(Y))}$ representation of F, i.e., the representation of f as a map $X \to \mathcal{A}(Y)$ induced by the ρ-representation on X and the $\psi_<$-representation on $\mathcal{A}(Y)$.*
2. *The representation $\mu_>$ of compact-valued upper-semicontinuous $F : X \rightrightarrows Y$ is equivalent to the $\delta_{(X \to \mathcal{K}(Y))}$ representation of F.*
3. *The (weak) preimage of an open set under a closed-valued lower-semicontinuous*

function is lower-semicomputable, i.e., $(F, V) \mapsto F^{-1}(V)$ *is* $(\mu_<, \theta_<; \theta_<)$*-computable.*

4. *The (weak) preimage of a closed set under a compact-valued upper-semicontinuous function is upper-semicomputable, i.e.,* $(F, B) \mapsto F^{-1}(B)$ *is* $(\mu_>, \psi_>; \psi_>)$*-computable. Equivalently, the strong preimage of an open set is lower-semicomputable i.e.,* $(F, V) \mapsto F^{\Leftarrow}(V)$ *is* $(\mu_>, \theta_<; \theta_<)$*-computable.*
5. *The closure of the image of a closed set under a closed-valued lower-semicontinuous function is lower-semicomputable, i.e.,* $(F, A) \mapsto \mathrm{cl}(F(A))$ *is* $(\mu_<, \psi_< ; \psi_<)$*-computable.*
6. *The image of a compact set under a compact-valued upper-semicontinuous function is upper-semicomputable, i.e.,* $(F, C) \mapsto \mathrm{cl}(F(A))$ *is* $(\mu_>, \kappa_>; \kappa_>)$*-computable.*

By property 1, if we have a closed-valued lower-semicontinuous multivalued function F and can compute $x \mapsto F(x)$ in the sense that given a ρ-name of x we have an algorithm to generate a $\psi_<$-name of $F(x)$, then we can generate a $\mu_<$-name of F. An analogous result holds for compact-valued upper-semicontinuous F.

We obtain similar results for partial functions.

LEMMA 3.10. *Let* $f : X \dashrightarrow Y$ *be a partial function with domain* D*, let* C *be compact, and let* A *be closed. Then the map* $(f, C) \mapsto f(C \cap D)$ *is* $(\gamma_>, \kappa_>; \kappa_>)$*-computable if* D *is closed, and the map* $(f, A) \mapsto \mathrm{cl}(f(A \cap D))$ *is* $(\gamma_<, \theta_<; \psi_<)$*-computable if* D *is open.*

Proof. $f(C \cap D) \subset J$ if, and only if, there is a finite cover of C by open sets I_k such that $f(\bar{I}_k \cap D) \subset J$ for all k.

$f(A \cap D)$ intersects J if, and only if, there exists $x \in A \cap D$ such that $f(x) \in J$. Then there exists I such that $\bar{I} \subset D$, $x \in I$, and $f(\bar{I} \cap A) \subset J$. ∎

3.8. Computability theory for multivalued flows. A multivalued flow or *multiflow* can be thought of as a function assigning to each initial point $x \in X$ a set of continuous trajectories $\eta : \mathbb{R}^+ \dashrightarrow X$. In other words, a multiflow is a function $\Phi : X \rightrightarrows C(\mathbb{R}^+; X)$ such that

1. $\eta(0) = x$ for all $\eta \in \Phi(x)$.

We also require multivalued versions of the standard semiflow properties as follows:

2. If $\eta \in \Phi(x)$ and $s \in \mathbb{R}^+$, then the function ζ defined by $\zeta(t) = \eta(t + s)$ is in $\Phi(\eta(s))$.
3. If $\eta_1 \in \Phi(x)$ and $\eta_2 \in \Phi(\eta_1(s))$, then the function ζ defined by $\zeta(t) = \xi_1(t)$ for $t \leqslant s$ and $\zeta(t) = \xi_2(t - s)$ for $t \geqslant s$ is in $\Phi(x)$.

Property 2 means that the system is *time-invariant*, and property 3 is the *property of state*. Such properties are well known in the dynamical systems and control theory literature; see, e.g., Kalman, Falb, and Arbib [34] or Polderman and Willems [43, Definitions 1.4.2 and 4.3.3]. Multivalued flows are the natural objects to represent solutions of nondeterministic continuous-time systems, such as differential inclusions.

Since we are interested only in continuous behavior between two events, we need to consider trajectories which are defined on subintervals of $\mathbb{R}$. For lower-semicontinuous multiflows, we consider continuous trajectories $\eta : [0, T[\to X$, and for upper-semicontinuous multiflows we consider $\eta : [0, T] \to X$ or $\eta : [0, \infty[\to X$. As usual, semicontinuity is defined in terms of preimages of open/closed subsets of $C(\mathbb{R}^+; X)$.

We have the following representations of multivalued flows.

DEFINITION 3.11.

1. *A $\phi_<$-name of a lower-semicontinuous multivalued flow Φ encodes a list of all $(I, J_1, K_1, \ldots, J_m, K_m) \in \beta_X \times (\beta_{\mathbb{R}} \times \beta_X)^*$ such that for all $x \in \overline{I}$, there is a solution ξ such that $\xi(0) = x$, and, for all $i = 1, \ldots, m$, $\overline{J}_i \subset \mathrm{dom}(\xi)$ and $\xi(\overline{J}_i) \subset K_i$.*
2. *A $\phi_>$-name of an upper-semicontinuous multivalued flow Φ encodes a list of all $(I, J, K_1, \ldots, K_m) \in \beta_X \times \beta_{\mathbb{R}} \times \beta_X^*$ such that for all $x \in \overline{I}$, and all curves ξ such that $\xi(0) = x$, $\xi(\overline{J} \cap \mathrm{dom}(\xi)) \subset \bigcup_{i=1}^m K_i$.*

Given a multiflow Φ, we can define a function $\tilde{\Phi} : X \times \mathbb{R}^+ \rightrightarrows X$ by $\tilde{\Phi}(x,t) = \{\xi(t) \mid \xi \in \Phi(x) \text{ and } t \in \mathrm{dom}(\xi)\}$. In other words, $\tilde{\Phi}(x,t)$ is the set of points reachable from x at time t. The standard representations of $\tilde{\Phi}$ are the representation $\mu_<$ and $\mu_>$ of $\tilde{\Phi}$ as a multivalued map. Explicitly,

3. a $\mu_<$-name of a lower-semicontinuous closed-valued multiflow Φ encodes a list of all $(I, J, K) \in \beta_X \times \beta_{\mathbb{R}} \times \beta_X$ such that for all $x \in \overline{I}$ and $t \in \overline{J}$, there is a solution ξ such that $\xi(0) = x$ and $\xi(t) \in K$.

The information given by a $\phi_>$-name of an upper-semicontinuous compact-valued multiflow Φ is exactly the same the information provided by a $\mu_>$-name of $\tilde{\Phi}$ considered as a multivalued function $\Phi : X \times \mathbb{R}^+ \rightrightarrows X$, but the information given by a $\phi_<$-name of a lower-semicontinuous multiflow Φ is strictly stronger than the information provided by a $\mu_<$-name of $\tilde{\Phi}$. It turns out that the information given by a $\mu_<$-name of $\tilde{\Phi}$ is insufficient for studying reachable sets of hybrid automata, since it encodes only information about the reachable sets and not about the trajectory in between.

In order to work with multiflows, the following result will be useful.

PROPOSITION 3.12.

1. *Let Φ be an upper-semicontinuous multiflow, and let $F : C(\mathbb{R}^+; X) \rightrightarrows Y$ be compact-valued upper-semicontinuous. Then a $\mu_>$-name of $F \circ \Phi : X \rightrightarrows Y$ can be computed from a $\phi_>$-name of Φ and a $\mu_>$-name of F.*
2. *Let Φ be a lower-semicontinuous multiflow, and let $F : C(\mathbb{R}^+; X) \rightrightarrows Y$ be closed-valued lower-semicontinuous. Then a $\mu_<$-name of $F \circ \Phi : X \rightrightarrows Y$ can be computed from a $\phi_<$-name of Φ and a $\mu_<$-name of F.*

Proof.

1. Suppose $F(\eta) \in \bigcup_{i=1}^m L_i$ whenever $\eta(\overline{J}_i) \subset K_{i,j}$ for all $i = 1, \ldots, m$ and $j = 1, \ldots, n_i$. Suppose further that $\eta(\overline{J}_i) \subset \bigcup_{j=1}^{n_i} K_{i,j}$ whenever $\eta \in \Phi(\overline{I})$. Then $F(\Phi(\overline{I})) \subset \bigcup_{i=1}^m L_i$. Further, if $F(\Phi(\overline{I})) \subset \bigcup_{i=1}^m L_i$, then such J_i and $K_{i,j}$ exist. The result follows from the definitions of the representations $\mu_>$ and $\phi_>$.
2. Suppose $F(\eta) \cap L \neq \emptyset$ whenever $\eta(\overline{J}_i) \subset K_i$ for all $i = 1, \ldots, m$. Suppose further that $\eta(\overline{J}_i) \subset K_i$ whenever $\eta \in \Phi(\overline{I})$. Then $F(\Phi(x)) \cap L \neq \emptyset$ for all $x \in \overline{I}$; equivalently, $\overline{I} \subset (F \circ \Phi)^{-1}(L)$. Further, $F(\Phi(x)) \cap L \neq \emptyset$ for all $x \in \overline{I}$, when such J_i and K_i exist. The result follows from the definitions of the representations $\mu_<$ and $\phi_<$. $\square$

The most common way of modeling continuous dynamics is as a differential system—either a differential equation $\dot{x} = f(x)$ or a differential inclusion $\dot{x} \in F(x)$. A solution to the differential inclusion $\dot{x} \in F(x)$ is an absolutely continuous function $\xi : T \to X$, where T is an interval in $\mathbb{R}$ containing 0, such that $\dot{\xi}(t) \in F(\xi(t))$ for almost all $t \in T$. The flow Φ of $\dot{x} \in F(x)$ takes x to the set of solution trajectories starting at x.

There is considerable work in the literature on semicontinuity properties of the solutions of a differential inclusion; see [3] for an overview. The solution of

a general locally Lipschitz continuous differential inclusion was shown to be computable (using different terminology) in [45]. We can refine this result and consider lower-semicomputability and upper-semicomputability separately; see [22]. We say $F : X \rightrightarrows TX$ has *linear growth* if there exists a constant c such that $||y|| \leqslant c(1 + |x|)$ for all $x \in X$ and $y \in F(x)$, where $|x|$ denotes the $d(x, x_0)$ from an arbitrary base-point x_0.

THEOREM 3.13 (computability of differential inclusions). *Let $\Phi : X \rightrightarrows C(\mathbb{R}^+; X)$ denote the flow of the differential inclusion $\dot{x} \in F(x)$.*

1. *If F is upper-semicontinuous with compact convex values and linear growth, then the solution operator $F \mapsto \Phi$ is upper-semicomputable; more precisely, $F \mapsto \Phi$ is $(\mu_>; \phi_>)$-computable..*
2. *If F is locally Lipschitz lower-semicontinuous with closed convex values, then the solution operator $F \mapsto \Phi$ is lower-semicomputable; more precisely, $F \mapsto \Phi$ is $(\mu_<; \phi_<)$-computable.*

For the theoretical results of this paper, we work with multiflows to avoid the technical complications of dealing with differential inclusions.

