Form Methods Syst Des
DOI 10.1007/s10703-013-0196-2

Safety verification of non-linear hybrid systems is quasi-decidable

Stefan Ratschan

Abstract Safety verification of hybrid systems is undecidable, except for very special cases. In this paper, we circumvent undecidability by providing a verification algorithm that provably terminates for all robust problem instances, but need not necessarily terminate for non-robust problem instances. A problem instance x is robust iff the given property holds not only for x itself, but also when x is perturbed a little bit. Since, in practice, well-designed hybrid systems are usually robust, this implies that the algorithm terminates for the cases occurring in practice. In contrast to earlier work, our result holds for a very general class of hybrid systems, and it uses a continuous time model.

Keywords Hybrid systems · Safety verification · Decidability · Robustness

1 Introduction

Terminating algorithms for the verification of hybrid systems are known only for very special cases. In fact, most classes of hybrid systems verification problems are known to be undecidable [15]. Recently, there have been attempts at circumventing this [9–11] by observing that, in practice, hybrid systems can never model a given real system precisely, but only up to perturbations. Hence it suffices to verify robust systems, that is, systems that do not change the desired property under perturbations.[1]

We say that a problem is *quasi-decidable* iff a (possibly non-terminating) algorithm exists whose result is always correct, but which is required to terminate only for robust inputs. We show quasi-decidability of safety verification of a class of hybrid systems that allows arbitrary Boolean combinations of non-linear differential equalities and inequalities for defining

[1] In fact, in the special case of timed automata, there is a whole stream of work on *avoiding* the verification of non-robust properties, see for example [20, 25].

This is an extended version of a conference paper [23]. This work was supported by the Czech Science Foundation (GAČR) grant number P202/12/J060 with institutional support RVO:67985807.

S. Ratschan (✉)
Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic
e-mail: stefan.ratschan@cs.cas.cz

the continuous flow, and arbitrary Boolean combinations of non-linear equalities and inequalities for defining the set of initial and unsafe states, and for defining the set of possible discontinuous jumps of the system.

Theoretical results such as this one (see also [7–11, 21]) have heavy practical consequences: Up to now, hybrid systems verification algorithms have been evaluated purely experimentally, on finitely many benchmark examples. However, one would like a practical verification algorithm to terminate for *all* robust inputs. Hence we now have a formal tool to evaluate the power of practical verification algorithms.

Our result holds for a very general class of hybrid systems that includes non-linear differential equalities and inequalities. In contrast to that, in Fränzle's result [10] all defining constraints have to be polynomial and especially, the continuous flow has to be given not in the form of differential equations but in the form of polynomial flows which, in general, does not even allow the modeling of linear differential equations. In the case where the safety property does not hold (i.e., falsification of the safety property), we in fact also restrict ourselves to polynomials, however, this includes polynomial *differential equations* (in contrast to polynomial flows). In the case where the safety property holds (i.e., verification of the safety property), we even allow the constraints defining the differential equations to contain transcendental function symbols such as sin and exp.

In contrast to our earlier work [9], for the result in this paper we use a continuous time model which results in fundamental additional difficulties which we will discuss in detail in Sect. 5, along with further related work.

The content of the paper is as follows: In Sect. 2 we define our basic notions and the main theorem of the paper. In the following two sections we prove this theorem by providing both an algorithm for verification (Sect. 3) and falsification (Sect. 4). In Sect. 5 we discuss related work, and in Sect. 6 we conclude the paper.

2 Hybrid systems and their quasi-decidability

In this section we describe the solved problem in detail. In the literature, state spaces of dynamical systems are usually defined using tuples (for example, tuples in $\mathbb{R}^n$). Here, we take a little bit more flexible approach, that allows us to directly access the individual tuple elements using names. For these names we use a finite set V whose elements we call *variables*. Moreover, we use the set $\dot{V} \doteq \{\dot{v} \mid v \in V\}$ to access the values of derivatives, and the set $V' \doteq \{v' \mid v \in V\}$ to access the result of a discrete state change (i.e., of jumps).

Moreover, we fix a finite set M whose elements we call *modes*, and use the additional specific variable name $\mathtt{mode}$ to access them, and the variable name $\mathtt{mode}'$ to access them in the case of the result of a discrete jump.

Now we call a function that assigns to some symbols from $\{\mathtt{mode}, \mathtt{mode}'\}$ a value from M and to some elements of $V \cup \dot{V} \cup V'$ a real value a *valuation*. These valuations will take the role of tuples to form the state space of hybrid systems. For a subset X of $\{\mathtt{mode}, \mathtt{mode}'\} \cup V \cup \dot{V} \cup V'$ we denote the set of valuations that assigns values exactly to the elements of X by $\Gamma(X)$.

For every valuation σ in $\Gamma(\{\mathtt{mode}\} \cup V)$, we denote by $\mathtt{Prime}(\sigma)$ the corresponding valuation with primed variables, that is, $\mathtt{Prime}(\sigma)$ is a valuation in $\Gamma(\{\mathtt{mode}'\} \cup V')$, and for all $v \in \{\mathtt{mode}\} \cup V$, $\mathtt{Prime}(\sigma)(v') = \sigma(v)$.

For two valuations σ_1, σ_2 that coincide on joint variables, we define their concatenation $\sigma_1 \bullet \sigma_2$ as the valuation that is defined on the union of the two domains of definition and always assigns the corresponding value. That is, for $\sigma_1 \in \Gamma(X_1)$ and $\sigma_2 \in \Gamma(X_2)$ such that for all $v \in X_1 \cap X_2$, $\sigma_1(v) = \sigma_2(v)$, we have that for all $v \in X_1$, $(\sigma_1 \bullet \sigma_2)(v) = \sigma_1(v)$, and for all $v \in X_2$, $(\sigma_1 \bullet \sigma_2)(v) = \sigma_2(v)$.

Definition 1 A *hybrid system* is a tuple of the form $(S, \mathit{Init}, \mathit{Flow}, \mathit{Jump}, \mathit{Unsafe})$ where S (the *state space* of the hybrid system) is a subset of $\Gamma(\{\mathtt{mode}\} \cup V)$ such that for every $v \in V$ we have a non-empty closed real interval I_v such that $S = \{\sigma \mid \sigma \in \Gamma(\{\mathtt{mode}\} \cup V), \forall v \in V, \sigma(v) \in I_v\}$. In other words, the continuous part of the state space has the form of a hyper-rectangle. In addition,

- $\mathit{Init} \subseteq S$,
- $\mathit{Flow} \subseteq \Gamma(\{\mathtt{mode}\} \cup V \cup \dot{V})$, such that for all $\sigma \in \mathit{Flow}$, for all $v \in V$, $\sigma(v) \in I_v$,
- $\mathit{Jump} \subseteq \Gamma(\{\mathtt{mode}\} \cup V \cup \{\mathtt{mode}'\} \cup V')$, such that for all $\sigma \in \mathit{Jump}$, for all $v \in V$, $\sigma(v) \in I_v$ and $\sigma(v') \in I_v$, and
- $\mathit{Unsafe} \subseteq S$.

That is, a hybrid system has a set of initial and unsafe elements that are sub-sets of the state space. Moreover, it relates derivatives to state space elements, and relates state space elements to primed versions of state space elements. Note that the set *Flow* does not necessarily relate a derivative to all state space elements. The case where no derivative is related to a certain state space element simply expresses the fact that no flow is possible, and hence a jump has to be taken (viz. the notion of a forced or urgent transition).

We will use the following objects to describe continuous evolution of hybrid systems:

Definition 2 A *flow of length t over $S \subseteq \Gamma(\{\mathtt{mode}\} \cup V)$* is a function $\phi : [0, t] \to S$ such that

- $\phi(s)(\mathtt{mode})$ is constant over all $s \in [0, t]$, and
- for every $v \in V$, the function ϕ^v that assigns to every $s \in [0, t]$ the value $\phi(s)(v)$, is differentiable.

Based on this, we define $\dot{\phi} : [0, t] \to \Gamma(\dot{V})$ in such a way that for every $s \in [0, t]$, $v \in V$, $\dot{\phi}(s)(\dot{v}) = \dot{\phi}^v(s)$.

The property we study in this paper is reachability of the set of unsafe states:

Definition 3 For a given hybrid system $(S, \mathit{Init}, \mathit{Flow}, \mathit{Jump}, \mathit{Unsafe})$, an *error trajectory* is a sequence of flows $(\phi_0, \ldots, \phi_n)$ over S of lengths $(t_1, \ldots, t_n)$ such that, for all $i \in \{0, \ldots, n\}$

- if $i < n$, then $\phi_i(t_i) \bullet \mathtt{Prime}(\phi_{i+1}(0)) \in \mathit{Jump}$ or $\phi_i(t_i) = \phi_{i+1}(0)$
- for all $s \in [0, t_i]$, $\phi_i(s) \bullet \dot{\phi}_i(s) \in \mathit{Flow}$,

and $\phi_0(0) \in \mathit{Init}$, $\phi_n(t_n) \in \mathit{Unsafe}$. A hybrid system is *safe* if it does not have an error trajectory.

A notion of solution of a hybrid system immediately follows from this definition after dropping the condition that $\phi_n(t_n) \in \mathit{Unsafe}$. However, we will not need an explicit definition of solution in this paper, since the notion of error trajectory is precisely what is needed for defining safety of a hybrid system.

For describing hybrid systems we use constraints. We define an *arithmetical term* to be an expression that may contain variables in V, rational constants, and function symbols in

$\{+, \times, \sin, \cos, \exp, \dots\}$.[2] Now we define a *constraint* to be a Boolean combination of two types of atomic constraints:

- equalities and inequalities of the form $\mathtt{t}\ r\ c$, where $\mathtt{t}$ is an arithmetical term, $r \in \{=, \leq, \geq\}$, and c is a rational number.
- equalities and inequalities of the form $\mathtt{mode} = m$ or $\mathtt{mode} \neq m$, where $m \in M$ (we call this a *mode constraint*).

