A Semantic Account of Rigorous Simulation[⋆]

Adam Duracz[1], Eugenio Moggi[2], Walid Taha[3], and Zhenchao Lin[4]

[1] Rice University, Houston, TX, USA, `adam.duracz@rice.edu`
[2] DIBRIS, Genova Univ., Genova, Italy, `moggi@unige.it`
[3] Halmstad Univ., Halmstad, Sweden, `name.surname@hh.se`
[4] Zhejiang University, Hangzhou, China, `cszclin@gmail.com`

Abstract. Hybrid systems are a powerful formalism for modeling cyber-physical systems. Reachability analysis is a general method for checking safety properties, especially in the presence of uncertainty and non-determinism. Rigorous simulation is a convenient tool for reachability analysis of hybrid systems. However, to serve as proof tool, a rigorous simulator must be correct wrt a clearly defined notion of reachability, which captures what is *intuitively* reachable in finite time.
As a step towards addressing this challenge, this paper presents a rigorous simulator in the form of an operational semantics and a specification in the form of a denotational semantics. We show that, under certain conditions about the representation of enclosures, the rigorous simulator is correct. We also show that finding a representation satisfying these assumptions is non-trivial.

Keywords: Reachability analysis; Correctness; Programming languages.

1 Introduction

The crux of the intellectual problem with Cyber-Physical Systems (CPS) is that, for the models that we use for the physical world, such as ODEs or DAEs, there is a huge body of knowledge that has built up since the 19th century on how to model physical systems using these abstractions. In the computing world, we also developed a lot of abstractions over a much shorter history, from the 1930s or so, to talk about computing. And those two classes of abstractions don't play together. Generally, one has a notion of time, the other doesn't. How do you make these systems play together? This is a big intellectual challenge. We are basically trying to take two fabulously developed sets of theories that have diverged, and bring them back together. — Edward A. Lee, 2012.[5]

⋆ Funded by USA National Science Foundation and Swedish Knowledge Foundation. Most of the work on this paper was done when the authors were at Halmstad Univ.

[5] Edward A. Lee, First Halmstad Colloquium, Halmstad Univ., February 10th, 2012. Minutes 1:17-1:20 in video http://bit.ly/HC-EAL, paraphrased slightly for clarity.

No sooner had the term CPS been coined by Helen Gill in 2006 [18] that Lee began, with such characteristic eloquence, to tirelessly inspire multitudes of researchers, including the authors, to address the challenges of modeling Cyber-Physical Systems. For us, the following issues are of particular interest:

1. A mathematics that can cope with both continuous and discrete changes.
2. The possibility of extending to an heterogeneous setting modeling methods and practices developed only for the continuous or the discrete setting.
3. Software tools that can support modeling in an heterogeneous setting.

Hybrid automata [4, 12] and the more general hybrid systems [11] appear to address the first issue. Reachability analysis is an important tool to address safety in both the continuous and discrete setting, and its extension to a broader setting is highly desirable. Other features make reachability analysis attractive. First, it can incorporate both symbolic and numerical methods for solving continuous dynamics, allowing a trade-off between speed and generality. Second, given the broad applicability of the notions of "safe sets" and "bad sets", reachability can be used to analyze the designs of a wide range of cyber-physical systems. Third, because of its similarity to numerical simulation, it has an intuitive appeal for a broad audience and a more gradual learning curve than other formal methods.

Motivated by these observations, the Acumen modeling language [28, 22, 10, 25, 24, 1] allows users to describe hybrid systems that can then be simulated either "traditionally" or "rigorously".

Rigorous simulation [8] uses a time-bounded reachability algorithm that proceeds in fixed size time steps, scanning the time domain from zero to a user-specified end time, and at each step computes an *over-approximation* of the states reachable in that time interval. In [10] rigorous simulation has been used to analyze early-stage designs of Advanced Driver Assistance Systems (ADAS).

1.1 Problem

Validated numerics, including directed rounding and other rigorous methods for programming with floating point numbers, address the question of correctness of numerical methods, and show how interval methods can be used to overcome this problem [21, 23]. However, two other steps are needed to establish correctness of a rigorous simulator (or some other tool for reachability analysis)

– to give a mathematical definition of the set of reachable states, and
– to prove that the tool computes an over-approximation of this set.