If $D \subset X$, we define the restriction of the trajectory ξ to D by setting

$$\xi|^D = \xi|_{\{t \in \mathbb{R}^+ | \xi([0,t]) \subset D\}}. \tag{14}$$

Note that $\mathrm{dom}(\xi|^D)$ is open in $\mathbb{R}^+$ if D is open in X, and closed if D is closed. For a multiflow Φ, we define $\Phi|^D(x) = \{\xi|^D \mid \xi \in \Phi(x)\}$.

4. Uncomputability of the evolution of hybrid automata. We now consider computability of the evolution operator Φ^H of a hybrid automaton H, as given by Problem 2.7. The main aim of this section is to show that, in general, it is impossible to compute the solutions of a general hybrid system to arbitrary accuracy. Since we are trying to prove uncomputability results, we obtain the strongest results by restricting our attention to the simplest class of systems, namely deterministic hybrid automata with dynamic $\dot{x} = f_q(x)$, resets $r_e(q, x)$, and guards $g_e(q, x) \geqslant 0$. We take state space $X = \bigcup_{q \in Q} \big(\{q\} \times \mathbb{R}^{n_q}\big)$ and set $X_i = \mathbb{R}^{n_{q_i}}$. We denote states by $(q, x_1, \ldots, x_{n_q})$; in particular, the state is written as (q) if $n_q = 0$ and as (q, x_1, x_2) or (q, x, y) if $n_q = 2$.

4.1. Temporal discontinuities. We first give a trivial example to show that the evolution may vary discontinuously in time.

Example 4.1. Let H be the deterministic hybrid automaton with two modes, q_1 and q_2, with $X_1 = \mathbb{R}$ and $X_2 = \mathbb{R}^0$. The dynamics in X_1 is constant, $\dot{x} = c$. There is a single event with reset map $r(q_1, x) = (q_2)$ with guard $g(q_1, x) = x - a$, yielding guard set $G = \{x \mid x - a \geqslant 0\}$. See Figure 2.

Let the initial condition be $x(0) = x_0$. Then if $x_0 > a$, the event is immediately activated and the final state is (q_2) for all $t > 0$. If $x_0 < a$ and $c > 0$, then the event e occurs when $t = t_1 = (a - x_0)/c$. Hence for $t < t_1$, the state is $(q_1, x_0 + ct)$, and for $t > t_1$, the state is (q_2). Hence the evolution is discontinuous in t.

Of course, time discontinuities are the essence of hybrid automaton dynamics. In section 5 we see that temporal discontinuities can be handled as long as they do not occur at the final evolution time and are not also associated with spatial discontinuities.

4.2. Spatial discontinuities. We now give several examples to show that the evolved sets vary *discontinuously* with system parameters and initial condition, even when no transition occurs at the final evolution time.

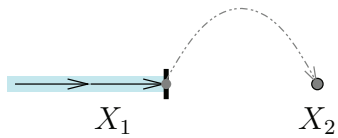

FIG. 2. *A hybrid automaton with two discrete modes and piecewise-constant dynamics exhibiting a temporal discontinuity.*

Example 4.2 (discontinuity induced by tangency with guard set). Let H be a hybrid automaton with two modes q_1 and q_2, with $X_1 = \mathbb{R}^2$ and $X_2 = \mathbb{R}^0$. The dynamics in X_1 is affine, $(\dot{x}, \dot{y}) = (2y, -1)$. There is a single reset map with $r(q_1, x, y) = (q_2)$ with guard $g(q_1, x) = x - a$ which is active when $x \geqslant a$. The solution to the continuous dynamics in mode q_1 is $(x(t), y(t)) = (x_0 + 2y_0 t - t^2, y_0 - t)$. The maximum value of x is $x_0 + y_0^2$ and is attained when $t = y_0$. See Figure 3.

Suppose the initial condition is (q_1, x_0, y_0) with $x_0 = -1$ and $y_0 = +1$. Then $x(t)$ reaches a maximum value of 0 at $t = 1$. Consider the set $\Psi^H((q_1, x_0, y_0), t)$ with $t = 2$. Then if $a > 0$, the constraint $g(x) \geqslant 0$ is not satisfied, and the reached state is $(q_1, -1, -1)$. However, if $a < 0$, the guard condition is satisfied for some $t < 1$, and the state at time $t = 2$ is (q_2). Hence the evolution is discontinuous in the parameter a.

Now suppose that a is fixed at 0, and the initial condition is $(q_1, x_0, 1)$ with $x_0 < 0$. Then for $x_0 < -1$, the maximum value of x is $1 + x_0$, which is less than a, so the event is never active and the reached state is $(q_1, x_0, -1)$. However, if $x_0 > -1$, then the guard condition is satisfied at some $t < 1$ and the reached state is (q_2). Hence the evolution is discontinuous in the parameter a.

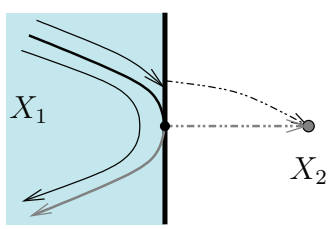

FIG. 3. *A hybrid automaton with two discrete modes and piecewise-affine dynamics exhibiting a grazing discontinuity.*

Example 4.3 (discontinuity induced by corner collisions). Let H be a hybrid automaton with three modes q_1, q_2, and q_3, with $X_1 = \mathbb{R}^2$ and $X_2 = X_3 = \mathbb{R}^0$. The dynamics in X_1 has constant derivative, $(\dot{x}, \dot{y}) = (1, 1)$. There are two events, e_2 and e_3, with reset maps, r_2 and r_3 with $r_2(q_1, x, y) = (q_2)$ and $r_3(q_1, x, y) = (q_3)$, and guards $g_2(q_1, x, y) = x - a$ and $g_3(q_1, x, y) = y - b$. See Figure 4.

Suppose the initial condition is (q_1, x_0, y_0) with $x_0 = y_0 = 0$. Then if $0 < a < b < 1$, the event e_2 is activated before e_3, and the state at time $t = 1$ is (q_2). If $0 < b < a < 1$, then event e_3 is activated before e_2, and the state at time $t = 1$ is (q_3). Hence the evolution is discontinuous in the parameters a and b. In a similar way, we can show that the evolution is discontinuous in the initial state.

Example 4.4 (discontinuity induced by immediately activated events). Let H be a hybrid automaton with three modes q_1, q_2, and q_3 with $X_1 = \mathbb{R}^2$, $X_2 = \mathbb{R}^1$, and

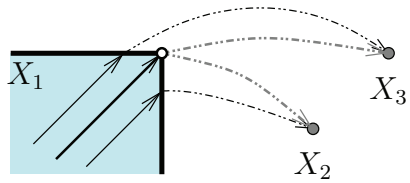

FIG. 4. *A hybrid automaton with three discrete modes, affine guard sets, and piecewise-constant dynamics exhibiting a corner discontinuity.*

$X_3 = \mathbb{R}^0$, and two events e_1 and e_2. The dynamics in X_1 has constant derivative, $(\dot{x}, \dot{y}) = (1, 0)$, and the dynamics in X_2 is $\dot{x} = 1$. The event e_1 may occur in mode q_1, with guard $x \geqslant a$ and reset $r_1(q_1, x, y) = (q_2, x + y)$. The event e_2 may occur in mode q_2, with guard $x \leqslant 0$ and reset $r_2(q_2, z) = (q_3)$. See Figure 5.

Suppose the initial condition is (q_1, x_0, y_0) with $x_0 = -1$ and $y_0 = 0$. Then the event e_1 is activated at $(q_1, a, 0)$ and the state is reset to (q_2, a). If $a < 0$, event e_2 is immediately activated, and a transition occurs to state (q_3). If $a > 0$, then the continuous state z in mode q_2 satisfies $z \geqslant a > 0$, and so event e_2 is never activated, and the state at time t for $t > 1$ is $(q_2, t - 1)$. Hence the evolution is discontinuous in the parameters.

If the initial state is $(q_1, -1, y_0)$, then the event e_1 is activated at (q_1, a, y_0) and the state is reset to (q_2, z) with $z = a + y_0$. If $y_0 > -a$, the state remains in mode q_2, whereas if $y_0 < -a$, then $z < 0$ and event e_2 is immediately activated and the state is reset to (q_3). Hence the evolution is discontinuous in the initial state.

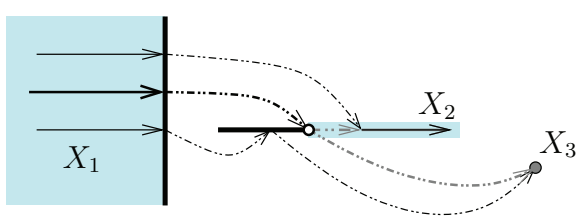

FIG. 5. *A hybrid automaton with three discrete modes, affine guard sets, and piecewise-constant dynamics exhibiting a discontinuity caused by an immediately activated event.*

4.3. Coherent semantics of evolution. We have seen that the evolution operator Ψ^H of a non-Zeno hybrid automaton may be discontinuous in both space and time, even for affine systems. By the fundamental theorem of computable analysis, this means that the evolution is uncomputable, at least near the discontinuity points. This does not in itself rule out the possibility of regularizing the evolution in some way so that the evolution becomes computable. In section 5 we shall show that by using appropriately defined nondeterministic semantics, we can make the evolution semicomputable. In this subsection we prove that it is impossible to regularize the evolution near continuity points to make the evolution fully computable, i.e., both lower- and upper-semicomputable.

DEFINITION 4.5 (coherent semantics of evolution). *Let $H = (X, f, r, g)$ be a deterministic hybrid automaton, and let $U \subset X \times \mathbb{R}^+$ be the domain of continuity of the evolution operator $\Psi^H : X \times \mathbb{R}^+ \to X$. We say that a set-valued evolution operator $\widehat{\Psi} : X \times \mathbb{R}^+ \rightrightarrows X$ has* coherent semantics *with respect to the domain U if $\widehat{\Psi}(x, t) = \{\Psi^H(x, t)\}$ for all $(x, t) \in U$.*

In other words, away from discontinuities, the evolution operator $\widehat{\Psi}$ must be single-valued, with the value given by Ψ^H. This condition eliminates trivial approximations, such as taking $\widehat{\Psi}(x,t) = X$ for all $x \in X$, $t \in \mathbb{R}^+$. For maximum flexibility, we give no restrictions on the discontinuity set.

THEOREM 4.6 (uncomputability of the evolution of hybrid automata). *Let $\mathcal{H}$ be a class of hybrid automata. Then for any coherent semantics of evolution, the finite-time evolution of a hybrid automaton is uncomputable. This result holds even if we restrict our attention to (x,t)-values for which no event occurs at time t.*

In particular, the operator $(x_0,t) \mapsto \Psi^H(x_0,t)$ is not $(\rho_X, \rho_{\mathbb{R}^+}; \kappa_X)$-computable. Further, even if no event is possible at time t, the operator $x \mapsto \Psi^H(x,t)$ is not in general $(\rho_X; \kappa_X)$-computable.

The result is immediate from the following general lemma, since from Examples 4.2, 4.3, and 4.4 the evolution may have unremovable discontinuities, even away from discrete events. For a function $f : U \to Y$ where $U \subset X$, a *single-valued extension* of f over X is a function $\hat{f} : X \to Y$ such that $\hat{f}(x) = f(x)$ for all $x \in U$. A *multivalued extension* of f over X is a multivalued function $\widehat{F} : X \rightrightarrows Y$ such that $\widehat{F}(x) = \{y\}$ for all $x \in U$.