A *flow constraint* is a constraint that, in addition to the above, allows atomic constraints of the form $\dot{v}\ r\ \mathtt{t}$, where $r \in \{=, \leq\}$, v is a variable from V, and $\mathtt{t}$ is an arithmetical term over V. A *jump constraint* is a constraint that, in addition to the variables in $\{\mathtt{mode}\} \cup V$, allows their primed versions, that is, variables in $\{\mathtt{mode}'\} \cup V'$.

The definition of the semantics of such constraints is straight-forward. We denote the function from valuations to real numbers described by a term $\mathtt{t}$ by $[\![\mathtt{t}]\!]$. We write $\sigma \models \mathtt{C}$ for the fact that a valuation σ satisfies a constraint $\mathtt{C}$, and we write $[\![\mathtt{C}]\!]$ for the set of valuations satisfying $\mathtt{C}$. We use corresponding definitions for flow and jump constraints in analogy.

Now we have a way of syntactically describing hybrid systems using constraints:

Definition 4 For a given state space S, and constraints $\mathtt{Init}$, $\mathtt{Flow}$, $\mathtt{Jump}$, and $\mathtt{Unsafe}$ we call the tuple $(S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})$ a *hybrid systems description*. Furthermore we denote by

$$[\![(S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})]\!]$$

the hybrid system $(S, [\![\mathtt{Init}]\!], [\![\mathtt{Flow}]\!], [\![\mathtt{Jump}]\!], [\![\mathtt{Unsafe}]\!])$.

In this case we also say that the hybrid system *fulfills* the corresponding hybrid systems description. We straightforwardly lift Definition 3 from hybrid systems to hybrid system descriptions.

Example 1 For illustrating the above definitions, consider the following simple hybrid system. We assume a set of variables $V = \{x_1, x_2\}$, and a set of modes $M = \{m_1, m_2\}$. The hybrid system has a state space $S = \{\sigma \mid \sigma \in \Gamma(\{\mathtt{mode}\} \cup V), \sigma(\mathtt{mode}) \in \{m_1, m_2\}, \sigma(x_1) \in [0, 1], \sigma(x_2) \in [0, 1]\}$. The set of initial states are given by the constraint

$$\mathtt{mode} = m_1 \wedge x_1 = 0 \wedge x_2 = 0.$$

The constraint $x_2 \geq 1$ describes the unsafe states, and hence, safety of a state does not depend on the mode of this state. The hybrid system may switch modes from m_1 to m_2 if $x_1 \geq 0.4$, that is, the constraint *Jump* is of the form

$$\mathtt{mode} = m_1 \wedge x_1 \geq 0.4 \wedge \mathtt{mode}' = m_2 \wedge x_1' = x_1 \wedge x_2' = x_2.$$

The continuous behavior is quite simple: In mode m_1 a flow is only possible as long as $x_1 \leq 0.5$. In both modes, x_1 evolves with a derivative in the interval $[0.9, 1.1]$, while x_2

[2]For a function $f : \mathbb{R}^n \to \mathbb{R}$, compact intervals $I_1, \dots, I_n$, we need to be able to compute an interval $J \supseteq f(I_1, \dots, I_n)$ such that the over-approximation of J over $f(I_1, \dots, I_n)$ can be made arbitrarily small. Note that this requires continuity of f but *not* Lipschitz continuity. For example, we could include $\sqrt{|x|}$, which is not Lipschitz continuous.

evolves deterministically with slope 1 in mode m_1 and -1 in mode m_2. So we have the flow constraint

$$(\texttt{mode} = m_1 \wedge \dot{x}_1 \geq 0.9 \wedge \dot{x}_1 \leq 1.1 \wedge \dot{x}_2 = 1 \wedge x_1 \leq 0.5)$$
$$\vee(\texttt{mode} = m_2 \wedge \dot{x}_1 \geq 0.9 \wedge \dot{x}_1 \leq 1.1 \wedge \dot{x}_2 = -1).$$

Observe that in mode m_2, the value of variable x_2 decreases. Moreover, the system can stay in mode m_1 only as long as $x_1 \leq 0.5$, and so x_2 can increase only for a limited time. So this hybrid system does not have an error trajectory, and hence it is safe.

It is well-known that—except for very special cases—checking whether a hybrid system is safe is an undecidable problem [15]. However, in the real world, we will not be able to implement a hybrid system description exactly and we do not want to prove a hybrid systems description safe, if the system fulfilling this description is safe but there is a system that fulfills the description up to small perturbations and is unsafe. Hence it suffices to have an algorithm that can prove safety of hybrid systems descriptions for which all hybrid systems that fulfill the description up to small perturbations are safe (such as the example above).[3]

This is a general situation for undecidable problems in domains prone to perturbations. Hence it is worthy to discuss the problem description in a general form, independent of a specific definition of the notion of ”to fulfill up to small perturbations”. We will only later (Definition 9) formalize what we exactly mean by that notion in our domain of hybrid systems. Moreover, by delaying that definition (which is quite involved), the reader can grasp the essence of the problem before delving into details.

In general, we have the situation illustrated in Fig. 1. Here the syntactic level is depicted on the top, and the semantic level on the bottom. We have a language $\mathcal{L}$ (in our case, the set of all hybrid systems descriptions) describing corresponding objects in $[\![\mathcal{L}]\!]$ (in our case hybrid

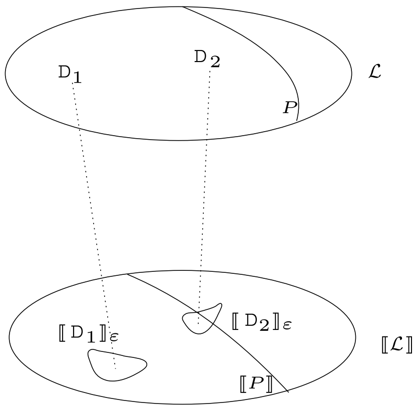
D1
D2
P
$\mathcal{L}$
$[\![\text{D}_1]\!]_\varepsilon$
$[\![\text{D}_2]\!]_\varepsilon$
$[\![P]\!]$
$[\![\mathcal{L}]\!]$

Fig. 1 Robustness

[3]Here it does not suffice to perturb hybrid systems without regard to the constraint language they are described in. The reason for this can be seen in the example of a constraint $0 = 0$. This constraint has the same solution set as the constraint $1 \geq 0$. However, in the case $0 = 0$ small perturbations of the constraint itself change the solution set essentially [22] whereas in the case $1 \geq 0$ they do not. The solution we take here, is to not only consider perturbations on the semantic level, but to also take into account syntactic perturbations. Another solution is, to base the definition of hybrid systems on set-valued functions, and then to perturb those functions. See, for-example, Definition 6.27 in the book by Goebel and others [12].

systems). We have some property P (in our case, safety of hybrid system descriptions) on $\mathcal{L}$ such that for a language element $\mathtt{D} \in \mathcal{L}$, $P(\mathtt{D})$ holds iff $[\![P]\!]([\![\mathtt{D}]\!])$ holds for a certain property $[\![P]\!]$ (in our case, safety of a hybrid system). However, when we try to implement $\mathtt{D}$ in the real world, the result will not precisely fulfill the hybrid systems description $\mathtt{D}$, but will fulfill $\mathtt{D}$ only up to some perturbation of size ε. Hence, we will have a set $[\![\mathtt{D}]\!]_\varepsilon$ of objects fulfilling $\mathtt{D}$ up to perturbations of size ε. It can be the case that $[\![P]\!]$ holds on all elements of $[\![\mathtt{D}]\!]_\varepsilon$ (this is the case for $\mathtt{D}_1$ in the figure), or only on some elements ($\mathtt{D}_2$ in the figure). This difference is described by the following definition.

Definition 5 Let $\mathcal{L}$ be a language describing elements of a set $[\![\mathcal{L}]\!]$, and let P be a property on L, and $[\![P]\!]$ a property on $[\![L]\!]$, such that for all $\mathtt{D} \in L$, $P(\mathtt{D})$ iff $[\![P]\!]([\![\mathtt{D}]\!])$. Let $\mathtt{D} \in [\![L]\!]$, and let $[\![\mathtt{D}]\!]_\varepsilon$ be the set of all elements of $[\![\mathcal{L}]\!]$ fulfilling $\mathtt{D}$ up to ε. Then P *holds robustly* on $\mathtt{D}$ iff there is a real number $\varepsilon > 0$ (the *robustness margin*) such that $[\![P]\!]$ holds on all elements of $[\![\mathtt{D}]\!]_\varepsilon$.

In the figure, P holds robustly on $\mathtt{D}_1$. For $\mathtt{D}_2$, the value ε is not a robustness margin. There might be a smaller value ε', and a smaller corresponding set $[\![\mathtt{D}_2]\!]_{\varepsilon'}$ such that $[\![P]\!]$ holds on all its elements. If, however small we choose ε', some elements of $[\![\mathtt{D}_2]\!]_{\varepsilon'}$ still do not fulfill $[\![P]\!]$, P does not hold robustly on $\mathtt{D}_2$. Only in that case we do not require our algorithms to terminate, that is, in that case an algorithm trying to verify safety of a given hybrid system is allowed to run forever. This is the essential point, why the following notion of quasi-decidability is weaker than decidability.