For a discrete system reachability is given by the reflexive and transitive closure $\to^*$ of the transition relation $\to$ describing how the state of the system changes at each tick of the clock. What is needed is a generalization, that copes with real time and continuous state spaces.

The category **Top** of topological spaces and continuous maps is an obvious choice, in fact: a set amounts to a discrete space, an Euclidean space (more generally a metric space) comes with the topology generated by its *open balls*, a complete lattice can be equipped with Alexandrov topology or Scott topology.

1.2 Contributions and Organization of this Paper

The main contributions of this paper are the denotational semantics used as a reference to define correctness criteria, the definition of a rigorous simulator in the form of an operational semantics *parameterized* wrt an abstract data-type of timed enclosures, a modular strategy for proving the correctness of an operational semantics with respect to the denotational semantics. The strategy captures our intuitive understanding of how implementation and specification should relate. At the same time, this approach places demands on the abstract data-type of timed enclosures. The rest of the paper is organized as follows:

- Section 2 gives an overview, driven by examples, of rigorous simulation.
- Section 3 gives the denotational semantics of a minimal modeling language, where a *model* is interpreted by a hybrid system [11]. We endorse hybrid systems for their simplicity and generality, in particular they fully support non-determinism, which is essential to model *known unknowns* and *don't care*. Then, we define (time-bounded) reachability in the form of a monotonic map induced by a (timed) transition relation.
- Section 4 describes a rigorous simulator as a small-step operational semantics manipulating *timed enclosure*. We make few connections with Section 3, in order to exemplify a possible interpretation of *timed enclosure*.
- Section 5 gives an interpretation for all entities used by the operational semantics and proves correctness in the form of an *assume-guarantee* result.

2 Hybrid Systems and Rigorous Simulation

In a modeling language hybrid systems can be described as collections of guarded jumps and guarded flows. To use a programming language metaphor, they are if-statements saying when the system should change discretely or continuously. Without going into syntactic details, we can illustrate some key concepts of rigorous simulation with two examples:

1. Saw Tooth: This is a system that climbs continuously at speed a per second until it reaches the height of b, at which point it resets to zero, from where it can resume its continuous climbing behavior. As defaults we will take parameters $a = 1$ and $b = 1$; and as initial value for height $x_0(0) = 0$.
2. Bouncing Ball: This is a system that starts at a certain height and a certain speed and it is subject to a downwards acceleration g until it hits the ground at height zero, at which point it loses energy and bounces with a speed equal to a fraction c of its speed. As defaults we will take parameters $g = 1$ and $c = 0.5$; and as initial values for height $x_0(0) = 1$ and for speed $x_1(0) = 0$.

2.1 Basic Concepts

Rigorous simulation proceeds through time by discrete steps. An key concept in rigorous simulation (or reachability analysis) is that of an *enclosure*, ie, a

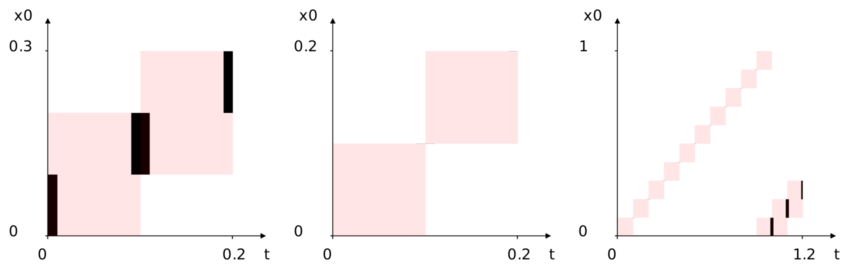

Fig. 1. Simulation steps, Triples, Flows, and Jumps

machine-representable entity that over-approximates the set of states reached within a time interval $[0, h]$ by the system being simulated (or analyzed). Whereas traditional simulation (assumes that the system is deterministic and) produces a *point approximation* for the state reached at the end of a time step, Acumen's rigorous simulation produces a set of *triples*.