LEMMA 4.7. *Let $f : U \to Y$ be single-valued and continuous on an open, dense subset U of X, and let Y be compact. Suppose f has no continuous single-valued extension over X. Then f has no continuous multivalued extension $\widehat{F}$ over X.*

Proof. Since f has no continuous single-valued extension over X, there exists $x \in X \setminus U$ such that every extension of f over X is discontinuous at x. Let $A = \bigcap_{V \ni x} \mathrm{cl}(f(V \cap U))$. Suppose $\widehat{F}(x) \subset\neq A$, let $y \in A \setminus \widehat{F}(x)$, and take a closed neighborhood B of y such that $A \cap B = \emptyset$. Then $\widehat{F}^{-1}(B)$ does not contain x but contains points arbitrarily close to x, so $\widehat{F}$ would not be upper-semicontinuous. Suppose $\widehat{F}(x) \supset A$ and that A has two distinct elements y and z. Let W be an open neighborhood of y such that $\mathrm{cl}(W)$ is disjoint from z. Then $\widehat{F}^{-1}(W)$ contains x but does not contain points in $\widehat{F}^{-1}(X \setminus \mathrm{cl}(W))$, which come arbitrarily close to x, so $\widehat{F}$ is not lower-semicontinuous. ∎

4.4. Sliding along switching boundaries. A particularly nasty form of discontinuity occurs when a solution slides along the boundary of a guard set before crossing.

Example 4.8 (discontinuity caused by sliding). Consider a hybrid automaton in two dimensions with a guard set $y \geqslant 0$. Consider the flow $\dot{x} = 1$, and

$$\dot{y} = \begin{cases} a + 3x^2 - y \text{ if } x \leqslant 0; \\ a - y \text{ if } 0 \leqslant x \leqslant b; \\ a + 3(x-b)^2 - y \text{ if } x \geqslant b. \end{cases}$$

For $a = 0$, let (x_0, y_0) be a point with $x_0 < 0$ such that the continuous orbit starting at (x_0, y_0) exactly reaches the point $(0,0)$. Then for $b > 0$, the continuous evolution starting at (x_0, y_0) slides along the surface $y = 0$ for $0 \leqslant x \leqslant b$, and then crosses into $y > 0$. The hybrid orbit therefore undergoes a discrete transition at the point $(0,0)$ with $0 \leqslant x \leqslant b$. For $a > 0$, we see that $\dot{y} > 0$ when $y = 0$, and the orbit starting at (x_0, y_0) undergoes a discrete transition with $x < 0$, whereas for $a < 0$, the orbit starting at (x_0, y_0) undergoes a discrete transition with $x > b$. Hence the spacial evolution is discontinuous at the parameter value $a = 0$. Since for a lower-approximation to the solution we may consider only solutions which persist under

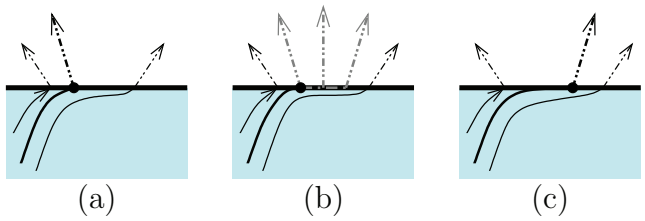

FIG. 6. *Sliding along a guard set. The discontinuity in* (b) *can be perturbed to give a continuous system in* (a) *and* (c)*, but the evolution of the original discontinuity point depends on the perturbation.*

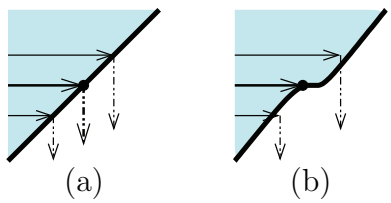

FIG. 7. *A C^0-perturbation of the guard set at a transverse crossing* (a) *can result in sliding at any point* (b)*. The evolution cannot be continued, since we cannot rule out the presence of sliding.*

perturbations, the hybrid evolution starting at (x_0, y_0) cannot be continued past the point $(0, 0)$. See Figure 6.

Now consider the case where $a = 0$ and $b = 0$, which is the limit of the cases $a = 0$ and $b > 0$. Since the hybrid orbit starting at (x_0, y_0) is blocked at $(0, 0)$ for $b > 0$, the orbit must also be blocked at $(0, 0)$ in the limit $b = 0$ when computing lower-approximations.

Hence for this parametrized family of systems, we cannot compute the evolution past the crossing point for the initial condition (x_0, y_0), even for the case $a = b = 0$ for which the dynamics in this case is given by the differential equation $(\dot{x}, \dot{y}) = (1, 3x^2 - y)$ and all crossings are topologically transverse.

The above example shows that topological transversality of crossing a guard set is not in itself sufficient to ensure that a discrete transition is enabled at the crossing point. However, even with differentially transverse crossings we need to be careful if we allow C^0-perturbations.

Example 4.9 (sliding under C^0-perturbation). Now consider the flow $(\dot{x}, \dot{y}) = (1, 0)$, the guard set $x = y$, and reset map $(x, y, q_0) \mapsto (y, q_1)$. The flow is transverse to the guard set, and if the initial state is (x, c, q_0) with $x < y$, then after the first reset the new state is (c, q_1). However, it is possible to make a C^0-perturbation of the guard set, so that the flow is parallel to the guard set for $y = a$. See Figure 7.

By the discussion following Example 4.8, this means that the evolution of the perturbed hybrid automaton cannot undergo a discrete transition for initial conditions with $y_0 = a$. Since we wish to compute lower approximations to the flow which persist under perturbation, this means that the evolution of the original hybrid automaton cannot undergo a discrete transition if $y_0 = a$. Since the above argument holds for arbitrary a, the evolution of the original hybrid automaton cannot undergo a discrete transition at any point of the guard set.

We have therefore demonstrated that, at least without additional information on

the behavior, the evolution of a hybrid automaton cannot be allowed to undergo a discrete transition at a crossing of a guard set if we are to compute lower-approximations to the evolution which are robust with respect to C^0-perturbations. However, the above situation is pathological in the sense that "most" hybrid systems do not exhibit this kind of sliding behavior. Further, transverse crossings are generic for hybrid automata with differentiable flows (such as from a Lipschitz differential equation) and differentiable guard sets, and in the C^1 topology on the guard condition and flow, transverse crossings cannot be perturbed away. This suggests that this pathological behavior can be treated numerically by computing derivatives, and this is indeed the case. However, trajectories which slide along the guard set can occur even in C^r flows with C^r guard sets in the neighborhood of a C^r singularity, and such singularities occur generically in r-dimensional hybrid automata. Hence, even taking higher-order derivatives might not be enough in some cases.

In this paper, we resolve the difficulty by giving a topological definition of a "detectable" crossing (which is weaker than topological transversality and allows tangencies) and show that if we restrict our attention to systems with detectable crossings, then it is possible to compute the evolution. It is possible to prove that crossings are detectable numerically by computing derivatives of the flow and guard set.

In the above example, a discontinuity in the evolution resulting in a loss of lower-semicomputability can occur at a degree-d crossing if perturbations of order $d-1$ are allowed. Hence, a purely topological approach to lower-approximations in systems with crossings of guard sets is bound to fail.

4.5. Urgent transitions. In approaches to hybrid systems modeling using only invariants (or progress predicates) and (nonurgent) guards, a transition which occurs as soon as the value of a variable x increases to a threshold value c is modeled using the invariant $x \leqslant c$ and the guard $x \geqslant c$. Syntactically, this is no different from an invariant $x \leqslant a$ and a guard $x \geqslant b$, where a and b have the same value c. This causes a discontinuity in the evolution with respect to the system parameters.

Example 4.10 (uncomputability caused by aliasing). Consider a system with initial condition $x(0) = x_0$, dynamic $\dot{x} = 1$, invariant $x \leqslant a$, and guard $x \geqslant b$ with $a, b > x_0$. If $a < b$, then the invariant is violated before the transition is activated, and further evolution is blocked. If $a > b$, then the transition is activated before the invariant is violated, and a transition may provably occur at any time $b < x(t) < a$. If $a = b$ and we are computing an over-approximation to the evolution, then a transition must occur exactly when $x(t) = a$, or equivalently, $x(t) = b$. However, if we are computing a lower-approximation to the evolution, we need to consider the possibility that a is slightly less than b. Hence a lower-approximation to the evolution must block, since this is the worst-case scenario. See Figure 8.

At first sight, it may seem that the evolution "should" continue for $a = b$. However, when considering lower-approximation, we cannot prove that continuation is possible since equality of a and b is undecidable. From the modeling viewpoint, if the invariant (or progress condition) and guard are determined by independent parameters, then it is only a coincidence that the transition is activated at exactly the same point as the continuous evolution is prevented, and under a small change in the parameters, the evolution may be blocked. It is only when we give the additional, *combinatorial* information that the invariant and guard boundaries lie exactly at the same point, and that we can deduce that the evolution may continue. This combinatorial information can be given by specifying that the event is urgent, which implicitly introduces an invariant for the continuous evolution. From an implementation stand-

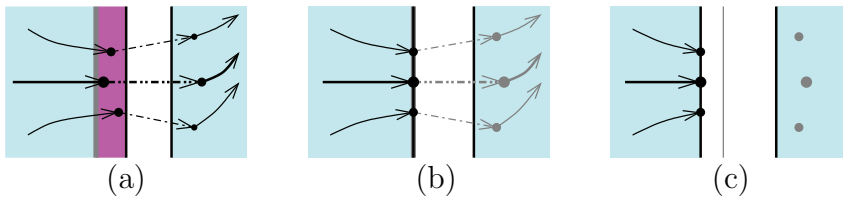

FIG. 8. *Discrete transitions may be blocked even at a transverse crossing. In* (a) *the invariant and guard regions overlap and crossings are possible. In* (b) *the boundaries of the invariant and guard regions touch, and discrete transitions are forced with upper semantics but disallowed using inner semantics. An arbitrarily small perturbation gives* (c) *in which no transitions are possible.*

point, we see that $x \leqslant a$ and $x \geqslant b$ are aliases for the same constraint $x \lesseqgtr c$ with $c = a = b$.

5. Semicontinuity of evolution of hybrid automata. In this section we state and prove the main results on semicomputability of system evolution. We first consider the case of upper-semicomputability, since this is more straightforward, and then consider lower-semicomputability. Recall that we consider the class of hybrid automata given by Definition 2.1 with the semantics given by Definition 2.4, and wish to compute the evolution operator Ψ^H of a hybrid automaton H as given in Definition 2.6.

In order to compute upper- or lower-approximations to the solution, we need to convert the system into either upper- or lower-semicontinuous form. We can do this by regularizing the guard sets to be open or closed and regularizing the flows and resets to be upper- or lower-semicontinuous maps.

DEFINITION 5.1. *Let $F : X \rightrightarrows Y$ be a multivalued function. Define the following:*

- $\overline{F} = \bigcap\{\widehat{F} \mid \widehat{F}$ *is upper-semicontinuous and* $\widehat{F} \supset F\}$, *and*
- $\underline{F} = \bigcup\{\widetilde{F} \mid \widetilde{F}$ *is lower-semicontinuous and* $\widetilde{F} \subset F\}$.