Definition 6 We call a given property P on a language $\mathcal{L}$ *quasi-semidecidable* iff there is an algorithm A such that for a given $\mathtt{D} \in \mathcal{L}$,

- if $A(\mathtt{D})$ terminates then $P(\mathtt{D})$ (i.e., A is correct),
- $A(\mathtt{D})$ terminates if P holds robustly on $\mathtt{D}$.

If both P and $\neg P$ are quasi-semidecidable then P is *quasi-decidable*.

The definitions above depend on the notion of ”fulfilling a language element up to ε”. We will spend the rest of this section on defining this in our case, that is, defining the notion of a hybrid system fulfilling a hybrid systems description up to ε.

Due to reasons discussed in footnote 3, we have to take into account *syntactic* perturbations here. We define this using a distance measure on constraints. Note however, that in the following only the limit case of this definition is relevant, since Definition 5 does not consider a fixed robustness margin ε, but only requires existence of an $\varepsilon > 0$.

The basic idea for defining this distance measure is, that two constraints are the same up to ”addition of constants up to a certain size”:

Definition 7

- We call a term *basic*, if it is either a variable, or a constant, or a term of the form $x + c$, where x is a variable, and c a constant. If the set of variables contained in a basic term (this is either a singleton set or the empty set) is the same in two basic terms, we define the distance between these terms as the distance between the corresponding constants, using the constant 0 if one of the terms does not contain a constant. If the set of contained variables is not the same in both basic terms, their distance is ∞.

– The distance $d(\mathtt{C}, \mathtt{C}')$ between two constraints $\mathtt{C}$ and $\mathtt{C}'$ is ε iff $\mathtt{C}'$ can be obtained from $\mathtt{C}$ by replacing some basic terms by basic terms of finite distance and ε is the maximum of these distances. Otherwise, the distance is ∞.

Example 2 For measuring the distance between the constraint $(x+2)^2 + 1x \leq 0$ and $x^2 + 2x \leq 0$ we observe that for getting from the first to the second constraint we have to replace the basic term $x+2$ by x, and the basic term 1 by the basic term 2. The distance is the maximum of the distances of corresponding basic terms, that is, the distance is 2.

Example 3 The constraints $(x-2)^2 - 1 \leq 0$ and $x^2 - 4x + 4 - 1 \leq 0$, although semantically equivalent, have infinite distance. This does not pose any problem here. On the contrary, this makes our result stronger, since it leads to many hybrid systems descriptions being robust, and hence to a strong termination condition for our algorithms (in Fig. 1 the blobs in the lower part become smaller, resulting in more blobs to completely lie in P).

We continue with defining an analogon of the notion of ”fulfilling a hybrid systems description up to ε” for our constraint language.

Definition 8 A set P of valuations is an *ε-perturbed solution set* of a constraint $\mathtt{C}$ iff

– for every valuation $\sigma \in P$, there is a constraint $\mathtt{C}^*$ with $d(\mathtt{C}, \mathtt{C}^*) \leq \varepsilon$ such that $\sigma \models \mathtt{C}^*$, and
– for every valuation $\sigma \notin P$, there is a constraint $\mathtt{C}^*$ with $d(\mathtt{C}, \mathtt{C}^*) \leq \varepsilon$ such that $\sigma \not\models \mathtt{C}^*$.

In other words, the set P may contain valuations that do *not* satisfy the constraint, and may not contain valuations that *do* satisfy constraint, but we have to make sure that in both cases the error that we make is not too large. Note that this does *not* necessarily mean that P is the solution set of a perturbed constraint $\mathtt{C}^*$:

Example 4 The interval $[-1, 1]$ is a 1-perturbed solution set of the constraint $x = 0$. However, there is no ε such that $x = \varepsilon$ has the solution set $[-1, 1]$.

Lifting this definition to hybrid systems is straightforward:

Definition 9 Given a hybrid system description $(\mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})$, a hybrid system $(\mathit{Init}, \mathit{Flow}, \mathit{Jump}, \mathit{Unsafe})$ fulfills $(\mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})$ up to ε iff

– *Init* is an ε-perturbed solution set of $\mathtt{Init}$,
– *Flow* is an ε-perturbed solution set of $\mathtt{Flow}$,
– *Jump* is an ε-perturbed solution set of $\mathtt{Jump}$, and
– *Unsafe* is an ε-perturbed solution set of $\mathtt{Unsafe}$.

Note that—in analogy to individual constraints—a hybrid system H fulfilling a hybrid system description $\mathtt{H}$ up to ε does *not* necessarily mean that H fulfills a hybrid system description that is a perturbation of $\mathtt{H}$ (see Example 4 above).

Since Definition 9 was the last missing element of the definition of quasi-decidability, after setting $\mathcal{L}$ to the set of hybrid system descriptions, and P to their safety in Definition 6, we now have a complete formalization of the notion of quasi-decidability of hybrid systems. So we are ready to formulate the main theorem of this paper:

Theorem 1 *Safety of hybrid system descriptions is quasi-semidecidable. Moreover, it is quasi-decidable in the case where we allow only addition and multiplication as function symbols in hybrid system descriptions.*

A proof of this theorem consists of two quasi-semidecidability proofs, one for the positive case of verification of the safety property, and one for the negative case of falsification of the safety property. We will use the following two sections for the two corresponding parts of the proof. Within these sections we will provide respective algorithms for verifying and falsifying hybrid systems.

3 Quasi-semidecidability of verification

For proving quasi-semidecidability of verification we use the fact that for every hybrid system there is a rectangular ε-approximation [14]. Here we have to overcome two major obstacles:

- The original proof of this existence property was not constructive.
- Although rectangular automata have a much simpler structure than general hybrid systems, their safety is still undecidable.

Before solving these problems, we introduce a representation of rectangular sets: A *box* is a function that assigns to some variables in $V \cup \dot{V} \cup V'$ a non-empty closed real interval, and to some variables in $\{\mathtt{mode}, \mathtt{mode}'\}$ a subset of modes from M. Throughout the paper we use the situation that a box B does not assign a value to a given variable as a shortcut for the value $B(v)$ being M, if $v \in \{\mathtt{mode}, \mathtt{mode}'\}$, and being $[-\infty, \infty]$, otherwise. We will say that a box has *dimension d* iff it assigns d real intervals (i.e., d intervals not equal to $[-\infty, \infty]$). We lift set membership to boxes by defining a valuation σ to be element of a box B iff for every variable v on which σ is defined, $\sigma(v) \in B(v)$. Analogously we lift other set operations such as $\subseteq$ and $\cap$ using the corresponding variable-wise operations on intervals and sets of modes, respectively. Box union $\uplus$ is defined by lifting union for variables in $\{\mathtt{mode}, \mathtt{mode}'\}$, and interval union (the smallest interval containing both arguments) for the other variables. For boxes we define concatenation analogously as for valuations. We call a box *proper*, if it only assigns intervals (and no modes).

A *sat-box* (for satisfiability box) is either a box, or the value $\bot$ which we call the *empty box*. Such sat-boxes will be used for flow constraints where we either deduce unsatisfiability or a box bounding the set of possible derivatives. A sat-box has *dimension d* iff it is equal to $\bot$ or if it is a box of dimension d (hence $\bot$ can have any dimension). Sometimes we will write $\mathbf{F}$ for $\bot$ and $\mathbf{T}$ for the unique zero-dimensional box, and use them in the role of the corresponding Boolean constants. The box operations $\cap$ and $\uplus$ can be easily lifted from boxes to sat-boxes by considering $\bot$ to be the smallest element in the $\subseteq$ order. Also, the element relation $\in$ can be naturally lifted by defining $\bot$ to have no element (which, of course, corresponds to its name "empty box").

Now we start with removing the first obstacle mentioned at the beginning of this section: computing a rectangular over-approximation of a hybrid system such that the over-approximation error is smaller than a given bound.

The algorithm uses interval arithmetic as its basis. For a term $\mathtt{t}$, and proper box B, let $I(\mathtt{t})(B)$ denote the evaluation of $\mathtt{t}$ on B using interval arithmetic [19]. For polynomials, computation with interval endpoints can be implemented exactly, in rational number arithmetic. For terms containing transcendental function symbols such as sin, however, one has to use (conservative) rounding [24]. Here we assume the usage of fixed-precision floating-point arithmetic. Moreover, to ensure convergencence (see Lemma 1 below for details), we assume that the used precision goes to infinity as the size of the box B goes to zero.

The result of interval arithmetic over-approximates the set of all values the term $\mathtt{t}$ takes in the box B, due to the so-called Fundamental Theorem of Interval Arithmetic [18].

Property 1

$$I(\mathtt{t})(B) \supseteq \left\{ [\![\mathtt{t}]\!](\sigma) \mid \sigma \in B \right\}$$

Now we can over-approximate the satisfiability information of constraints by defining the symbol $\models_I$ (*interval satisfiability check*) for a box B as follows:

- $B \models_I \mathtt{mode} = m$ is $\mathbf{T}$ if $m \in B(\mathtt{mode})$, and $\mathbf{F}$ otherwise,
- $B \models_I \mathtt{t}\ r\ 0$, where $\mathtt{t}$ does not contain dotted variables, is $\mathbf{T}$ iff there exists a real value $x \in I(\mathtt{t})(B)$ such that $x\ r\ 0$, and $\mathbf{F}$, otherwise,
- $B \models_I \mathtt{C}_1 \wedge \mathtt{C}_2$ is $B \models_I \mathtt{C}_1 \cap B \models_I \mathtt{C}_2$, and
- $B \models_I \mathtt{C}_1 \vee \mathtt{C}_2$ is $B \models_I \mathtt{C}_1 \uplus B \models_I \mathtt{C}_2$.