The **first plot in Figure 1** illustrates the results for the first two time steps in a simulation of the saw tooth example. To make the visual representation easy to read, we start the system with an initial value that is not a point but a set of possible values, namely, the interval $[0, 0.1]$. Each triple consists of three intervals: the first (black box) over-approximates the set of values at the start, the second (pink box) over-approximates the set of values taken by x_0 during the entire time step, and the third (black box) over-approximates the set of values at the end, or equivalently at the start of the next time step. This plot displays two triples that over-approximate the *trajectory* $x_0(t) = t$ with $t < b$. The second interval in a triple always contains the other two, which give more precise bounds for the start and the end, and help main precision across steps.

A powerful feature of rigorous simulation is the ability to start, work, and compute with sets values. We started with a set of initial values because it is easier to see on the visualization. One can also start with a single initial value, but this exact value can be harder to see. The **second plot in Figure 1** is produced when we start with the value zero. Visualizing triples in this manner allows us to distinguish between uncertainty due to the size of the time step and uncertainty in the set of values being passed from one step to the next. Visualizing triples enables the user to pinpoint the sources of uncertainty in results, be it uncertainty about inputs, due to underlying numerics, or due to the fact that an algorithm is stepping discretely through time. For example, if we allow the Saw Tooth system start from a single initial value and run longer, we can observe some important artifacts of how rigorous simulation deals with discrete events.

In general, the exact time when an event occurs in a continuous system may not be representable nor computable. This means that a rigorous simulation algorithm must reason about what happens when an event occurs at some unknown time within the time step. The **third plot in Figure 1** runs the simulation

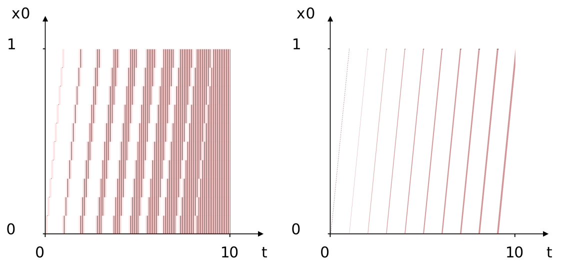

Fig. 2. Jumps, Uncertainty, and Simulation Step Size

longer and shows the results after the first jump in the saw tooth system. The results give a hint of how events are handled. In essence, we consider all values taken by $x_0(t)$ from the start to the end of the time step. Then, we compute the result of the jump from that set. The only question that remains is how long the system can evolve after that point and until the end of the time step. We must work with the worst case, ie, it can evolve for anywhere between zero and the length of the time step. The result is that the final value at the end of the time step is often "blurred" with uncertainty.

The reader may be alarmed that this means that uncertainty can quickly accumulate due to simulation. This is only half true. Errors can also decrease during a simulation. The **two plots in Figure 2** give the simulation of the saw tooth system for ten seconds, one with step size 0.1, the other with step size 0.01. The plots show that a smaller step size can slow the rate at which error is added, but it is unlikely to stop it. There are two features of rigorous simulation that can stop and even reduce error. The first is explicit constraints. For example, in the saw tooth example, even though the each event adds uncertainty, the value of x_0 remains bounded between 0 and 1. This is due to exploiting the information present in the guards to the events using, for examples, the contractor techniques advocated by Jaulin [6, 13]. The second is that when the system being studied has stable dynamics, this dynamics can be used to absorb the uncertainties due to simulation, and the error can eventually become smaller. This means that, as long we are designing stable systems, accurate rigorous simulation should be possible for good designs [9].

2.2 Zeno Behavior

More challenging problems for rigorous simulation arise from the interaction between continuous and discrete dynamics, including Zeno [14, 26, 27, 15] and chattering behavior [19, 17, 2, 3]. In these behaviors there is an infinite number of discrete events (jumps) in a finite amount of time, thus the simulator has to handle an unknown number of jumps within a single time step. Previous work involving some of the authors [16, 15] showed how such systems can nevertheless be rigorously simulated using enclosures, ie, by demonstrating that "no transition

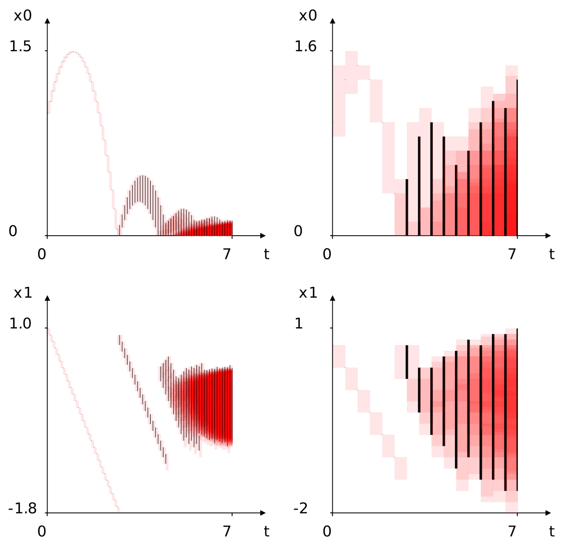

Fig. 3. Rigorous Simulation past Zeno point. Two valid but divergent simulations.