An alternative definition of $\overline{F}$ is in terms of its graph; $\mathrm{graph}(\overline{F}) = \bigcap_{\epsilon>0} N_\epsilon(\mathrm{graph}(F))$. It is easy to show that if F locally takes precompact values (i.e., $\mathrm{cl}(F(\bar{I}))$ is compact for any compact $\bar{I}$), then $\overline{F}$ is compact-valued upper-semicontinuous, and that $\underline{F}$ is closed-valued lower-semicontinuous. Further, it is trivial that $\overline{F} = F$ if F is upper-semicontinuous, and that $\underline{F} = F$ if F is lower-semicontinuous.

Throughout this section, we shall denote the interior of a set D by D°.

In order to compute the continuous evolution of a hybrid system, we need to restrict our attention to the invariant domain. Given a multiflow Φ, a subset D, a time t, and point x, define

$$\mathrm{rs}(\Phi, D)(x,t) = \Phi|_t^D(x) = \{y \in X \mid \exists \xi \in \Phi(x) \text{ s.t. } \xi([0,t]) \subset D \text{ and } \xi(t) = y\}. \tag{15}$$

For a trajectory η, define $\mathrm{dm}(\eta, D) = \{t \in \mathbb{R}^+ \mid \eta([0,t]) \subset D\}$ and $\mathrm{rs}(\eta, D)(t) = \eta|^D(t) = \eta(\{t\} \cap \mathrm{dm}(\eta, D))$; note that $\eta|^D(t) = \{\eta(t)\}$ if $\eta([0,t]) \subset D$ and $\eta|^D(t) = \emptyset$ otherwise.

5.1. Upper-semicomputability of the evolution and closure semantics. In this section, we consider conditions under which the evolution of a hybrid system is upper-semicomputable, i.e., the values of the evolution $\Psi^H(x_0, t)$ are $\kappa_>$-computable as compact sets. We observe that there are two main obstructions to upper-semicomputability:

1. The data describing the system need not be of the form required for computing over-approximations. A minimal requirement is that the guards and invariants are closed, and the dynamic and resets are compact-valued upper-semicontinuous.
2. An urgent transition occurs as soon as the guard is satisfied, even if this is a tangential contact with the guard set and so cannot be effectively detected.

In order to overcome these obstructions, we define a new *semantics of evolution.*

DEFINITION 5.2 (closure semantics). *An execution of the hybrid automaton* $H = (E, X, I, F, P, R_e, G_e, E_U)$ *with the* closure semantics *is a hybrid trajectory* $\xi : \mathcal{T} \to X$ *such that there exist events* $e_1, e_2, \ldots$ *with*

(CS1) $\xi(t, n) \in \overline{I}$ *whenever* $t \in [t_n, t_{n+1}]$,
(CS2) $\dot{\xi}(t, n) \in \overline{F}(\xi(t, x))$ *for almost every* $t \in [t_n, t_{n+1}]$,
(CS3) $\xi(t_n, n) \in \overline{R}_{e_n}(\xi(t_n, n-1))$,
(CS4) $\xi(t, n) \in \overline{P}$ *whenever* $t \in [t_n, t_{n+1}[$,
(CS5) $\xi(t_n, n-1) \in \overline{G}_{e_n}$, *and*
(CS6) $\xi(t, n) \notin G_u^\circ$ *whenever* $t \in [t_n, t_{n+1}[$ *and* $u \in E_U$.

Note that if I, P, and all G_e's are closed, and F and R_e are upper-semicontinuous, then the only change is in (CS6), where the condition $\xi(t, n) \notin G_u$ whenever $t \in [t_n, t_{n+1}[$ is replaced by $\xi(t, n) \notin G_u^\circ$ whenever $t \in [t_n, t_{n+1}[$ (equivalently, $\xi(t, n) \in \overline{X \setminus G_u}$).

If D is a closed set, then the condition $\xi(t, n) \in D$ whenever $t \in [t_n, t_{n+1}[$ is equivalent to $\xi(t, n) \in D$ whenever $t \in]t_n, t_{n+1}[$ and to the condition $t_n = t_{n+1}$ or $\xi(t, n) \in D$ whenever $t \in [t_n, t_{n+1}]$.

Combining results in Lemma 2.5, we see that the evolution of the hybrid system $H = (E, X, I, F, P, R_e, G_e, E_U)$ using closure semantics is the same as the evolution of the single-event system

$$\left(X, \overline{F}, \overline{I} \cap \overline{P} \setminus \bigcup_{e \in E_U} G_e^\circ, (\bigcup_{e \in E} \overline{R}_e|_{\overline{G}_e})|^{\overline{I}}, \bigcup_{e \in E} \overline{G}_e, \emptyset\right) \tag{16}$$

using standard semantics.

DEFINITION 5.3. *A hybrid automaton* $H = (E, X, I, F, P, R_e, G_e, \emptyset)$ *is* upper-semicontinuous *if*

1. I, P, *and all* G_e*'s are closed sets,*
2. F *and all* R_e*'s are upper-semicontinuous functions with compact values, and*
3. F *has linear growth.*

To prove upper-semicomputability using closure semantics, it suffices to show that the evolution is upper-semicomputable for a subclass of single-event hybrid automata. Upper-semicontinuity of the evolution for a similar class of hybrid automata was proved in [30]; the result given above gives upper-semicomputability as well as upper-semicontinuity.

THEOREM 5.4. *Let* $H = (X, I, F, P, R, G)$ *be an upper-semicontinuous single-event hybrid automaton. Then the evolution* Φ^H *using standard semantics is upper-semicomputable.*

More precisely, let $x_0 \subset X$ *be a compact initial state set,* $T \subset \mathbb{R}^+$ *a compact set of times, and* N *a bound on the number of events. Then the operator* $(I, F, P, R, G, x_0, t) \mapsto \Psi^H(x_0, t)$ *is* $(\psi_>, \mu_>, \psi_>, \mu_>, \psi_>, \rho, \rho; \kappa_>)$*-computable. Equivalently, the operator* $(I, F, P, R, G) \mapsto \Psi^H$ *is* $(\psi_>, \mu_>, \psi_>, \mu_>, \psi_>; \mu_>)$*-computable.*

The basic idea of the proof is to restrict the continuous dynamics to trajectories which remain in I, and the discrete dynamics to points in G. We encapsulate these parts into lemmas and give proofs in terms of the computable operations.

LEMMA 5.5. *Let $R : X \rightrightarrows Y$ be a compact-valued upper-semicontinuous map, and let $A \subset X$ be a closed set. Then the operator $(R, A) \mapsto R|_A$ is $(\mu_>, \psi_>; \mu_>)$-computable.*

Let $R : X \rightrightarrows Y$ be a compact-valued upper-semicontinuous map, and let $B \subset Y$ be a closed set. Then the operator $(R, B) \mapsto R|^B$ is $(\mu_>, \psi_>; \mu_>)$-computable.

Let $R_1, R_2 : X \rightrightarrows Y$ be compact-valued upper-semicontinuous maps. Then the operator $(R_1, R_2) \mapsto R_1 \cup R_2$, is $(\mu_>, \mu_>; \mu_>)$-computable.

Proof. For any compact set, $R|_A(C) = R(C \cap A)$. Given a $\psi_>$-name of A and a $\kappa_>$-name of C, we can compute a $\kappa_>$-name of $C \cap A$ by Theorem 3.6(4). Given a $\mu_>$-name of R and a $\kappa_>$-name of $C \cap A$, we can compute a $\kappa_>$-name of $R(C \cap A)$ by Theorem 3.9(6).

For any $x \in X$, $R|^B(x) = R(x) \cap A$. Given a $\mu_>$-name of R and a ρ-name of x, we can compute a $\kappa_>$-name of $R(x)$ by Theorem 3.9(6). Given a $\psi_>$-name of B and a $\kappa_>$-name of $R(x)$, we can compute a $\kappa_>$-name of $R(x) \cap B$ by Theorem 3.6(4).

For any compact set C, $(R_1 \cup R_2)(C) = R_1(C) \cup R_2(C)$. By Theorem 3.9(6) we can compute $\kappa_>$-names of $R_1(C)$ and $R_2(C)$. By Theorem 3.6(4), we can compute a $\kappa_>$-name of $R_1(C) \cup R_2(C)$. ∎

LEMMA 5.6. *Let Φ be an upper-semicontinuous compact-valued flow, and let D be a closed set. Then the operator $(\Phi, D) \mapsto (\Phi|^D)$ is $(\phi_>, \psi_>; \mu_>)$-computable.*

Proof. For $\eta \in C(\mathbb{R}^+; X)$, the set $\mathrm{dm}(\eta, D) = \{t \in \mathbb{R}^+ \mid \eta([0, t]) \subset D\}$ is $\psi_>$-computable given a γ-name of η and a $\psi_>$-name of D by Theorem 3.9(3) since we can compute a $\psi_<$-name of $[0, t]$ from a $\rho_{\mathbb{R}^+}$-name of t, and hence a $\psi_<$-name of $\eta([0, t])$. Since $\mathrm{rs}(\eta, D)(t) = \eta(\{t\} \cap \mathrm{dm}(\eta, D))$, we can compute a $\kappa_>$-name of $\mathrm{rs}(\eta, D)(t)$, since we can compute a $\kappa_>$-name of $\{t\}$ from a ρ-name of t by Theorem 3.6(7), a $\kappa_>$-name of $\{t\} \cap \mathrm{dm}(\eta, D)$ by Theorem 3.6(4), and a $\kappa_>$-name of $\eta(\{t\} \cap \mathrm{dm}(\eta, D))$ by Theorem 3.6(8).

Hence the function $(\eta, D, t) \mapsto \eta|^D(t)$ is $(\gamma, \theta_<, \rho; \kappa_>)$-computable. By Proposition 3.12(2), we can compute a $\kappa_>$-name of $\Phi|_t^D(x) = \bigcup\{\eta|^D(t) \mid \eta \in \Phi(x)\}$. ∎

Proof of Theorem 5.4. Define $R|_G^I : X \rightrightarrows X$ by $R|_G^I(x) = R(\{x\} \cap G) \cap I$. By Lemma 5.5, we can compute a $\mu_>$-name of $R|_G^I$ from a $\mu_>$-name of R and $\psi_>$-names of I and G. By Lemma 5.6, we can obtain a $\mu_>$-name of the restricted flow evolutions $\Phi|_t^{P \cap I}$. It remains to compute a $\mu_>$-name of the evolution $\Psi = \Psi^H$.