Example 5 Let $\mathtt{C}$ be the constraint $x^2 - 1 = 0 \wedge x - 2 \geq 0$, and let B be the box $x \mapsto [-10, 0]$. Interval arithmetic evaluates the terms in $\mathtt{C}$ recursively. So $I(x^2)(B) = [0, 100]$, and $I(x^2 - 1)(B) = [-1, 99]$. Since this interval contains zero, $(B \models_I x^2 - 1 = 0) = \mathbf{T}$. Moreover, $I(x - 2)(B) = [-12, -2]$, and $(B \models_I x - 2 \geq 0) = \mathbf{F}$. In the zero-dimensional case, intersection and union of boxes implements conjunction and disjunction of the corresponding Boolean values. So $(B \models_I \mathtt{C}) = \mathbf{T} \cap \mathbf{F} = \mathbf{F}$.

Remember that, by default, variables are assigned the interval $[-\infty, \infty]$. Hence the semantics is also well-defined in cases where the branches of a conjunction (or disjunction) contain different variables.

We generalize the interval satisfiability check to constraints containing dotted variables (denoting derivatives). In this case, the result is a sat-box, whose dimension (if containing a box) is equal to the number of dotted variables. The purpose of this definition is to over-approximate the projection of the solution set of the constraint to these variables:

- $B \models_I \dot{a} = \mathtt{t}$ is defined as $\{\dot{a} \mapsto I(\mathtt{t})(B)\}$
- $B \models_I \dot{a} \leq \mathtt{t}$ is defined as $\{\dot{a} \mapsto [-\infty, \overline{I(\mathtt{t})(B)}]\}$
- $B \models_I \dot{a} \geq \mathtt{t}$ is defined as $\{\dot{a} \mapsto [\underline{I(\mathtt{t})(B)}, \infty]\}$

The rest of the definition is kept unchanged.

Example 6 Let $\mathtt{C}$ be the flow constraint $\dot{x} = x^2 \wedge x - 2 \geq 0$, and let B be the box $x \mapsto [1, 3]$. Then $B \models_I \dot{x} = x^2$ is the box $\{\dot{x} \mapsto [1, 9]\}$ and $(B \models_I x - 2 \geq 0) = \mathbf{T}$. Hence $\{\dot{x} \mapsto [1, 9]\} \cap \mathbf{T} = \{\dot{x} \mapsto [1, 9]\}$ (remember that $\mathbf{T}$ is the unique zero dimensional box that assigns to every variable the default interval $[-\infty, \infty]$).

For the slightly modified constraint $x^2 - 1 = 0 \wedge x - 10 \geq 0$, however, $B \models_I x - 10 \geq 0$ evaluates to $\mathbf{F}$, and hence also the whole constraint.

This definition fulfills its purpose due to the following generalization of the fundamental theorem of interval arithmetic to our constraints:

Theorem 2 *For every constraint* $\mathtt{C}$, *box* B *on the undotted variables of* $\mathtt{C}$, *valuation* $\sigma \in B$, *and valuation* $\dot{\sigma}$ *on the dotted variables of* $\mathtt{C}$ *such that* $\sigma \bullet \dot{\sigma} \models \mathtt{C}$, *we have* $\dot{\sigma} \in B \models_I \mathtt{C}$.

Proof Let $B, \sigma, \dot{\sigma}$ arbitrary, but fixed, fulfilling the assumptions above. We prove that $\dot{\sigma} \in B \models_I \mathtt{C}$. We proceed by induction over the structure of $\mathtt{C}$. We have the following base cases:

- $\mathtt{C}$ is of the form $\mathtt{t} = 0$, where $\mathtt{t}$ is a term. We have $\sigma \bullet \dot{\sigma} \models \mathtt{t} = 0$, and hence $[\![\mathtt{t}]\!](\sigma \bullet \dot{\sigma}) = 0$ and since $\mathtt{t}$ does not contain dotted variables, also $[\![\mathtt{t}]\!](\sigma) = 0$. To prove that $\dot{\sigma} \in B \models_I \mathtt{C}$, we have to prove that $0 \in I(\mathtt{t})(B)$. This holds, since due to the fundamental theorem of interval arithmetic, $[\![\mathtt{t}]\!](\sigma) \in I(\mathtt{t})(B)$ for $\sigma \in B$.
- $\mathtt{C}$ is of the form $\dot{x} = \mathtt{t}$. In this case, since $\sigma \bullet \dot{\sigma} \models \dot{x} = \mathtt{t}$, it holds that $\dot{\sigma} = [\![\mathtt{t}]\!](\sigma)$. To prove that $\dot{\sigma} \in B \models_I \mathtt{C}$, we have to prove that $\dot{\sigma} \in I(\mathtt{t})(B)$. This holds since due to the fundamental theorem of interval arithmetic, $[\![\mathtt{t}]\!](\sigma) \in I(\mathtt{t})(B)$ for $\sigma \in B$.
- $\mathtt{C}$ is of the form $\dot{x} \leq \mathtt{t}$. In this case, since $\sigma \bullet \dot{\sigma} \models \dot{x} \leq \mathtt{t}$, it holds that $\dot{\sigma} \leq [\![\mathtt{t}]\!](\sigma)$. To prove that $\dot{\sigma} \in B \models_I \mathtt{C}$, we have to prove that $\dot{\sigma} \in [-\infty, \overline{I(\mathtt{t})(B)}]$, that is, $\dot{\sigma} \leq \overline{I(\mathtt{t})(B)}$. This holds since due to the fundamental theorem of interval arithmetic, for all x, $[\![\mathtt{t}]\!](x) \in I(\mathtt{t})(B)$, and hence $[\![\mathtt{t}]\!](x) \leq \overline{I([\![\mathtt{t}]\!])(B)}$.
- the other cases of atomic constraint are analogous to the previous cases.

The induction step is easy. □

Example 7 Continuing Example 6, let in addition σ be the valuation $\{x \mapsto 2\}$ (which is an element of B), and let $\dot{\sigma}$ be the valuation $\{\dot{x} \mapsto 4\}$ (for which $\sigma \bullet \dot{\sigma} \models \mathtt{C}$). Then the box $B \models_I \mathtt{C}$ which is $\{\dot{x} \mapsto [1, 9]\}$ contains the valuation $\{\dot{x} \mapsto 4\}$.

In the special case of constraints without dotted variables, interval satisfiability just over-approximates satisfiability:

Corollary 1 *For every constraint* $\mathtt{C}$ *without dotted variables, box* B *on the variables of* $\mathtt{C}$, *if* B *contains a valuation* σ *such that* $\sigma \models \mathtt{C}$, *then* $(B \models_I \mathtt{C}) = \mathbf{T}$.

Equivalently, $(B \models_I \mathtt{C}) = \mathbf{F}$ implies that there is no valuation $\sigma \in B$ such that $\sigma \models \mathtt{C}$. Since the implication only points in one direction, in the case of $(B \models_I \mathtt{C}) = \mathbf{T}$ one cannot conclude anything about the satisfiability of $\mathtt{C}$, and in the case of an unsatisfiable constraint one cannot conclude anything about $(B \models_I \mathtt{C})$. In particular, it can, but need not necessarily happen that for a non-robust constraint unsatisfiable $\mathtt{C}$, $(B \models_I \mathtt{C})$ gives the precise result:

Example 8 For the term x^2 and the box $[-10, 10]$ interval arithmetic $I(x^2)([-10, 10])$ may compute the precise result $[0, 100]$. Then, for the unsatisfiable but not robust constraint $x^2 < 0$, we get the precise result $(B \models_I x^2 < 0) = \mathbf{F}$. If however, $I(x^2)([-10, 10])$ is computed as $[-0.00001, 100]$ then the result is $(B \models_I x^2 < 0) = \mathbf{T}$.

Now we present an algorithm for which we will prove that it over-approximates a given hybrid system arbitrarily closely. For bounding the over-approximation error we use a bound on the size of the boxes. For a non-empty interval $[a, b]$, its width is defined to be $b - a$, and for a non-empty set of modes $M^* \subseteq M$, we define its width to be zero if M^* is a singleton set, and ∞, otherwise. We define the diameter $\mathrm{diam}(B)$ of a box B to be the maximum width of $B(v)$ over all variables v on which B is defined (i.e., not equal $[-\infty, \infty]$).

The algorithm in Fig. 2 approximates a given hybrid systems description using a hybrid systems description completely defined by boxes. Here we use the notation $x \in [\underline{a}, \overline{a}]$ as a short-cut for the constraint $\underline{a} \leq x \wedge x \leq \overline{a}$. The idea is to put a grid of boxes onto the state space, and then

```
Input:
– a hybrid systems description
  $(S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})$,
– a strictly positive real value $\delta$
$G \leftarrow$ set of boxes of diameter $\delta$ covering the state space $S$
$\mathtt{Init}_R \leftarrow \bigvee_{B \in G, B \models_I \mathtt{Init}} [\mathtt{mode} = B(\mathtt{mode}) \wedge \bigwedge_{v \in V} v \in B(v)]$
$\mathtt{Flow}_R \leftarrow \bigvee_{B \in G} [\mathtt{mode} = B(\mathtt{mode}) \wedge$
        $\bigwedge_{v \in V} v \in B(v) \wedge \bigwedge_{v \in V} \dot{v} \in (B \models_I \mathtt{Flow})]$
$\mathtt{Jump}_R \leftarrow \bigvee_{B, B' \in G, \langle B, B' \rangle \models_I \mathtt{Jump}} [$
        $\mathtt{mode} = B(\mathtt{mode}) \wedge \bigwedge_{v \in V} v \in B(v) \wedge$
        $\mathtt{mode}' = B'(\mathtt{mode}) \wedge \bigwedge_{v \in V} v' \in B'(v)]$
$\mathtt{Unsafe}_R \leftarrow \bigvee_{B \in G, B \models_I \mathtt{Unsafe}} [$
        $\mathtt{mode} = B(\mathtt{mode}) \wedge \bigwedge_{v \in V} v \in B(v)]$
$(S, \mathtt{Init}_R, \mathtt{Flow}_R, \mathtt{Jump}_R, \mathtt{Unsafe}_R)$
```

Fig. 2 Over-approximating abstraction

- to test on each box using the interval satisfiability check, whether it might contain an initial or unsafe state,
- to test for every pair of boxes whether it might contain a jump between them, and
- to compute an interval containing the possible derivatives for each box.