can take the system outside a given enclosure". The bouncing ball is a classic example of a system that can exhibit Zeno behavior.

The **plots on the left of Figure 3** display the height x_0 and the speed x_1 of the ball for the first simulation. Triples generated during a simulation step can overlap, which leads to a red (darker pink) color. In general, a simulation step can generate many triples, due to a wide range of uncertainties, including whether or not a guard is true. For example, the third "falling band" for speed starts before the end of the second falling band. This is because in some of the possible trajectories the ball has already bounced twice, while in some others not. Such uncertainty is natural in the presence of discrete events, and increases close to Zeno points. This increase is captured by the increased intensity of the red color. In this example, the maximum height that the ball reaches after each bounce forms a geometric series. The Zeno point for this example is reached before $t = 7$. Thus, this simulation successfully goes beyond the Zeno point. What is not achieved, however, is to stop the increase of uncertainty. Increasing uncertainty is confirmed by the (slowly) increasing size of the enclosure for both variables as we get closer to $t = 7$. The **plots on the right of Figure 3** confirm this divergence. They show the simulation results for the same system with the same initial conditions but with a bigger time step. Because this system is self-similar as we approach the Zeno point, using a proportionally larger time step is equivalent to zooming in around the Zeno point. The second diagram confirms that the overall size of the enclosures is increasing.

The **plots in Figure 4** show how adding an extra variable x_2 for the (kinetic plus potential) energy of the system (which changes at bounces and stays constant

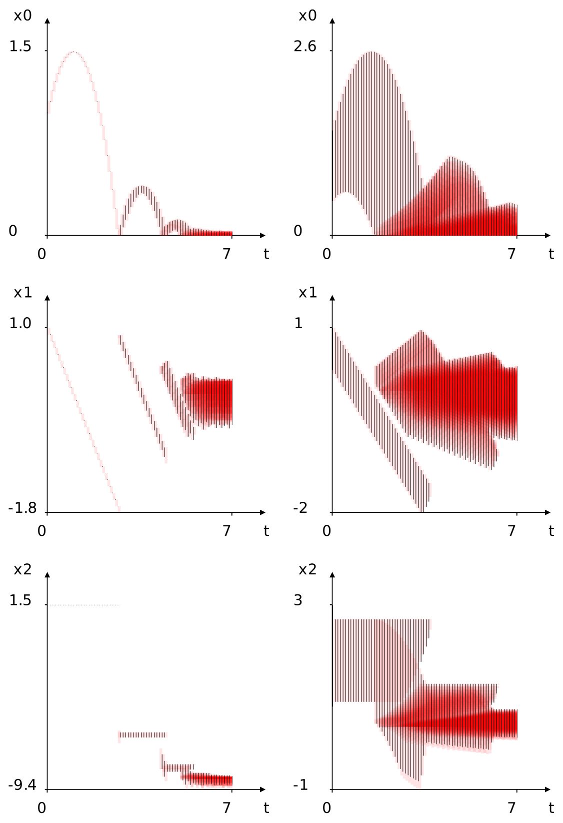

Fig. 4. Rigorous Simulation past Zeno point - valid and convergent enclosures

otherwise) allows to achieve contracting enclosures for the system. To illustrate the robustness of this phenomena, the plots on the right show a simulation for the system but with larger time steps *and* much larger uncertainty about the initial value of the height and speed at the start of simulation. As the graph shows, the enclosures still converge. This is confirmed by the falling energy levels, which strongly limit the set of possible values for height and speed.

3 Denotational Semantics

We define the criteria that a simulator for hybrid systems on an Euclidean space $\mathbb{S} \triangleq \mathbb{R}^n$ must satisfy to be considered *rigorous*, namely it must *over-approximate* the *safe evolution map* E_h (see Def. 3) of the system over an initial segment $[0, h]$ of the continuous time-line $\mathbb{T} \triangleq \{d{:}\,\mathbb{R} | d \geq 0\}$.