Define

$$\Psi(x, t, (t_1, \ldots, t_n)) = \{y \in X \mid \exists \text{ execution } \xi \text{ of } H \text{ with event times } t_1 \leqslant \cdots \leqslant t_n \leqslant t \text{ s.t. } \xi(0, 0) = x \text{ and } \xi(t, n) = y\}.$$

Then since we can write

$$\Psi(x, t, (t_1, \ldots, t_n)) = \Phi|_{t-t_n}^{P \cap I} \circ R|_G^I \circ \Phi|_{t_n - t_{n-1}}^{P \cap I} \circ \cdots \circ R|_G^I \circ \Phi|_{t_1}^{P \cap I}(x),$$

we see that the map $(x, t, (t_1, \ldots, t_n)) \mapsto \Psi(x, t, (t_1, \ldots, t_n))$ is a composition of functions for which we have $\mu_>$-names, and hence we can compute $\mu_>$-name of $\Psi(x, t)$. We can write $\Psi(x, t, n) = \Psi(x, t, T_{t,n})$ where $T_{t,n} = \{(t_1, \ldots, t_n) \in \mathbb{R}^n \mid \text{for all } i,\ 0 \leqslant t_i \leqslant t_{i+1} \leqslant t\}$. Since $t \mapsto T_{t,n}$ is $(\rho, \kappa_>)$-computable, we can compute a $\mu_>$-name of the function $(x, t, n) \mapsto \Psi(x, t, n)$. Then $\Psi(x, t, [0, N]) = \bigcup_{n=0}^{\infty} \Psi(x, t, n) = \bigcup_{n=0}^{N} \Psi(x, t, n)$ is a finite union of $\mu_>$-computable functions. Hence we can compute a $\mu_>$-name of Ψ. ∎

We say a system is *uniformly non-Zeno* if there exist (T, N) such that for any execution, there occur at most N discrete events in any time interval of length at most

T. As shown in [30], any non-Zeno upper-semicontinuous hybrid automaton with a compact global attractor must be uniformly non-Zeno. For non-Zeno systems, we can drop the bounds on the number of events.

COROLLARY 5.7 (upper-semicomputability for non-Zeno hybrid automata). *Let $H = (E, X, I, F, P, R_e, G_e, E_U)$ be a hybrid automaton which is uniformly non-Zeno using closure semantics. Let $X_0 \subset I$ be a compact initial state set, and let $T \subset \mathbb{R}^+$ be a compact set of times. Then the operator $(I, F, P, R_e, G_e, X_0, t) \mapsto \Psi^H(X_0, T)$ is $(\psi_>, \mu_>, \psi_>, \mu_>, \psi_>, \kappa_>, \rho; \kappa_>)$-computable.*

By results in [20], the smallest compact-valued upper-semicomputable over-approximation to a multivalued map $R : X \rightrightarrows Y$ is $\overline{R}$, and the smallest compact-valued upper-semicomputable over-approximation to the flow Φ of $F : X \rightrightarrows X$ is the flow of $\overline{F}$, assuming F has linear growth. We now extend these results to hybrid systems and show that $\overline{H}$ is a "smallest" hybrid automaton for which the evolution is upper-semicomputable. In other words, *any* attempt to compute an over-approximation to the evolved set using approximative methods must necessarily compute an over-approximation to the evolved set of $\overline{H}$.

THEOREM 5.8. *Let $H = (E, X, I, F, P, R_e, G_e, \emptyset)$ be a uniformly non-Zeno hybrid automaton, and suppose that $\hat{\Psi} : X \times \mathbb{R} \mapsto \mathcal{K}(X)$ is upper-semicomputable and $\hat{\Psi}(x, t) \supset \Psi^H(x, t)$ for all x, t. Then $\hat{\Psi}(x, t) \supset \Psi_{\overline{H}}(x, t)$.*

Proof. Suppose $y_k \in \Psi^H(x_k, t_k)$ and $x_k \to x_\infty$, $t_k \to t_\infty$, and $y_k \to y_\infty$ as $k \to \infty$. Then there exist executions ξ_n of H and $n_k \in \mathbb{N}$ such that $\xi_k(t_k, n_k) = y_k$ for all k.

Since $\hat{\Psi}$ is upper-semicontinuous, we must have $y_\infty \in \hat{\Psi}(x_\infty, t_\infty)$. Further, by restricting our attention to a subsequence, we can ensure that all ξ_k have the same events e_i up to time t_k, that the ith-event times $t_{k,i}$ converge to $t_{\infty,i}$, and, by boundedness of the R_e and linear growth of F, that the ξ_k converge to ξ_∞.

Then since $\xi_k(t, i) \in I$ and $\xi_\infty(\cdot, i) = \lim_{k\to\infty} \xi_k(\cdot, i)$, we must have $\xi_\infty(t, i) \in \overline{I}$ for all $t \in [t_{\infty,i}, t_{\infty,i+1}]$. If $t \in [t_i, t_{i+1}[$, then since $\xi_k(t, i) \in P$, we must have $\xi_\infty(t, i) \in \overline{P}$ for all $t \in [t_{\infty,i}, t_{\infty,i+1}]$. Since $\xi_k(t_{k,i}, i-1) \in G_{e_i}$ and $\xi_\infty(t_{\infty,i}, i-1) = \lim_{k\to\infty} \xi_k(t_{k,i}, i-1)$, we must have $\xi_\infty(t_{\infty,i}, i-1) \in \overline{G}_{e_i}$. Since $\xi_k(t_{k,i}, i) \in R_{e_i}(\xi_k(t_{k,i}, i-1))$ for all k, we have $\xi_\infty(t_{\infty,i}, i) \in \overline{R}_{e_i}(\xi_\infty(t_{\infty,i}, i-1))$. Finally, using methods similar to those of [20], we can show that $\dot{\xi}_\infty(t, i) \in \overline{F}(\xi_\infty(t, i))$ a.e. Hence $y_\infty \in \Psi_{\overline{H}}(x_\infty, t_\infty)$. ∎

5.2. Lower-semicomputability of evolution. We now define a new semantics of evolution, *interior semantics*, and show that the evolution is lower-semicomputable using this semantics.

DEFINITION 5.9 (interior semantics). *An* execution *of the hybrid automaton $H = (E, X, F, I, R_e, G_e, U)$ with the* interior semantics *is a hybrid trajectory $\xi : \mathcal{T} \to X$ such that there exist events $e_1, e_2, \ldots$ with*

(IS1) $\xi(t, n) \in I^\circ$ whenever $t \in [t_n, t_{n+1}]$,
(IS2) $\dot{\xi}(t, n) \in \underline{F}(\xi(t, x))$ for almost every $t \in [t_n, t_{n+1}]$,
(IS3) $\xi(t_n, n) \in \underline{R}_{e_n}(\xi(t_n, n-1))$,
(IS4) if $t_{n+1} > t_n$, then $\xi(t, n) \in P^\circ$ whenever $t \in [t_n, t_{n+1}]$,
(IS5) $\xi(t_n, n-1) \in G^\circ_{e_n}$, and
(IS6) if $t_{n+1} > t_n$, then $\xi(t, n) \notin \overline{G}_u$ whenever $t \in [t_n, t_{n+1}]$ and $u \in E_U$.

Note that if I, P, and all G_e's are open, and all the F and R_e are lower-semicontinuous, then the only changes are in (IS4), where the condition $\xi(t, n) \in P$ to $t \in [t_n, t_{n+1}[$ is replaced by $t_n = t_{n+1}$ or $\xi(t, n) \in P^\circ$ for $t \in [t_n, t_{n+1}]$, and in (IS6), where the condition $\xi(t, n) \notin G_u$ whenever $t \in [t_n, t_{n+1}[$ is replaced by $t_n = t_{n+1}$ or $\xi(t, n) \notin G^\circ_u$ whenever $t \in [t_n, t_{n+1}]$. Unfortunately, this latter change causes a

difficulty, since G_u° and $(X \setminus G_u)^\circ$ are disjoint open sets, so a trajectory cannot cross from $(X \setminus G_u)^\circ$ to G_u°. For this reason, interior semantics is only really useful for systems with no urgent events. We shall return to this difficulty in section 4.5 and give an improved semantics in section 5.3.

Combining with Lemma 2.5, we see that the evolution of the hybrid automaton $H = (E, X, I, F, P, R_e, G_e, \emptyset)$ using interior semantics is the same as the evolution of the single-event hybrid automaton

$$\left(X, \underline{F}, P^\circ \cap I^\circ, (\textstyle\bigcup_{e\in E} \underline{R}_e|_{G_e^\circ})|^{I^\circ}, \bigcup_{e\in E} G_e^\circ\right) \tag{17}$$

using standard semantics.

Definition 5.10. *A hybrid automaton $H = (E, X, I, F, P, R_e, G_e, \emptyset)$ is* lower-semicontinuous *if*

- *I, P, and all G_e's are open sets,*
- *F and all R_e's are lower-semicontinuous with closed values, and*
- *F is one-sided-Lipschitz with convex values.*

In this situation, we have the following computability result.

Theorem 5.11. *The evolution of a lower-semicontinuous hybrid automaton is lower-semicomputable.*

More precisely, let $H = (X, I, F, P, R, G)$ be a single-event hybrid automaton where I, P, and G are open, F defines a lower-semicontinuous multivalued flow Φ, and the $R : X \rightrightarrows X$ are lower-semicontinuous. Let $x_0 \in X$ be closed, and let $t \in \mathbb{R}^+$. Then the operator $(I, F, P, R, G, x_0, t) \mapsto \mathrm{cl}(\Psi^H(x_0, t))$ is $(\theta_<, \mu_<, \theta_<, \mu_<, \theta_< \rho, \rho; \psi_<)$-computable.

The proof is very similar to that of Theorem 5.4.

Lemma 5.12. *Let $R : X \rightrightarrows Y$ be a closed-valued lower-semicontinuous map, and let $U \subset X$ and $V \subset Y$ be open sets. Then the operators $(R, U) \mapsto R|_U$ and $(R, U) \mapsto \mathrm{cl}(R|^V)$ are $(\mu_<, \theta_<; \mu_<)$-computable.*

Proof. We have $\bar{I} \subset R|_U^{-1}(J)$ if, and only if, $\bar{I} \subset U \cap R^{-1}(J)$. Since J is basic, we have access to a $\theta_<$-name of J. By Theorem 3.9(3) we can compute a $\theta_<$-name of $R^{-1}(J)$ from a $\mu_<$-name of R and a $\theta_<$-name of J. By Theorem 3.6(1), we can compute a $\theta_<$-name of $U \cap R^{-1}(J)$. Since $\bar{I}$ is basic, we have access to a $\kappa_>$-name of $\bar{I}$. Since, given a $\kappa_>$-name of compact C and a $\theta_<$-name of open U, we can verify $C \subset U$, we can enumerate all (I, J) such that $\bar{I} \subset R|_U^{-1}(J)$.

Given a ρ-name of x, we can compute a $\psi_<$-name of $R(x)$ by Theorem 3.9(1), and hence a $\psi_<$-name of $\mathrm{cl}(R(x) \cap V)$ by Theorem 3.6(1). ∎

Lemma 5.13. *Let Φ be a lower-semicontinuous closed-valued flow, and let D be an open set. Then the operator $(\Phi, D, t) \mapsto \Phi|_t^D$ is $(\phi_<, \theta_<, \rho; \mu_<)$-computable.*

Proof. We first claim that the operator $(\eta, D) \mapsto \mathrm{dm}(\eta, D)$ is $(\gamma, \theta_<; \theta_<)$-computable. Recall $\mathrm{dm}(\eta, D) = \{t \in \mathbb{R}^+ \mid \eta([0, t] \subset D\}$. Since the operator $t \mapsto [0, t]$ is $(\rho; \kappa_>)$-computable, by Theorem 3.6(8), the operator $(\eta, t) \mapsto \eta([0, t])$ is $(\gamma, \rho; \kappa_>)$-computable. By Theorem 3.9(2,3), the operator $(\eta, D) \mapsto \{t \mid \eta([0, t]) \subset D\}$ is $(\gamma, \theta_<; \theta_<)$-computable. By Theorem 3.6(3), the operator $(\eta, D, t) \mapsto \{t\} \cap \{s \mid \eta([0, s]) \subset D\}$ is $(\gamma, \theta_<, \rho; \psi_<)$-computable. By Theorem 3.6(8), the operator $(\eta, D, t) \mapsto \eta(\{t\} \cap \sigma(\eta, D))$ is $(\gamma, \theta_<, \rho; \psi_<)$-computable. By Proposition 3.12(2), the operator $(\Phi, D, t, x) \mapsto \Phi|_t^D(x)$ is $(\gamma, \theta_<, \rho, \rho; \psi_<)$-computable. Hence the operator $(\Phi, D, t) \mapsto \Phi|_t^D$ is $(\phi_<, \theta_<, \rho; \mu_<)$-computable. ∎

Proof of Theorem 5.11. Define $R|_G^I : X \rightrightarrows X$ by $R|_G^I(x) = \mathrm{cl}(R(\{x\} \cap G) \cap I)$. By Lemma 5.12, we can compute a $\mu_<$-name of $R|_G^I$ from a $\mu_<$-name of R and $\theta_<$-name

of G. By Lemma 5.13, we can obtain a $\mu_<$-name of the restricted flow evolutions $\Phi|_t^{P\cap I}$. It remains to compute a $\mu_<$-name of the evolution $\Psi = \Psi^H$.