In contrast to the discrete time case [9], here it does not suffice to abstract to a purely discrete system. The reason is that in discrete time, if the hyper-rectangles are sufficiently small, they can separate two subsequent steps of the system. However, for continuous evolution, this is not possible.

We denote the result computed by the algorithm in Fig. 2 by $A(\mathtt{H}, \delta)$. This is again a hybrid system description, and from Theorem 2 it easily follows that $A(\mathtt{H}, \delta)$ over-approximates $\mathtt{H}$:

Theorem 3 *For the result* $(S, \mathtt{Init}_R, \mathtt{Flow}_R, \mathtt{Jump}_R, \mathtt{Unsafe}_R)$ *of the algorithm application* $A((S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe}), \delta)$, $\mathtt{Init}$ *implies* $\mathtt{Init}_R$, $\mathtt{Flow}$ *implies* $\mathtt{Flow}_R$, $\mathtt{Jump}$ *implies* $\mathtt{Jump}_R$, *and* $\mathtt{Unsafe}$ *implies* $\mathtt{Unsafe}_R$.

And hence the result of the algorithm in Fig. 2 can be used to prove safety of the original system.

Corollary 2 *If* $[\![A((S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe}), \delta)]\!]$ *is safe, then* $[\![(S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})]\!]$ *is also safe.*

However, this does not guarantee anything about the amount of over-approximation of the algorithm. In order to arrive at bounds for this over-approximation, we first study such bounds for constraints. In earlier work [9] we proved results bounding the over-approximation of $\models_I$ for constraints without dotted variables. We generalize those results here to the case with dotted variables:

Lemma 1 *For every constraint* $\mathtt{C}$, *box* B *defined on all undotted variables of* $\mathtt{C}$, *for all* $\varepsilon > 0$ *there is a* $\delta > 0$ *such that for every box* B' *with* $B' \subseteq B$, $\mathrm{diam}(B') < \delta$, *for every* $\sigma \in B'$, *and* *for every* $\dot{\sigma} \in (B' \models_I \mathtt{C})$, *there is a* $\mathtt{C}^*$ *with* $d(\mathtt{C}, \mathtt{C}^*) \leq \varepsilon$, *such that*

$$\sigma \bullet \dot{\sigma} \models \mathtt{C}^*.$$

Proof For proving this lemma we use the fact (which we will call *convergence of interval arithmetic* in the rest of the proof) that for every arithmetical term $\mathtt{e}$ with function symbols in the set $\{+, *, \hat{}, \exp, \sin, \cos\}$, denoting a function $[\![\mathtt{e}]\!]$ and box S, for every $\varepsilon > 0$ there is a $\delta > 0$ such that for every box B with $B \subseteq S$, $\mathrm{diam}(B) < \delta$, for all $y \in I(\mathtt{e})(B)$, there is an $x \in B$ such that $d([\![\mathtt{e}]\!](x), y) \leq \varepsilon$. This fact follows from Lipschitz continuity of interval arithmetic (e.g., Theorem 2.1.1 in Neumaier's book [19]). Moreover, due to Theorem 2.1.5 in the same book, this holds even in rounded interval arithmetic, as long as we let the used precision go to infinity as the size of the box B goes to zero which is precisely how we defined evaluation of terms in interval arithmetic.

Now let $\mathtt{C}$, B, ε be as required by the assumptions of the lemma. We start with proving the special case that $\mathtt{C}$ is of the form: $\mathtt{t} = 0$:

Let $\delta_{\mathtt{t}}$ be the value ensured for $\mathtt{t}$, B, and ε by convergence of interval arithmetic. We choose δ as $\min\{\delta_{\mathtt{t}}, \varepsilon\}$, and assume an arbitrary, but fixed box B', σ, and $\dot{\sigma}$ with $B' \subseteq B$, $\mathrm{diam}(B') < \delta$, $\sigma \in B'$, and $\dot{\sigma} \in (B' \models_I \mathtt{t} = 0)$.

From $\dot{\sigma} \in (B' \models_I \mathtt{t} = 0)$ we know that $0 \in I(\mathtt{t})(B')$. We construct a $\mathtt{C}^*$ with $d(\mathtt{C}, \mathtt{C}^*) \leq \varepsilon$, $\sigma \bullet \dot{\sigma} \models \mathtt{C}^*$ by providing the necessary perturbations of $\mathtt{C}$.

Let x be an element of B' such that $d([\![\mathtt{t}]\!](x), 0) \leq \varepsilon$, as ensured by the convergence of interval arithmetic. We perturb (by adding corresponding constants)

- every undotted variable v in $\mathtt{C}$ by $x(v) - \sigma(v)$ (this perturbation is smaller than ε since $d(\sigma(v), x(v)) \leq \mathrm{diam}(B') \leq \delta = \min\{\delta_{\mathtt{t}}, \varepsilon\} \leq \varepsilon$),
- and perturb the right-hand side of the constraint by $[\![\mathtt{t}]\!](x)$, which is smaller than ε by choice of x.

Then $\sigma \bullet \dot{\sigma} \models \mathtt{C}^*$, which is equivalent to $\sigma \models \mathtt{C}^*$, is equivalent to $x \models \mathtt{t} = c$, with $c = [\![\mathtt{t}]\!](x)$. This holds according to the definition of $\models$.

Now we look at the case where $\mathtt{C}$ is of the form $\dot{a} = \mathtt{t}$.

Let $\delta_{\mathtt{t}}$ be the value ensured for $\mathtt{t}$, B, and ε by convergence of interval arithmetic. We choose δ as $\min\{\delta_{\mathtt{t}}, \varepsilon\}$, assume an arbitrary but fixed box B', σ, and $\dot{\sigma}$ with $B' \subseteq B$, $\mathrm{diam}(B') < \delta$, $\sigma \in B'$, and $\dot{\sigma} \in (B' \models_I \dot{a} = t)$.

From $\dot{\sigma} \in B' \models_I \dot{a} = \mathtt{t}$ we know that $\dot{\sigma} \in I(\mathtt{t})(B')$. We construct a $\mathtt{C}^*$ with $\sigma \bullet \dot{\sigma} \models \mathtt{C}^*$ by providing the necessary perturbations. Let x be such that $d([\![\mathtt{t}]\!](x), \dot{\sigma}) \leq \varepsilon$, as ensured by the convergence of interval arithmetic.

We perturb

- every undotted variable v of $\mathtt{C}$ by $x(v) - \sigma(v)$ (this perturbation is smaller than ε since $d(\sigma(v), x(v)) \leq \mathrm{diam}(B') \leq \delta = \min\{\delta_{\mathtt{t}}, \varepsilon\} \leq \varepsilon$),
- the dotted variables by $[\![\mathtt{t}]\!](x) - \dot{\sigma}$ (this perturbation is smaller than ε by choice of x),
- and do not perturb the right-hand side of the constraint.

Then $\sigma \bullet \dot{\sigma} \models \mathtt{C}^*$ is equivalent to $x \bullet \{\dot{a} \mapsto [\![\mathtt{t}]\!](x)\} \models \mathtt{C}$, that is, $x \bullet \{\dot{a} \mapsto [\![\mathtt{t}]\!](x)\} \models \dot{a} = \mathtt{t}$ which holds according to the definition of $\models$.

In the case where $\mathtt{C}$ is an inequality, for example, of the form $\dot{a} \leq \mathtt{t}$, we have to consider two sub-cases:

- $\dot{\sigma} \in I(\mathtt{t})(B')$: in this case, the proof for the equality case above works.
- $\dot{\sigma} \notin I(\mathtt{t})(B')$: in this case, we choose $\mathtt{C}^*$ as $\mathtt{C}$, and we have: $\sigma \bullet \dot{\sigma} \models \mathtt{C}^*$ is $\sigma \bullet \dot{\sigma} \models \dot{a} \leq t$, which according to the definition of $\models$ is equivalent to $\dot{\sigma} \leq [\![\mathtt{t}]\!](\sigma)$. This holds since $[\![\mathtt{t}]\!](\sigma) \in I(\mathtt{t})(B')$, $\dot{\sigma} \notin I(\mathtt{t})(B')$, and $\dot{\sigma} < \overline{I(\mathtt{t})(B')}$.

In the case where $\mathtt{C}$ is of the form $\mathtt{mode} = m$, the lemma easily holds by choosing $\mathtt{C}^*$ to be equal to $\mathtt{C}$, in which case $d(\mathtt{C}, \mathtt{C}^*) = 0$.

For considering general constraints with conjunction and disjunction, we proceed by induction. This easily goes through by choosing the minimum of the δ for the different atomic constraints and combining the $\mathtt{C}^*$ for the different branches. □

Using Lemma 1 we can bound the over-approximation of the algorithm in Fig. 2 up to arbitrary precision.

Theorem 4 *For every hybrid system description* $\mathtt{H}$, *for all* $\varepsilon > 0$ *there is a* $\delta > 0$ *such that* $[\![A(\mathtt{H}, \delta)]\!]$ *is an* ε*-perturbed instance of* $\mathtt{H}$.