In this section we use the cartesian closed category **Po** of complete lattices and monotonic maps, in particular E_h is such a map. **Po** is also the natural setting for defining and comparing *abstract interpretations* [7]. We assume familiarity with the category **Top** of topological spaces and continuous maps, some topological notions (such as open, close and compact subset) and the definition of derivative (in the context of Euclidean spaces).

In the rest of the paper, we write $x\colon X$ for membership $x \in X$, $\mathsf{P}(X)$ for the set of subsets of X, $\mathsf{P_f}(X)$ for the set of finite subsets of X, and make limited use of the category $\mathbf{Set}_p$ of sets and partial maps.

Definition 1 (HS [11]). *A* ***Hybrid System*** *(HS for short) is a pair (F, G) of binary relations on $\mathbb{S}$, respectively called* ***flow*** *and* ***jump*** *relation, its* ***support*** *is given by $\mathsf{S}(F, G) \triangleq \{s \mid \exists s'.s\, F\, s' \vee s\, G\, s' \vee s'\, G\, s\}$. Finally, $\mathbb{H}(\mathbb{S}) \triangleq \mathsf{P}(\mathbb{S}^2)^2$ denotes the complete lattice of HS on $\mathbb{S}$ ordered by component-wise inclusion.*

As customary in mathematical logic, we must interpret syntactic entities by mathematical entities. This *semantic* link is essential to relate the transformations implemented by a computer program (like a simulator) to some mathematical function.

For our purposes it is useful to split the syntax in two layers:

- The upper layer considers modes q as primitive entities, and it suffices to define our denotational and operational semantics and to prove correctness
- The lower layer gives the concrete syntax for modes, which usually depends on the expressions handled by the libraries used, while the cardinality of X determines the Euclidean space $\mathbb{R}^n$ used by the denotational semantics

mode	$q \in \mathsf{Q} ::= \dots$
model	$m \in \mathsf{M} ::= q \mid m_1, m_2$
variable	$x \in X$ finite set
real exp	$e ::= x \mid f(e_i \mid i\colon \#f)$
bool exp	$b ::= p(e_i \mid i\colon \#p) \mid b_1 \wedge b_2 \mid b_1 \vee b_2$
mode	$q \in \mathsf{Q} ::= \mathsf{if}\ b\ \mathsf{flow}\ (x' = e_x \mid x\colon X) \mid \mathsf{if}\ b\ \mathsf{jump}\ (x^+ = e_x \mid x\colon X)$

More generally, flows and jumps could be boolean expressions (with variables $X, \dot{X}$ and X, X^+ respectively) denoting binary relations on $\mathbb{R}^n$.

We interpret a mode $q\colon \mathsf{Q}$ by a HS $[\![q]\!]\colon \mathbb{H}(\mathbb{S})$, and extend the interpretation to models m and sets Q of modes by taking component-wise union, eg

$$[\![m_1, m_2]\!] \triangleq [\![m_1]\!] \cup [\![m_2]\!] = (F_1 \cup F_2, G_1 \cup G_2) \text{ when } [\![m_i]\!] = (F_i, G_i)\colon \mathbb{H}(\mathbb{S})$$

Example 1. We describe a simple system with a parameter $b\colon [0, 1]$, namely a timer v with a timeout u, that exhibits a Zeno behaviour when $b\colon (0, 1)$. Its description as a model m_T is q_0, q_1, q_2, where

$q_0 = \mathsf{if}\ 0 < v < u\ \mathsf{flow}\ v' = 1, u' = 0$	timer increases as time flows
$q_1 = \mathsf{if}\ 0 < v = u\ \mathsf{jump}\ v^+ = 0, u^+ = bu$	timer reset to 0 and timeout updated
$q_2 = \mathsf{if}\ v = u = 0\ \mathsf{jump}\ v^+ = 0, u^+ = 1$	timeout reset to 1

Its description as a HS $\mathcal{H}_T = (F, G) = [\![m_T]\!]$ on $\mathbb{R}^2$ is

$$F = \{((v, u), (1, 0))|0 < v < u\} \quad G = \{((u, u)(0, bu))|0 < u\} \uplus \{((0, 0), (0, 1))\}$$

For cardinality reasons it is impossible to have finitary representations for all elements of an Euclidean space $\mathbb{S}$. In a complete lattice, like $\mathsf{P}(\mathbb{S})$, the order allows us to tell when (the interpretation of) a representation approximates an element. Similar considerations motivate the use of interval arithmetic.