Define

$$\Psi(x,t,(t_1,\dots,t_n)) = \{y \in X \mid \exists \text{ execution } \xi \text{ of } H \text{ with event times } t_1 \leqslant \cdots \leqslant t_n \leqslant t \text{ s.t. } \xi(0,0) = x \text{ and } \xi(t,n) = y\}.$$

Then since we can write

$$\Psi(x,t,(t_1,\dots,t_n)) = \Phi|_{t-t_n}^{P\cap I} \circ R|_G^I \circ \Phi|_{t_n-t_{n-1}}^{P\cap I} \circ \cdots \circ R|_G^I \circ \Phi|_{t_1}^{P\cap I}(x)$$

we see that the map $(x,t,(t_1,\dots,t_n)) \mapsto \Psi(x,t,(t_1,\dots,t_n))$ is a composition of functions for which we have $\mu_<$-names, and hence we can compute a $\mu_<$-name of $\Psi(x,t,(t_1,\dots,t_n))$. We can write $\Psi(x,t,n) = \Psi(x,t,T_{t,n})$, where $T_{t,n} = \{(t_1,\dots,t_n) \in \mathbb{R}^n \mid \text{for all } i,\ 0 \leqslant t_i \leqslant t_{i+1} \leqslant t\}$. Since $t \mapsto T_{t,n}$ is $(\rho,\psi_<)$-computable, we can compute a $\mu_<$-name of the function $(x,t,n) \mapsto \Psi(x,t,n)$. Then $\Psi(x,t,[0,N]) = \bigcup_{n=0}^{\infty} \Psi(x,t,n) = \bigcup_{n=0}^{N} \Psi(x,t,n)$ is a countable union of $\mu_<$-computable functions. Hence we can compute a $\mu_<$-name of Ψ. ∎

Similar to the case of upper-semicomputability, we have an optimality result for lower-semicomputability.

THEOREM 5.14. *Let $H = (E,X,I,F,P,R_e,G_e,E_U)$ be a hybrid automaton with $E_U = \emptyset$, and suppose that $\Psi : X \times \mathbb{R} \mapsto \mathcal{A}(X)$ is lower-semicomputable and $\Psi(x,t) \subset \mathrm{cl}(\Psi^H(x,t))$ for all x,t. Then $\Psi(x,t) \subset \mathrm{cl}(\Psi_{\underline{H}}(x,t))$.*

The proof is similar to that of Theorem 5.8 and is omitted.

5.3. Lower-semicomputability of hybrid automata using crossing semantics. Let Φ be a lower-semicontinuous multiflow, and suppose B, A, and D are open sets with $\overline{D} = \overline{B \cup A}$ and that Φ is closed-valued lower-semicontinuous. We would like to know when trajectories of Φ cross instantaneously from B to A within D.

DEFINITION 5.15. *A continuous trajectory η crosses from B to A at time t and point x in D if $\eta(t) \in D$ and for all $\delta > 0$, $\eta(]t-\delta,t[) \cap B \neq \emptyset$ and $\eta(]t,t+\delta[) \cap A \neq \emptyset$. We say that x is a* crossing point *for η.*

Note that trivially if $x \in B \cap A$, then x is a crossing point for any trajectory through it. If $x \notin \mathrm{cl}(B) \cap \mathrm{cl}(A)$, then x cannot be a crossing point. The requirement that $x \in D$, which implies $x \in (\overline{B \cup A})^\circ$, is to ensure that perturbations of η remain in $\overline{B \cup A}$ near x.

The real interest is when x lies in ∂B and ∂A. By the observations of Example 4.10, if B and A are disjoint, then by a small perturbation, we can make their boundaries disjoint, and so any lower approximation to the flow has a blocking trajectory. We therefore need more information about the sets B and A given by a $\theta_<$-name.

Let us first consider the case in which B and A form a topological partition of X; that is, $B \cap A = \emptyset$ and $\mathrm{cl}(B) \cup \mathrm{cl}(A) = X$. Suppose η is a trajectory such that $\eta(t_1) \in B$ and $\xi(t_2) \in A$ for $t_1 < t_2$, so η leaves B and enters A at some time $t \in]t_1,t_2[$. We would like to be able to deduce that η crosses from B to A in the sense of Definition 5.15. Unfortunately, from Example 4.8, it may be the case that η slides inside $\partial B \cap \partial A$ rather than crossing transversely, and, as we have seen, we cannot handle sliding solutions.

DEFINITION 5.16. *Let Φ be a flow, B and A be open sets, and $\delta > 0$. We say Φ has* δ-detectable crossings *if for all trajectories η of Φ such that $\eta(0) \in B$ and*

$\eta(t) \in A$ for some $t < \delta$, there exists $c \in \mathbb{R}$ and $\zeta \in \Phi$ such that $\zeta(t) = \eta(t)$ for $t \in [0, c]$, $\zeta([0, c[) \subset B$, $\zeta(c) \in (\overline{B \cup A})^\circ$, and for all $\epsilon > 0$, $\zeta(]c, c+\epsilon[) \cap A \neq \emptyset$.

We say Φ has effectively *detectable crossings if it has δ-detectable crossings for some* known *δ.*

In other words, if there is a trajectory η which moves from B to A in time less than δ, then from the point where the state leaves B, there is a possibly different trajectory ζ which immediately enters A. Note that the condition of detectable crossings precludes the degenerate situation in which a solution slides along a common boundary of B and A for time less than δ, and also the case of Example 4.10 in which the solution leaves B briefly before entering A.

Remark 5.17. A sufficient condition for a single-valued flow ϕ to have detectable crossings of a manifold $G = \{x \in X \mid g(x) = 0\}$ is that ϕ and g are analytic. In this case, any trajectory either slides along G or touches/crosses at a single point.

We now define a new notion of solution for hybrid automata which allows the lower-approximation of the evolution for hybrid automata with urgent transitions.

DEFINITION 5.18. *An* execution *of the hybrid automaton $H = (E, X, F, I, R_e, G_e, U)$ with the* crossing semantics *is a hybrid trajectory $\xi : \mathcal{T} \to X$ such that there exist events $e_1, e_2, \ldots$ with*

(XS1) $\xi(t, n) \in \underline{I}$ whenever $t \in [t_n, t_{n+1}]$,
(XS2) $\dot{\xi}(t, n) \in \underline{F}(\xi(t, x))$ for almost every $t \in [t_n, t_{n+1}]$,
(XS3) $\xi(t_n, n) \in \underline{R}_{e_n}(\xi(t_n, n-1))$,
(XS4) if $t_n < t_{n+1}$, then $\xi(t, n) \in \underline{I}$ whenever $t \in [t_n, t_{n+1}]$,
(XS5) $\xi(t_n, n-1) \in G^\circ_{e_n}$ if $e_n \notin E_U$,
(XS6) $\xi(t, n) \notin \overline{G}_u$ whenever $t \in [t_n, t_{n+1}[$ and $u \in E_U$, and
(XS7) $\xi(t_n, n-1)$ is a crossing point of ∂G_{e_n} if $e_n \in E_U$.

Intuitively, between discrete events, solutions must remain in the interior of I, P, and $X \setminus G_u$ for $u \in E_U$; this prevents grazing contact with guard sets. A discrete event may occur at the boundary of G_u if it is possible to continue the trajectory instantaneously into $(X \setminus G_u)^\circ$.

Using this notion of solution, we can prove the following result.

THEOREM 5.19 (lower-semicomputability of the evolution of hybrid automata with detectable crossings). *Let H be a lower-semicontinuous hybrid automaton with δ-detectable crossings. Then the evolution $H \mapsto \Psi_H(x, t)$ is lower-semicomputable using crossing semantics.*

More precisely, let $H = (E, X, I, F, P, R_e, G_e, E_U)$ be a hybrid automaton where I, P, and all G_e's are open sets and F and R are lower-semicontinuous with closed values. Suppose that crossings of the flow $\Phi|^F$ from G°_u to $(X \setminus G_u)^\circ$ are detectable for each $u \in E_U$. Let X_0 be a closed set of initial states, and let T be an open set of times. Then the operator $(I, F, P, R_e, G_e, \overline{G}_u) \mapsto \mathrm{cl}\Psi^H$ is $(\theta_<, \mu_<, \theta_<, \mu_<^{|E|}, \theta_<^{|E|}, \psi_>^{|E_U|}; \mu_<)$-computable.

The main difference between the proof of Theorem 5.19 and that of Theorem 5.11 is that we cannot apply the reset R_u for an urgent event u at the correct time by restricting our attention to the crossing set, since this is a subset of ∂G_u and hence not open. We therefore have to compute the possible times and locations of the events for a given initial state directly. The simplest way of organizing the computation is to extend the state with the current time and perform an untimed reachability computation. For timed reachability, we can project onto the desired set of times.

We use the following lemma, which shows that the crossing points can be computed.

LEMMA 5.20. *Let D, A, and B be open sets such that $\overline{D} = \overline{A \cup B}$, and let Φ be a lower-semicontinuous closed-valued flow. Define the activation function $\Gamma = \Gamma(B, A, D, \Phi) : X \rightrightarrows X$ by*

(18)
$$\Gamma(x) = \{y \in X \mid \exists \eta \in \Phi(x),\ t \in \mathbb{R}^+ \text{ s.t. } \eta(t) = y,\ \eta([0,t[) \subset B, \eta(t) \in D, \text{ and } \eta(]t, t+\epsilon[) \cap A \neq \emptyset \text{ for all } \epsilon > 0\}.$$

Suppose that the crossings of trajectories of Φ from B to A in D are δ-detectable. Then the function $(B, A, D, \Phi) \mapsto \Gamma$ is $(\theta_<, \theta_<, \theta_<, \phi_<; \mu_<)$-computable.

Proof. It suffices to prove that $\Gamma(x)$ is $\psi_<$-computable given a ρ-name of x. We consider two cases, $t = 0$ and $t > 0$.

Suppose $\eta \in \Phi(x)$ and $t_1, t_2 \in \mathbb{Q}^+$ are such that $\eta([0, t_1]) \subset B$, $\eta([t_1, t_2]) \subset D$ and $\eta(t_2) \in A$ and $0 < t_1 < t_2 < t_1 + \delta$. Then since crossings from B to A are δ-detectable, there is a crossing of η at time $c \in (t_1, t_2)$. Conversely, if there is a crossing of η at time $t > 0$, then we can find such t_1, t_2.