Proof Let $\delta_{\mathtt{C},B,\varepsilon}$ be the value of δ, as ensured by Lemma 1 for the constraint $\mathtt{C}$, the box B, and ε. Let $\varepsilon > 0$ be arbitrary, but fixed. Choose δ as the minimum of $\delta_{\mathtt{C},B,\varepsilon}$ over all constraints $\mathtt{C}$ defining $\mathtt{H}$, and boxes B forming the state space (one box for each mode).

We assume that $\mathtt{H}$ is of the form $(S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})$, and $[\![A(\mathtt{H}, \delta)]\!]$ is of the form $(S, [\![\mathtt{Init}_R]\!], [\![\mathtt{Flow}_R]\!], [\![\mathtt{Jump}_R]\!], [\![\mathtt{Unsafe}_R]\!])$. To prove that $[\![A(\mathtt{H}, \delta)]\!]$ is an ε-perturbed instance of $\mathtt{H}$ we have to prove the corresponding result for each pair of corresponding constraints of $\mathtt{H}$ and $A(\mathtt{H}, \delta)$. Here, in each case, Theorem 3 implies the second item of Definition 8. Hence it suffice to prove the first item for each pair of corresponding constraints:

- To prove that $[\![\mathtt{Init}_R]\!]$ is a ε-perturbed instance of $\mathtt{Init}$, we have to prove that for every $\sigma \in [\![\mathtt{Init}_R]\!]$, there is a constraint $\mathtt{Init}^*$ with $d(\mathtt{Init}, \mathtt{Init}^*) \leq \varepsilon$ such that $\sigma \models \mathtt{Init}^*$. Let σ be an arbitrary, but fixed element of $[\![\mathtt{Init}_R]\!]$. Then σ satisfies at least one disjunct of $\mathtt{Init}_R$. Let B be the mode/box pair generating this disjunct. Then $\sigma \in B$, $B \models_I \mathtt{Init}$ and $\mathrm{diam}(B) \leq \delta_{\mathtt{Init},S,\varepsilon} \leq \delta$, where S is the box forming the state space of the mode of σ. Then, by Lemma 1, there is a constraint $\mathtt{Init}^*$ with $d(\mathtt{Init}, \mathtt{Init}^*) \leq \varepsilon$, $\sigma \models \mathtt{Init}^*$.
- Flow: To prove that $[\![\mathtt{Flow}_R]\!]$ is a ε-perturbed instance of $\mathtt{Flow}$, we have to prove that for every $\sigma \bullet \dot{\sigma} \in [\![\mathtt{Flow}_R]\!]$, there is a constraint $\mathtt{Flow}^*$ with $d(\mathtt{Flow}, \mathtt{Flow}^*) \leq \varepsilon$ such that $\sigma \bullet \dot{\sigma} \models \mathtt{Flow}^*$. Let $\sigma \bullet \dot{\sigma}$ be an arbitrary, but fixed element of $[\![\mathtt{Flow}_R]\!]$. Then $\sigma \bullet \dot{\sigma}$ satisfies at least one disjunct of $\mathtt{Flow}_R$. Let B the mode/box pair generating this disjunct. Hence $\sigma \in B$, $\dot{\sigma} \in (B \models_I \mathtt{Flow})$ and $\mathrm{diam}(B) \leq \delta_{\mathtt{Flow},S,\varepsilon} \leq \delta$, where S is the box forming the state space of the mode of σ. Then, by Lemma 1, there is a constraint $\mathtt{Flow}^*$ such that $d(\mathtt{Flow}, \mathtt{Flow}^*) \leq \varepsilon$, and $\sigma \bullet \dot{\sigma} \models \mathtt{Flow}^*$.
- $\mathtt{Jump}$ and $\mathtt{Unsafe}$: analogous to $\mathtt{Init}$ □

The hybrid system $A(\mathtt{H}, \delta)$ has a very simple form that is equivalent to a rectangular automaton. Still, this rectangular automaton is not necessarily initialized and hence it belongs to an undecidable class [15]. However, after explicitly solving the flow constraints, it can be completely defined by polynomials. Moreover, it has a bounded state space. Hence one can apply a result by Fränzle [10] which provides an algorithm that, while it does not terminate always, still terminates for all *robust* inputs. Hence we have:

Theorem 5 *Safety verification of the results of* $A(\mathtt{H}, \delta)$ *is quasi-decidable.*

However, it is possible that $A(\mathtt{H}, \delta)$ is not robust—even if $\mathtt{H}$ is robust. In the case of such non-robustness Fränzle's algorithm does not terminate. This can be circumvented:

Theorem 6 *Safety verification of non-linear hybrid systems is quasi-semidecidable*

Proof Let F_t be a version of Fränzle's algorithm [10] for safety verification that, if it terminates within t time units, it return the corresponding (Boolean) result, and otherwise returns false. We use the following algorithm:

```
$i \leftarrow 1$
while there is no $j \in \{1, \ldots, i\}$ such that $F_{2^i}(A(\mathtt{H}, 1/2^j))$
    $i \leftarrow i + 1$
return true
```

This algorithm obviously is correct. It remains to prove termination for robustly safe $\mathtt{H}$.

Due to Theorem 4, if $\mathtt{H}$ is robustly safe, then there is a strictly positive real number δ such that also $[\![A(\mathtt{H}, \delta)]\!]$ is robustly safe. Moreover, due to the nature of Definition 8, also for all $\delta' < \delta$, $[\![A(\mathtt{H}, \delta')]\!]$ is robustly safe. Hence we can choose n such that $[\![A(\mathtt{H}, 1/2^n)]\!]$ is robustly safe. Assume that Fränzle's algorithm (that terminates for all robustly safe inputs) needs time t to prove safety of $[\![A(\mathtt{H}, 1/2^n)]\!]$. Eventually the above algorithm will start $F_{2^i}(A(\mathtt{H}, 1/2^n))$ with 2^i being greater than t which will prove safety. □

4 Quasi-semidecidability of falsification

In this section we will present an algorithm for falsifying safety of hybrid systems. Here we will take the assumption that all terms in the constraints defining hybrid systems are polynomial (i.e., do not contain any function symbols distinct from addition and multiplication).

We are looking for an algorithm that terminates for all robustly unsafe inputs. Recall that robustness is defined based on the notion of ε-perturbed solution sets of constraints. Note that—as a consequence of Definition 8—ε-perturbed solution sets of a flow constraint $\dot{x} = f(x)$ correspond to ε-perturbed solution sets of $\dot{x} \geq f(x) \wedge \dot{x} \leq f(x)$ (see also the discussion and example after the definition). Since in the latter both occurrences of f can be perturbed independently, the empty set is a ε-perturbed solution set of $\dot{x} = f(x)$ and a corresponding perturbed hybrid system may have no flows at all, and hence be vacuously not unsafe.

This corresponds to the fact that modeling a physical system using ordinary differential equations introduces some modeling error that is not captured by the plain ODE $\dot{x} = f(x)$. Hence we expect a user to explicitly include the possible modeling error, for example, by writing inequalities of the form $\dot{x} \geq f(x) - \varepsilon \wedge \dot{x} \leq f(x) + \varepsilon$, for a small but non-zero real constant ε.

For an algorithm for falsifying safety it suffices to abstract to a finite state system. We approximate trajectories using piecewise affine functions. We start with showing how to test whether the affine pieces fulfill the given flow constraint. Here we will use the term "point" to denote valuations in the state space of a given hybrid system, and we call a flow ϕ affine iff for every $v \in V$, ϕ^v is affine (see Definition 2, note that this means that the derivative of ϕ^v is constant).

Definition 10 Two non-identical points p and $\tilde{p}$ with $p(\mathtt{mode}) = \tilde{p}(\mathtt{mode})$ *satisfy a flow constraint* $\mathtt{Flow}$ iff there exists an affine flow ϕ of length t, such that $\phi(0) = p$, $\phi(t) = \tilde{p}$, and for all $s \in [0, t]$, $\phi(s) \bullet \dot{\phi}(s) \models \mathtt{Flow}$. In such a case we also write $p \xrightarrow{\mathtt{Flow}} \tilde{p}$.

```
Input:
– a hybrid systems description
  $(S, \mathtt{Init}, \mathtt{Flow}, \mathtt{Jump}, \mathtt{Unsafe})$,
– a strictly positive real value $\delta$
$G \leftarrow$ set of boxes of diameter $\delta$ covering the state space $S$
$\mathit{Init} \leftarrow \{s(B) \mid B \in G, s(B) \models \mathtt{Init}\}$
$\mathit{Trans} \leftarrow \{\langle s(B), s(B')\rangle \mid B \in G, B' \in G,$
        $s(B) \bullet \mathtt{Prime}(s(B')) \models \mathtt{Jump} \vee s(B) \xrightarrow{\mathtt{Flow}} s(B')\}$
$\mathit{Unsafe} \leftarrow \{s(B) \mid B \in G, s(B) \models \mathtt{Unsafe}\}$
$(\mathit{Init}, \mathit{Trans}, \mathit{Unsafe})$
```

Fig. 3 Under-approximating abstraction

This definition requires the existence of functions (flows), that is, it contains higher-order quantifiers. Such quantifiers cannot directly be handled algorithmically. But, using the fact that the derivative of an affine flow is constant on the whole corresponding line segment, one can replace the higher-order quantifier by a first-order quantifier, that is, a quantifier over real numbers:

Lemma 2 *For two non-identical points p and $\tilde{p}$ such that $p(\mathtt{mode}) = \tilde{p}(\mathtt{mode})$, we have that $p \xrightarrow{\mathtt{Flow}} \tilde{p}$ iff there is a real constant $\lambda > 0$, such that for all points q on the line segment between p and $\tilde{p}$,*

$$q \bullet \left\{\dot{v} \mapsto \lambda\big(\tilde{p}(v) - p(v)\big) \mid v \in V\right\} \models \mathtt{Flow}.$$

The check provided by this lemma is decidable due to our assumption that our constraints defining hybrid systems, and in particular the constraint `Flow`, are polynomial [26]. Hence it can serve as a basis for an algorithm for computing under-approximating abstractions.