Definition 2 (TR). *The* ***transition relation*** $\xrightarrow[(F,G)]{} : \mathsf{P}(\mathbb{S} \times \mathbb{T} \times \mathbb{S})$ *of a HS is* $s \xrightarrow[(F,G)]{d} s' \overset{\Delta}{\iff} d = 0 \wedge s\,G\,s'$ *or* $d > 0 \wedge \exists f \colon \mathbf{Top}([0, d], \mathbb{S})$ *such that*

- *the derivative* $\dot{f}$ *of* f *is defined and continuous in* $(0, d)$
- $s = f(0)$*,* $s' = f(d)$ *and* $\forall t \colon (0, d).f(t)\,F\,\dot{f}(t)$*.*

In the later case we say that f realizes *the transition.*

The transition relation allows to define the *safe evolution map*, which computes an over-approximation of the states reachable at a given time by a HS, even when the HS has Zeno behaviors.

Definition 3 (Safe evolution). *Let* $\mathsf{C}(\mathbb{S})$ *be the complete lattice of* ***closed subsets*** *of a topological space* $\mathbb{S}$ *(ordered by inclusion). The time-bounded* ***transition*** *map* $\mathsf{T}_h \colon \mathbf{Po}(\mathbb{H}(\mathbb{S}) \times \mathsf{P}([0], h] \times \mathbb{S}), \mathsf{P}([0, h] \times \mathbb{S}))$ *and* ***safe evolution*** *map* $\mathsf{E}_h \colon \mathbf{Po}(\mathbb{H}(\mathbb{S}) \times \mathsf{P}([0, h] \times \mathbb{S}), \mathsf{C}([0, h] \times \mathbb{S}))$ *are given by*

- $\mathsf{T}_h(H, I) \overset{\Delta}{=} \{(t + d, s')|\exists s \colon \mathbb{S}.(t, s) \colon I \wedge s \xrightarrow[H]{d} s' \wedge t + d \leq h\}$
- $\mathsf{E}_h(H, I) \overset{\Delta}{=}$ *the smallest* $E \colon \mathsf{C}([0, h] \times \mathbb{S})$ *such that* $I \cup \mathsf{T}_h(H, E) \subseteq E$*.*

Remark 1. The map $\mathsf{T}_h(\mathcal{H}, -)$ corresponds to the binary relation $R_{\mathcal{H}}$ on $[0, h] \times \mathbb{S}$ st $(t, s)R_{\mathcal{H}}(t', s') \overset{\Delta}{\iff} (0 \leq t \leq t' \leq h) \wedge s \xrightarrow[\mathcal{H}]{t'-t} s'$. In fact $\mathsf{T}_h(\mathcal{H}, I) = R_{\mathcal{H}}(I)$. $R^*_{\mathcal{H}}(I)$, where $R^*_{\mathcal{H}}$ is the reflexive and transitive closure of $R_{\mathcal{H}}$, captures only what is reachable from I in finitely many transitions, but may fail to capture what is reachable in finite time. The safe evolution map $\mathsf{E}_h(\mathcal{H}, I)$ avoids this pitfall by requiring E to be a closed subset (see [20]). An equivalent definition of $\mathsf{E}_(\mathcal{H}, I)$ in terms of $R_{\mathcal{H}}$ is the smallest $E \colon \mathsf{P}([0, h] \times \mathbb{S})$ such that $I \cup R_{\mathcal{H}}(E) \cup \overline{E} \subseteq E$, and E is closed because $E \subseteq \overline{E}$ is always true.