Suppose $\eta \in \Phi(x)$ and $t_2 \in \mathbb{Q}^+$ are such that $\eta([0, t_2]) \subset D$ and $\eta(t_2) \in A$ and $0 < t_2 < \delta$. Then there is a crossing of η at time $c = \inf\{t \geqslant 0 \mid \eta([t, t_2]) \subset A\}$ and $c \in [0, t_2[$, which is open in $\mathbb{R}^+$. Conversely, if there is a crossing of η at time 0, then we can find such a t_2.

We can therefore compute all open rational intervals $]t_1, t_2[$ or $[0, t_2[$ containing crossing times of η, so the set of crossing times of η is $\psi_<$-computable in $\mathbb{R}^+$ by Definition 3.5. The set of crossing points of η is the image of the set of crossing times, so it is computable by Theorem 3.6(8). The set of crossing points starting at x is the union of all crossing points of curves in $\Phi(x)$, so it is computable by Proposition 3.12(2). ∎

We will use the above result with B and A disjoint, $D = (\overline{B \cup A})^\circ$ for urgent events, and $D = B \cup A$ for nonurgent events.

We can use Lemma 5.20 to prove computability of the *untimed* reachable set.

LEMMA 5.21. *Under the conditions of Theorem 5.19, the untimed reachable set $\Psi^H(X_0, \mathbb{R}^+)$ is $\psi_<$-computable.*

Proof. The set of states reachable starting at state x after event e is given by $R_e(\Gamma(B_e, A_e, D, \Phi)(x)) \cap I$, where $B_e = I \cap P \setminus \bigcup_{u \in E_U} \overline{G}_u$, $A_e = G_e \setminus \bigcup_{u \in E_U \setminus \{e\}} \overline{G}_u$, and $D = I \setminus \bigcup_{u \in E_U \setminus e} \overline{G}_u$. Since we can compute a $\mu_<$-name of $\Gamma(B_e, A_e, D, \Phi)$ by Lemma 5.20, we can compute a $\psi_<$-name of the set reachable after any event. The (untimed) reachable set is then

$$\widehat{\Psi}(X_0, \mathbb{R}^+) = \bigcup_{(e_1, \ldots, e_k) \in E^*} (\Phi|^D \circ R_{e_n} \circ \Gamma(F, D, A_{e_n}) \circ \cdots \circ R_{e_1} \circ \Gamma(F, D, A_{e_1}))(X_0)$$

which is $\psi_<$-computable, being a countable union of $\psi_<$-computable sets. ∎

Proof of Theorem 5.19. We first enhance the hybrid system by taking state space $\widehat{X} = X \times (\mathbb{R}^+ \sqcup \{\infty\})$, adding an explicit time variable τ, and yielding a new state $\hat{x}$. The new dynamic is $\hat{F}(x, t) = F(x) \times \{1\}$, the new resets are $\hat{R}_e(x, t) = R_e(x) \times \{t\}$, the new invariant is $\hat{I} = I \times \mathbb{R}$, the new progress condition is $\hat{P} = P \times \mathbb{R}$, and the new guards are $\hat{G}_e = G_e \times \mathbb{R}$. Further, we add a new urgent event $\hat{e}$ with guard $\widehat{G}_{\hat{e}} = \{(x, \tau) \mid \tau \geqslant t\}$ and reset $\widehat{R}_{\hat{e}}(x, \tau) = (x, \infty)$. When $\tau = \infty$, there is no further dynamics.

The result follows, since the reachable set of H at time t is simply the component of the reachable set of $\widehat{H}$ with $t = \infty$. ∎

Remark 5.22. The strategy of the proof of Theorem 5.19 could also be applied to prove Theorems 5.4 and 5.11, but at some cost in complexity.

6. Modeling, simulation, implementation, and control.

6.1. Modeling hybrid automata. The description of hybrid automata introduced in section 2 is sufficient to define the dynamic evolution but is inexpressive as a modeling framework. Many hybrid automaton models are not written in terms of invariant and guard *sets*, but using invariant and guard *predicates*. Further, the continuous evolution is usually written using differential equations, possibly with disturbance input variables, rather than differential inclusions. However, it is easy to convert from standard modeling frameworks to sets and differential inclusions.

Sets can be conveniently described using *constraint functions*. A *constraint* is a continuous function $c : X \to \mathbb{R}$, and we say a constraint is *regular* if $\{x \in X \mid c(x) = 0\}$ is a codimension-1 topological manifold. If c is differentiable and $\nabla c(x) \neq 0$ whenever $c(x) = 0$, then c is regular and changes sign on a differentiable manifold. A constraint defines sets $\{x \in X \mid c(x) \lesseqgtr 0\}$ and $\{x \in X \mid c(x) \lesseqgtr 0\}$. The operator $c \mapsto \{x \mid c(x) < 0\}$ is $(\gamma; \theta_<)$-computable, and the operator $c \mapsto \{x \mid c(x) \leqslant 0\}$ is $(\gamma; \psi_>)$-computable.

Differential inclusions can be conveniently described using differential equations with *bounded noise*. Let X be a subset of $\mathbb{R}^n$, and let V be a compact subset of $\mathbb{R}^p$. We can specify the continuous dynamics $\dot{x}(t) = f(x(t), v(t))$, where f is continuous and Lipschitz in x, and $v(t)$ is a measurable function from $\mathbb{R}^+$ to V. Using the standard Filippov solution concept, we can rewrite this system as $\dot{x}(t) \in F(x(t))$, where $F(x) = \mathrm{conv}(f(x, V))$. Then it is possible to compute a $\mu_>$-name of F from a γ-name of f and a $\kappa_>$-name of V.

We note that the semantics are only specified for a complete hybrid system model. In a compositional modeling framework, we also need to define models with inputs and consider parallel composition of different components.

6.2. Reliable simulation of hybrid automata. We wish to be able to reliably simulate the trajectory of a deterministic hybrid automaton starting at some initial point x. Away from discontinuity points in the spatial dependence of the evolution, the meaning of a simulation is clear; there is a unique trajectory, which we can compute using either upper or lower semantics. However, at the discontinuity points, there are at least two possible choices for how to continue the evolution; at a grazing or external corner collision point, we must choose between carrying on with the continuous dynamics or applying a discrete reset. At a point where multiple events are activated, we must choose between which of the two or more events occurs. Even near the discontinuity points, we may not be able to reliably distinguish which of the possible continuations occurs due to numerical error.

One way of resolving these different possibilities is to either make a random choice, or rank the possible events in some order and apply the preferred event. However, this runs the risk of missing qualitatively different evolutions. Another option is to continue with all possible different evolutions. This is feasible only if the discontinuity set is entered at a discrete set of time instances.

If crossings with the guard set $G = \partial D \cap \partial A$ are δ-detectable, then we can compute the set of grazing points as $G_0 = \{x \in G \mid \Phi(x, [0, \delta/2]) \subset D\}$, which is $\psi_<$-computable from Φ and D. Taking e_0 to be the special *grazing event* with guard G_0, we can compute the discontinuity set as the union of all intersections of pairs of guard sets. Hence the discontinuity set is $\bigcup_{(e_i,e_j)\in E\cup\{e_0\}} G_i \cap G_j$ and is $\psi_<$-computable from the system data.

We can think of simulation as computing the evolution Ψ of the system from a single initial point. In order to distinguish between the multiple possibilities at branching points, we need only store a list of event labels and times. Since we can compactify $\mathbb{R}^+$ by adding the point at infinity, the set $E \times (\mathbb{R}^+ \cup \{\infty\})$ is itself countably based and locally compact in the product topology. By extending the state variable with the time t, and the reset relation by updating the list of events whenever a reset occurs, we obtain a new hybrid automaton in which the evolution operator stores the sequence of discrete events and the total time used to reach a particular state, from which the entire trajectory can be completely reconstructed.

Hence by Theorem 5.4, we can compute the set of all possible distinct evolutions from a given initial point for a uniformly non-Zeno hybrid automaton.

6.3. Implementation issues. Throughout the paper, every effort has been made to present the minimal assumptions necessary in order to perform a computation. In particular, no assumptions on the differentiability of various objects were made. Further, the counterexamples to computability were all based on simple affine systems, so adding differentiability assumptions makes no difference in the ability to compute arbitrarily accurate approximations to the evolution. However, efficient numerical methods require differentiability assumptions on the inputs in order to obtain high-order convergence. Therefore, when implementing the operations involved, particularly the algorithms for computing system evolution and crossing of guard sets, it is important to use differentiability to obtain efficient algorithms. As an example, the crossing time to a transverse guard set can be computed to an order which is the maximum differentiability of the guard constraint and the flow. This can allow more efficient stepping over guard constraints than methods relying purely on checking for crossing using set inclusions.

The theory presented in this paper has been implemented in the tool Ariadne for reachability analysis of hybrid automata. Examples of computations performed using Ariadne can be found in [6].

6.4. Implications for control. The results of this paper are directly relevant to the analysis of hybrid systems with control inputs and indirectly to control design. The results on lower-semicomputability can be interpreted as controllability results if the nondeterminism in the continuous dynamics, the switching, and the resets describe possible control inputs. In this case, the reachable set $\underline{\Psi}^H(x_0, \mathbb{R}^+)$ represents the set of points y for which we can prove the existence of a solution of H from x_0 to y. Further, since $\underline{\Psi}^H$ is lower-semicomputable, if $T \subset X$ is an open set of *target* points, then by Theorem 3.9(3), the set $\{x_0 \in X \mid \underline{\Psi}^H(x_0, \mathbb{R}^+) \cap T \neq \emptyset\}$ is computable and is the *provably controllable* set to T. See [21] for more details. The results on upper-semicomputability can correspondingly be used to show that a control problem is *unsolvable*.

Where the nondeterminism in the system description represents the effect of environmental disturbances or noise, the upper-semicomputability results show that the closure semantics is the appropriate semantics for proving correctness of a closed-loop system. This may be extremely useful when considering control design of a continuous plant with a discrete controller.

7. Concluding remarks. In this paper, we have considered the computability of the evolution of a hybrid automaton, in which input and output data are specified by arbitrarily accurate approximations to the exact values.

The main points are summarized below:

1. It is impossible in general to compute the evolution (simulation, reachable sets) of a hybrid automaton to arbitrary accuracy, and this holds even for simple classes of hybrid automaton, such as piecewise-constant derivative systems.
2. The obstruction to computability is due to discontinuities in the temporal evolution and in the spatial dependence on the initial conditions. Away from discontinuity points, the evolution is computable. Essentially, the only hybrid automata for which the evolution can be computed to arbitrary accuracy for any initial condition are those for which every trajectory starting in a given mode undergoes the same sequence of discrete events.
3. It is possible to regularize any hybrid automaton such that it is possible to compute convergent approximations to the evolution from above ("closure semantics") or below ("interior semantics"), but the regularizations admit different solution sets. The regularization of a deterministic system necessarily either is nondeterministic or admits blocking.
4. The regularization using interior semantics cannot handle crossings of guard sets properly. Instead, we need to use a different regularization "crossing semantics." Under a regularity condition on the crossings of the guard sets, it is possible to compute convergent lower-approximations to the evolution; otherwise spurious solutions may be introduced.
5. The semicomputability results are valid for general classes of systems, including nonsmooth, discontinuous, and nondeterministic systems. Restricting our attention to a special subclass of hybrid automata does not change what is possible to compute, but may allow for more efficient algorithms.
6. The framework of "computable analysis" is a powerful machinery for discussing computational aspects of hybrid automata theory. It provides a clear notion of what we should be aiming to compute about a given mathematical object, which can be interpreted in terms of convergent sequences of approximations. It gives natural topological conditions under which the computation can be proved to be impossible. It also provides a methodology of proving computability results using natural mathematical language without having to resort to the details of ϵ-δ style proofs.