In this algorithm—as shown in Fig. 3—we again put a grid of a certain diameter onto the state space. Then, for each box B in this grid we choose a sample point $s(B)$, for example, the midpoint of B, and check the constraints defining the hybrid system on these sample points. Again, this check (which is undecidable for more general constraints [24]) is possible due to our restriction to polynomials.

We denote the result computed by the algorithm by $\check{A}(\mathtt{H}, \delta)$. This is a finite state system $(\mathit{Init}, \mathit{Trans}, \mathit{Unsafe})$. An error trajectory of such a system is a sequence $x_1, \ldots, x_n$, such that $x_1 \in \mathit{Init}$, $x_n \in \mathit{Unsafe}$ and for all $i \in \{1, \ldots, n-1\}$, $\langle x_i, x_{i+1}\rangle \in \mathit{Trans}$.

The algorithm is sound, that is, it in fact computes an under-approximation:

Theorem 7 *For a given hybrid system description $\mathtt{H}$ and $\delta > 0$, if $\check{A}(\mathtt{H}, \delta)$ has an error trajectory, then also $[\![\mathtt{H}]\!]$ has one.*

For proving a bound on the amount of under-approximation, we use the following metric on valuations.

Definition 11 The *distance* between two valuations $\sigma_1 \in \Gamma(X)$ and $\sigma_2 \in \Gamma(X)$ is defined by

$$d(\sigma_1, \sigma_2) \doteq \max_{v \in X}\left\{(d\big(\sigma_1(v), \sigma_2(v)\big)\right\},$$

where

- for modes $m_1, m_2 \in M$, $d(m_1, m_2) = 0$, if $m_1 = m_2$, and ∞, otherwise, and
- for real numbers a_1 and a_2, $d(a_1, a_2) \doteq |a_1 - a_2|$.

Now we prove that it is possible to approximate flows arbitrarily closely by a piecewise affine function with the pieces starting and ending at grid points.

Lemma 3 *Let ϕ be a flow of length t that is both Lipschitz and differentiable. Then, for every $\varepsilon > 0$ there is a $\delta > 0$ such that for every regular grid on the state space S of mesh $\delta > 0$ there is a sequence $\psi_1, \ldots, \psi_k$ of affine flows of length $t_1, \ldots, t_k$ such that for all $i \in \{1, \ldots, k\}$,*

- *if $i < k$, then $\psi_i(t_i) = \psi_{i+1}(0)$,*
- *both $\psi_i(0)$ and $\psi_i(t_k)$ are grid elements, and*
- *for every point $t_\psi \in [0, t_i]$, there is t_ϕ in $[0, t]$ such that $d(\psi_i(t_\psi), \phi(t_\phi)) < \varepsilon$, and $d(\dot{\psi}_i(t_\psi), \dot{\phi}(t_\phi)) < \varepsilon$.*

The intuition of the proof is the following: We decompose ϕ into segments where for each variable, the corresponding slope stays within some interval of bounded size. As a consequence, every line starting near the beginning of the segment and ending near its end has bounded distance from ϕ. This allows us to construct a sequence of lines being close enough to ϕ.

For formalizing this idea (see proof below) we will need the following:

Lemma 4 *For every $\delta > 0$ there is a bound $\beta_\delta > 0$ such that for every flow ϕ of length t with $t < \delta$ and every box D of width δ such that for all $s \in [0, t]$, $\dot{\phi}(s) \in D$, for every affine flow ψ of length t such that $d(\phi(0), \psi(0)) < \delta$ and $d(\phi(t), \psi(t)) < \delta$, the distance of ϕ and ψ, and of their derivatives, is bounded by β_δ. Moreover, the bound β_δ goes to zero as δ goes to zero.*

Proof W.l.o.g. we can assume that $d(\phi(0), \psi(0)) = d(\phi(t), \psi(t)) = \delta$. Here, we only prove the case where $\psi(0) = \phi(0) - \delta$, and $\psi(t) = \phi(t) - \delta$, the other case is dual.

Then (see Fig. 4), the maximal distance between ϕ and ψ is bounded by the maximal distance of two line segments ϕ_a with domain $[0, \frac{t}{2}]$ and ϕ_b with domain $[\frac{t}{2}, t]$, such that

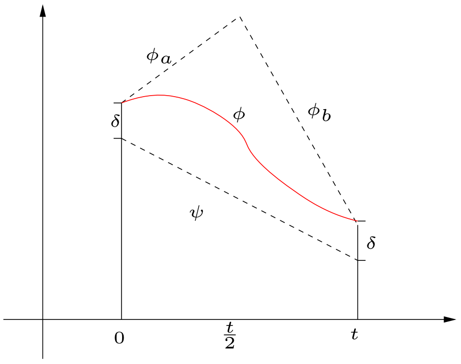

Fig. 4 Maximal distance

$\phi_a(0) = \phi(0)$ and ϕ_a has slope $\max D$, and $\phi_b(t) = \phi(t)$ and ϕ_b has slope $\min D$ (max and min are taken variable-wise). This maximal distance goes to zero with δ.

Moreover, due to the mean-value theorem, ϕ attains the slope of ψ somewhere in the interval $[0, t]$. Hence, for every $s \in [0, t]$, $\dot{\psi}(s) \in D$, and since also $\dot{\phi}(s) \in D$ the distance $d(\dot{\phi}(s), \dot{\psi}(s))$ is bounded and goes to zero with δ. □

Now we are ready to prove Lemma 3:

Proof Let $\varepsilon > 0$ be arbitrary, but fixed. Let δ be such that the β_δ ensured by Lemma 4 is smaller than ε.

Due to Lipschitz continuity of ϕ there are $t_1, \dots, t_k$ and boxes $D_i, \dots, D_k$ such that

- $0 = t_0 < t_1 < \dots < t_k = t$
- for every $i \in \{1, \dots k\}$, $t_i - t_{i-1} < \delta$,
- for every $i \in \{1, \dots, k\}$ the box D_i has width δ and for all $t \in [t_{i-1}, t_i]$, the vector $\dot{\phi}(t) \in D_i$.

Take a grid of mesh δ, and construct $\psi_1, \dots, \psi_k$ as the sequence of affine flows such that for every $i \in \{0, \dots, k\}$, ψ_i has length $t_i - t_{i-1}$, and the i-th vertex consists of a grid element close to $\phi(t_i)$. Due to Lemma 4 the distance between ϕ and $\psi_1, \dots, \psi_k$ is bounded by β_δ, and hence also by ε. □

Now we observe that if a valuation x' is sufficiently close to a valuation x that robustly satisfies a constraint, than x' satisfies the constraint also:

Lemma 5 *Let* $\mathtt{C}$ *be a constraint and let* $\varepsilon > 0$. *Let* x' *be such that there is an* x *with* $d(x, x') \leq \varepsilon$, *such that* x *is an element of all sets that fulfill* $\mathtt{C}$ *up to* ε. *Then* $x' \models \mathtt{C}$.

Proof Since x is in all sets that fulfill $\mathtt{C}$ up to ε, not only $x \models \mathtt{C}$, but also $x \models \mathtt{C}^*$, if $d(\mathtt{C}, \mathtt{C}^*) \leq \varepsilon$. Hence, for all x' with $d(x, x') < \varepsilon$, $x' \models \mathtt{C}$. □

Now we can now state the main theorem of this section:

Theorem 8 *If a hybrid system description* $\mathtt{H}$ *is robustly unsafe, then there is a* $\delta > 0$ *such that* $\check{A}(\mathtt{H}, \delta)$ *is unsafe.*

Before proceeding with the proof of this theorem we note once more that—according to Definition 5—a hybrid system description $\mathtt{H}$ is robustly unsafe iff there is a real number $\varepsilon > 0$ such that all elements of $[\![\mathtt{H}]\!]_\varepsilon$ are unsafe. The elements of $[\![\mathtt{H}]\!]_\varepsilon$ are the hybrid systems H fulfilling the hybrid system description $\mathtt{H}$ up to ε. As already discussed (e.g., directly after Definition 9), this does *not* necessarily mean that H fulfills a hybrid system description that is the result of perturbing $\mathtt{H}$. In particular, as discussed at the beginning of this section, a flow constraint of the form $\dot{x} = c$ has the empty set as a perturbed solution set.

We now prove Theorem 8:

Proof We assume that the finite state system $\check{A}(\mathtt{H}, \delta)$ has the form $(\mathit{Init}, \mathit{Trans}, \mathit{Unsafe})$. Let $\mathtt{H}$ be robustly unsafe with robustness margin ε. Let $\phi_1, \dots, \phi_p$ be a robust error trajectory of $\mathtt{H}$, that is, a trajectory that is an error trajectory of all H that fulfill $\mathtt{H}$ up to ε.

Such a robust error trajectory of $\mathtt{H}$ exists due to the following observation: Consider a constraint $\mathtt{C}$. Let x be such that for all $\mathtt{C}^*$ with $d(\mathtt{C}, \mathtt{C}^*) \leq \varepsilon$, $x \models \mathtt{C}^*$. Due to Definition 8, such an x is in every ε-perturbed solution set of $\mathtt{C}$. The hybrid system containing, for every defining constraint, all those x, has an error trajectory. This is the common error trajectory we need.