There are many interesting areas for further research, especially in the analysis of lower-semicomputability. In particular, it would be useful to have generic verifiable conditions under which all crossings are detectable. For general systems, it would also be interesting to give an exact classification of the computability of the evolution with respect to the arithmetic hierarchy. It would also be interesting to extend this analysis to other problems, such as verification and control synthesis, and to other classes of systems. There is some evidence to suggest that stochastic systems with a diffusion term [17, 8] may have better computability properties than deterministic systems. It would also be interesting to compare decidability of system properties in the framework of computable analysis, in which *only* approximations to the input are considered, with computability in some algebraic framework in which exact computations are possible. In light of previous work on piecewise-constant derivative systems [2] and (non-)o-minimal affine systems [38, 14], it seems likely that while there are specific problem instances which can be decided using algebraic methods, using exact descriptions does not fundamentally change the class of solvable problems. Finally, it is vital to develop more efficient numerical algorithms for the computation of upper- and lower-approximations to the system evolution.

REFERENCES

[1] E. Asarin, T. Dang, and O. Maler, *d/dt: A verification tool for hybrid systems*, in Proceedings of the 40th IEEE Conference on Decision and Control, IEEE Press, New York, 2001, pp. 2893–2898.

[2] E. Asarin, O. Maler, and A. Pnueli, *Reachability analysis of dynamical systems having piecewise-constant derivatives*, Theoret. Comput. Sci., 138 (1995), pp. 35–65.

[3] J.-P. Aubin and A. Cellina, *Differential Inclusions: Set-valued Maps and Viability Theory*, Grundlehren Math. Wiss. [Fundamental Principles of Mathematical Sciences] 264, Springer-Verlag, Berlin, 1984.

[4] J.-P. Aubin, J. Lygeros, M. Quincampoix, and S. Sastry, *Impulse differential inclusions: A viability approach to hybrid systems*, IEEE Trans. Automat. Control, 47 (2002), pp. 2–20.

[5] B. I. Silva, K. Richeson, B. Krogh, and A. Chutinan, *Modeling and verifying hybrid dynamic systems using CheckMate*, in Proceedings of the 4th International Conference on Automation of Mixed Processes, S. Engell, S. Kowalewski, and J. Zaytoon, eds., Shaker Verlag, Aachen, Germany, 2000, pp. 323–328.

[6] A. Balluchi, A. Casagrande, P. Collins, A. Ferrari, T. Villa, and A. L. Sangiovanni-Vincentelli, *Ariadne: A framework for reachability analysis of hybrid automata*, in Proceedings of the 17th International Symposium on the Mathematical Theory of Networks and Systems, Kyoto, Japan, 2006, pp. 1269–1267.

[7] A. Bauer, *The Realizability Approach to Computable Analysis and Topology*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, 2000.

[8] J. Bect, H. Baili, and G. Fleury, *Generalized Fokker-Planck equation for piecewise-diffusion processes with boundary hitting resets*, in Proceedings of the 17th International Symposium on the Mathematical Theory of Networks and Systems, Kyoto, Japan, 2006, pp. 1360–1367.

[9] G. Beer, *Topologies on Closed and Closed Convex Sets*, Math. Appl. 268, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[10] A. Benveniste, B. Caillaud, and M. Pouzet, *The fundamentals of hybrid systems modelers*, in Proceedings of the 49th IEEE Conference on Decision and Control, IEEE Press, New York, 2010, pp. 4180–4185.

[11] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation*, Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.

[12] O. Botchkarev and S. Tripakis, *Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations*, in Hybrid Systems: Computation and Control, N. Lynch and B. Krogh, eds., Lecture Notes in Comput. Sci. 1790, Springer-Verlag, Berlin, Heidelberg, New York, 2000, pp. 73–88.

[13] V. Brattka and G. Presser, *Computability on subsets of metric spaces*, Theoret. Comput. Sci., 305 (2003), pp. 43–76.

[14] T. Brihaye and C. Michaux, *On the expressiveness and decidability of o-minimal hybrid systems*, J. Complex., 21 (2005), pp. 447–478.

[15] M. Broucke and A. Arapostathis, *Continuous interpolation of solutions of Lipschitz inclusions*, J. Math. Anal. Appl., 258 (2001), pp. 565–572.

[16] M. Broucke and A. Arapostathis, *Continuous selections of trajectories of hybrid systems*, Systems Control Lett., 47 (2002), pp. 149–157.

[17] M. L. Bujorianu and J. Lygeros, *Theoretical foundations of stochastic hybrid systems*, in Proceedings of the Sixteenth International Symposium on Mathematical Theory of Networks and Systems, Katholiek Univ. Leuven, Belgium, 2004. CD-Rom.

[18] P. Collins, *A trajectory-space approach to hybrid systems*, in Proceedings of the International Symposium on the Mathematical Theory of Networks and Systems, Katholiek Univ. Leuven, Belgium, 2004. CD-Rom.

[19] P. Collins, *Continuity and computability of reachable sets*, Theoret. Comput. Sci., 341 (2005), pp. 162–195.

[20] P. Collins, *Optimal semicomputable approximations to reachable and invariant sets*, Theory Comput. Syst., 41 (2007), pp. 33–48.

[21] P. Collins, *Controllability and falsification of hybrid systems*, in Proceedings of the 10th European Control Conference, Budapest, 2009.

[22] P. Collins and D. Graça, *Effective computability of solutions of differential inclusions—The ten thousand monkeys approach*, J. UCS, 15 (2009), pp. 1162–1185.

[23] P. Collins and J. Lygeros, *Computability of finite-time reachable sets for hybrid systems*, in Proceedings of the 44th IEEE Conference on Decision and Control, IEEE Press, New York, 2005, pp. 4688–4693.

[24] J. Davoren and I. Epstein, *Topologies, Convergence, and Uniformities in General Hybrid Path Spaces*, preprint, 2008. Available online at http://people.eng.unimelb.edu.au/davoren/00-davoren-epstein-topology-hybrid-paths-feb2008.pdf

[25] M. Dellnitz, G. Froyland, and O. Junge, *The algorithms behind GAIO-set oriented numerical methods for dynamical systems*, in Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems, B. Fiedler, ed., Springer, Berlin, 2001, pp. 145–174, 805–807.

[26] A. Edalat and D. Pattinson, *Denotational semantics of hybrid automata*, J. Logic Algebraic Programming, 73 (2007), pp. 3–21.

[27] G. Frehse, *Phaver: Algorithmic verification of hybrid systems past hytech*, in Hybrid Systems: Computation and Control, M. Morari and L. Thiele, eds., Lecture Notes in Comput. Sci. 3414, Springer, Berlin, New York, 2005, pp. 258–273.

[28] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott, *Continuous Lattices and Domains*, Encyclopedia Math. Appl. 93, Cambridge University Press, Cambridge, UK, 2003.

[29] R. Goebel, J. Hespanha, A. R. Teel, C. Cai, and R. Sanfelice, *Hybrid systems: Generalized solutions and robust stability*, in Proceedings of the Symposium on Nonlinear Control Systems, Elsevier, Amsterdam, 2004, pp. 1–12.

[30] R. Goebel and A. R. Teel, *Solutions to hybrid inclusions via set and graphical convergence with stability theory applications*, Automatica J. IFAC, 42 (2006), pp. 573–587.

[31] L. Grüne and O. Junge, *Optimal stabilization of hybrid systems using a set oriented approach*, in Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, 2006, pp. 2089–2093.

[32] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, *Beyond hytech: Hybrid systems analysis using interval numerical methods*, in Hybrid Systems: Computation and Control, N. Lynch and B. Krogh, eds., Lecture Notes in Comput. Sci. 1790, Springer-Verlag, Berlin, Heidelberg, New York, 2000, pp. 130–144.

[33] L. Jaulin, M. Kieffer, O. Didrit, and É. Walter, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*,With 1 CD-ROM (UNIX, Sun Solaris), Springer-Verlag London Ltd., London, 2001.

[34] R. E. Kalman, P. L. Falb, and M. A. Arbib, *Topics in Mathematical System Theory*, McGraw-Hill, New York, 1969.

[35] E. Klein and A. C. Thompson, *Theory of Correspondences: Including Applications to Mathematical Economics*, Canad. Math. Soc. Ser. Monogr. Adv. Texts, John Wiley & Sons Inc., New York, 1984.

[36] K.-I. Ko, *Complexity Theory of Real Functions*, Progr. Theoret. Comput. Sci., Birkhäuser Boston Inc., Boston, MA, 1991.

[37] W. Kohn, A. Nerode, J. B. Remmel, and A. Yakhnis, *Viability in hybrid systems*, Theoret. Comput. Sci., 138 (1995), pp. 141–168.

[38] G. Lafferiere, G. J. Pappas, and S. Sastry, *O-minimal hybrid systems*, Math. Control Signals Syst., 13 (2000), pp. 1–21.

[39] J. Lygeros, K. H. Johansson, S. Sastry, and M. Egerstedt, *On the existence of executions of hybrid automata*, in Proceedings of the 38th IEEE Conference on Decision and Control, IEEE Press, New York, 1999, pp. 2249–2254.

[40] J. Lygeros, K. H. Johansson, S. N. Simić, J. Zhang, and S. S. Sastry, *Dynamical properties of hybrid automata*, IEEE Trans. Automat. Control, 48 (2003), pp. 2–17.

[41] J. R. Munkres, *Topology*, 2nd ed., Prentice Hall, Upper Saddle River, NJ, 2000.

[42] A. Nerode and W. Kohn, *Models for Hybrid Systems: Automata, Topologies, Controllability, Observability*, in Hybrid Systems, Springer-Verlag, London, 1993, pp. 317–356.

[43] J. W. Polderman and J. C. Willems, *Introduction to Mathematical Systems Theory: A Behavioral Approach*, Texts Appl. Math. 26, Springer-Verlag, New York, 1998.

[44] A. Puri and P. Varaija, *Decidable hybrid systems*, Math. Comput. Model., 23 (1996), pp. 191–202.

[45] A. Puri, P. Varaiya, and V. Borkar, *Epsilon-approximation of differential inclusions*, in Hybrid Systems III, R. Alur, T. A. Henzinger, and E. D. Sontag, eds., Lecture Notes in Comput. Sci. 1066, Springer, Berlin, 1996, pp. 362–376.

[46] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, Grundlehren Math. Wiss. 317 [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1998.

[47] D. van Beek, M. Reniers, R. Schiffelers, and J. Rooda, *Foundations of a compositional interchange format for hybrid systems*, in Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control, Pisa, Italy, 2007, pp. 587–600.

[48] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, Lecture Notes in Control and Inform. Sci. 251, Springer, London, 2000.

[49] K. Weihrauch, *Computable Analysis: An Introduction*, Texts in Theoret. Comput. Sci. EATCS Ser., Springer-Verlag, Berlin, 2000.