Now let $l_1, \dots, l_p$ be the lengths of $\phi_1, \dots, \phi_p$. For each $i \in \{1, \dots, p\}$, ϕ_i satisfies the assumptions of Lemma 3 which ensures a $\delta_i > 0$ corresponding to our robustness margin ε. Choose δ as $\min\{\delta_1, \dots, \delta_i\}$. We will construct an error trajectory of $\check{A}(\mathtt{H}, \delta)$.

Take a grid of mesh δ. By Lemma 3 we know that there is a sequence of affine flows $\psi_1, \dots, \psi_k$ of lengths $l'_1, \dots, l'_k$ whose end-points are grid elements, and such that for every $i' \in \{1, \dots, k\}$ and $t' \in [0, l'_{i'}]$ there is an $i \in \{1, \dots, p\}$, $t \in [0, l_i]$ such that $d(\phi_i(t), \psi_{i'}(t')) < \varepsilon$, and $d(\dot{\phi}_i(t), \dot{\psi}_{i'}(t')) < \varepsilon$. Hence, by robustness of $\mathtt{H}$, and Lemma 5, $\psi_i(0) \xrightarrow{\mathtt{Flow}} \psi_i(l'_{i'})$, and so $\langle \psi_i(0), \psi_i(l'_{i'}) \rangle \in \mathit{Trans}$.

Moreover, due to similar reasoning, for every $i' \in \{1, \dots, k-1\}$, $\langle \psi_{i'}(l'_{i'}), \psi_{i'+1}(0) \rangle \in \mathit{Trans}$, $\psi_0(0) \in \mathit{Init}$, and $\psi_k(l'_k) \in \mathit{Unsafe}$. Hence the endpoints of $\psi_1, \dots, \psi_k$ form an error trajectory of $\check{A}(\mathtt{H}, \delta)$. □

This result, and the fact that $\check{A}(\mathtt{H}, \delta)$ is a finite system and hence algorithmically checkable, together with Theorem 6 proves the main theorem of the paper, as stated at the end of Sect. 2.

5 Related work

A recent article [4, Sect. 5] includes a survey on the role of noise and robustness in continuous-time dynamical systems.

Similar quasi-decidability results as the ones presented in the present paper have been obtained (under different names) for systems with simpler dynamics: Fränzle [10, 11] provides results for the case where the input system is completely defined by polynomials. Especially, continuous evolution is given by explicit polynomial flows which, in general, does not even allow the modeling of linear differential equations, since these can have non-polynomial flows as solutions.

Puri and co-authors [21] show how to compute an over-approximation of Lipschitz differential inclusions with known Lipschitz constant over a finite time horizon. This implies a corresponding quasi-decidability result. In contrast to that, our result allows unbounded time, and does not require a previously known Lipschitz constant.

Collins [7] studies approximation of reach sets of dynamical system in an effective computable analysis framework which again implies a corresponding quasi-decidability result. He uses a discrete time model (such a model can in certain cases encode a continuous time model). In the continuous time case there is corresponding work on approximating reach sets over a finite time horizon [8].

Damm and co-authors [9] provide a similar result as ours for a discrete time model. The continuous time model employed in this paper, implies several additional difficulties:

– When considering syntactic descriptions of systems, in a discrete time model all variables vary over the state space of the system, whereas in a continuous time model, some variables (describing differentiation) do not. Hence these variables may take unbounded values even if the state space is bounded. This needs additional deduction mechanisms for capturing the set of possible values that these variable may take and proofs of their correctness (Theorem 2) and convergence (Lemma 1).

– In a discrete time model, a trajectory only reaches finitely many states in a finite time interval, whereas in a continuous time model it usually reaches uncountably many. This uncountable set has to be captured by corresponding algorithms. As a consequence, in the case of verification, abstraction to a finite state systems, as used in the earlier paper, cannot capture system behavior arbitrarily closely, since even arbitrary refinements cannot separate two sub-sequent steps of the system. In the case of falsification, instead of just having to consider finitely many points (due to state space compactness and discrete time), we have to bound the distance of the abstraction to uncountably many points on an error trajectory.

Studying the effect of perturbations on dynamical systems is a classical research topic for continuous systems, as a summary see for example the textbook by Khalil [17]. However, only recently such have such questions received broader attention in the case of hybrid [13] or even completely discrete systems [1].

On the negative side, Henzinger and Raskin showed that certain undecidability results for hybrid systems continue to hold, even if in the proof one only allows encodings into robust trajectories [16]. This does not contradict our result for two reasons: First, quasi-decidability allows an algorithm that does not always (i.e., for non-robust inputs) terminate, whereas undecidability (even when based on robust trajectories) proves non-existence of an algorithm that *terminates always*. Second, in a similar way as Fränzle [10, 11], we require a compact state space, whereas Henzinger and Raskin do not (although their dynamics is *much* simpler than ours).

Regarding falsification, recent work [3, 6] explores so-called resolution-complete simulation algorithms. These give some completeness assurance but miss a few elements for a full quasi-decidability proof (e.g., the algorithms assume a Lipschitz constant on the function defining the differential equation, and they ignore errors due to time-discretization).

6 Conclusion

We proved that safety-verification of non-linear hybrid systems is quasi-decidable. Some of the algorithms used in the proof of quasi-decidability are not efficient in practice (especially checking robust rectangular hybrid systems). It remains an open problem to find verification algorithms that terminate for robust hybrid systems *and* are efficient in practice. Also, it is open, whether quasi-decidability holds even in the case of a non-compact state space.

A further interesting question is the precise relationship between the syntactic perturbations used in this paper and approaches studying perturbation of hybrid systems based on set-valued analysis [2], especially the application of corresponding results around well-posedness questions for hybrid systems [12, e.g.].

References

1. Asarin E, Bouajjani A (2001) Perturbed Turing machines and hybrid systems. In: Proc LICS'01, pp 269–278
2. Aubin J-P, Frankowska H (1990) Set-valued analysis. Birkhäuser, Boston
3. Bhatia A, Frazzoli E (2007) Sampling-based resolution-complete safety falsification of linear hybrid systems. In: 46th IEEE conference on decision and control, pp 3405–3411
4. Bournez O, Campagnolo ML (2008) A survey on continuous time computations. In: Cooper S, Löwe B, Sorbi A (eds) New computational paradigms. Springer, New York, pp 383–423
5. Caviness BF, Johnson JR (eds) (1998) Quantifier elimination and cylindrical algebraic decomposition. Springer, Berlin

6. Cheng P, Kumar V (2008) Sampling-based falsification and verification of controllers for continuous dynamic systems. Int J Robot Res 27(11–12):1232–1245
7. Collins P (2005) Continuity and computability of reachable sets. Theor Comput Sci 341:162–195
8. Collins P (2011) Semantics and computability of the evolution of hybrid systems. SIAM J Control Optim 49(2):890–925
9. Damm W, Pinto G, Ratschan S (2007) Guaranteed termination in the verification of LTL properties of non-linear robust discrete time hybrid systems. Int J Found Comput Sci 18(1):63–86
10. Fränzle M (1999) Analysis of hybrid systems: an ounce of realism can save an infinity of states. In: Flum J, Rodriguez-Artalejo M (eds) Computer science logic (CSL'99). LNCS, vol 1683. Springer, Berlin
11. Fränzle M (2001) What will be eventually true of polynomial hybrid automata. In: Kobayashi N, Pierce BC (eds) Theoretical aspects of computer software (TACS 2001). LNCS, vol 2215. Springer, Berlin
12. Goebel R, Sanfelice RG, Teel AR (2012) Hybrid dynamical systems: modeling, stability, and robustness. Princeton University Press, Princeton
13. Goebel R, Teel A (2006) Solutions to hybrid inclusions via set and graphical convergence with stability theory applications. Automatica 42(4):573–587
14. Henzinger TA, Ho P-H, Wong-Toi H (1998) Algorithmic analysis of nonlinear hybrid systems. IEEE Trans Autom Control 43:540–554
15. Henzinger TA, Kopke PW, Puri A, Varaiya P (1998) What's decidable about hybrid automata. J Comput Syst Sci 57:94–124
16. Henzinger TA, Raskin J-F (2000) Robust undecidability of timed and hybrid systems. In: Lynch N, Krogh B (eds) Proc HSCC'00. LNCS, vol 1790. Springer, Berlin
17. Khalil HK (2002) Nonlinear systems, 3rd edn. Prentice Hall, New York
18. Moore RE (1966) Interval analysis. Prentice Hall, Englewood Cliffs
19. Neumaier A (1990) Interval methods for systems of equations. Cambridge University Press, Cambridge
20. Puri A (2000) Dynamical properties of timed automata. Discrete Event Dyn Syst 10(1):87–113
21. Puri A, Borkar V, Varaiya P (1996) ε-Approximation of differential inclusions. In: Alur R, Henzinger TA, Sontag ED (eds) Hybrid systems. LNCS, vol 1066. Springer, Berlin
22. Ratschan S (2002) Quantified constraints under perturbations. J Symb Comput 33(4):493–505
23. Ratschan S (2010) Safety verification of non-linear hybrid systems is quasi-semidecidable. In: TAMC 2010: 7th annual conference on theory and applications of models of computation. LNCS, vol 6108. Springer, Berlin, pp 397–408
24. Richardson D (1968) Some undecidable problems involving elementary functions of a real variable. J Symb Log 33:514–520
25. Swaminathan M, Fränzle M, Katoen J-P (2008) The surprising robustness of (closed) timed automata against clock-drift. In: 5th Ifip int conf on theoretical comp sc, pp 537–553
26. Tarski A (1951) A decision method for elementary algebra and geometry. University of California Press, Berkeley. Also in [5]