In a metric space there is another reason to use $\mathsf{C}(\mathbb{S})$ instead of $\mathsf{P}(\mathbb{S})$. If the accuracy to *discriminate* among points in $\mathbb{S}$ is δ, then a subset $S \colon \mathsf{P}(\mathbb{S})$ cannot be distinguished from the open subset $B(S, \delta) \overset{\Delta}{=} \{s'|\exists s \colon S.d_{\mathbb{S}}(s, s') < \delta\}$. But $S \subseteq \overline{S} \subseteq B(S, \delta)$, where $\overline{S}$ is the *closure* of S, ie, the smallest $S' \colon \mathsf{C}(\mathbb{S})$ containing S. Therefore, one cannot distinguish two subsets of $\mathbb{S}$ with the same closure, no matter how small δ is.

Example 2. Let $\mathcal{H}$ be $\mathcal{H}_T$ in Example 1, then

– The transition relation $\xrightarrow[\mathcal{H}]{}$ is $(v,u) \xrightarrow{d} (v{+}d,u)$ when $0 \leq v < v{+}d \leq u$, $(u,u) \xrightarrow{0} (0,bu)$ when $0 < u$, and $(0,0) \xrightarrow{0} (0,1)$; in particular, one has $(0,u) \xrightarrow{u} (u,u) \xrightarrow{0} (0,bu)$ when $0 < u$.
– The relation $R_{\mathcal{H}}$, which determines the map $\mathsf{T}h(\mathcal{H},-)$, is

$$\{((t,v,u),(t+d,v+d,u))|0 \leq t < t+d \leq h \wedge 0 \leq v < v+d \leq u\} \uplus$$
$$\{((t,u,u),(t,0,bu))|0 \leq t \leq h \wedge 0 < u\} \uplus \{((t,0,0),(t,0,1))|0 \leq t \leq h\}$$

thus $s_0 R^*_{\mathcal{H}} s_n = (t_n, 0, b^n u)$ when $0 < b, u$ and $t_n = \sum_{i:n} b^i u \leq h$, but the Zeno point $s_\omega = (t_\omega, 0, 0)$ is not reachable, even when $t_\omega = \sum_{i:\omega} b^i u \leq h$.
– The set $E = \mathsf{E}_h(\mathcal{H}, I)$ includes $R^*_{\mathcal{H}}(I)$, $s_\omega : E$ when $\forall n : \omega . s_n : E$, because s_ω is the limit of a sequence $(s_n | n : \omega)$ in E, and $R^*_{\mathcal{H}}(s_\omega) \subseteq E$ when $s_\omega : E$. However, if $\mathcal{H}$ is modified so that the $(0,0) \xrightarrow{0} (0,1)$ is removed or replaced by $(0,0) \xrightarrow{0} (0,0)$, then the system cannot progress, ie $R^*_{\mathcal{H}}(s_\omega) = \{s_\omega\}$.

A minimal requirement for a simulator used for *safety analysis* should be *partial correctness* wrt E_h. Namely, given a symbolic description m of a HS and a representation *over-approximating* a set $I : \mathsf{P}(\mathbb{S})$ of initial states the simulator should either fail or compute an *over-approximation* of $\mathsf{E}_h([\![m]\!], [0] \times I)$, or of the bigger set $\mathsf{E}_h([\![m]\!], [0] \times \overline{I})$ (see the above considerations on indistinguishability). The following result is relevant to prove correctness in Section 5.

Lemma 1. *Let F_i and G_j denote the HS $(F_i, \varnothing)$ and $(\varnothing, G_j)$ on $\mathbb{S}$, then*

1. $\xrightarrow[F_i \cup G_j]{} = \xrightarrow[F_i]{} \cup \xrightarrow[G_j]{}$
2. $\xrightarrow[G_0 \cup G_1]{} = \xrightarrow[G_0]{} \cup \xrightarrow[G_1]{}$
3. $\xrightarrow[F_0 \cup F_1]{} = \xrightarrow[F_0]{} \cup \xrightarrow[F_1]{}$ *, if $\overline{\mathsf{S}(F_0)}$ and $\overline{\mathsf{S}(F_1)}$ are disjoint subsets of $\mathbb{S}$*

where $\overline{S}$ is the ***closure*** *of $S : \mathsf{P}(\mathbb{S})$, ie, the smallest $S' : \mathsf{C}(\mathbb{S})$ such that $S \subseteq S'$.*

Proof. We prove only the last claim. If $f : \mathbf{Top}([0,d], \mathbb{S})$ realizes $s \xrightarrow[F_0 \cup F_1]{d} s'$, then the image of f is a subset of $\overline{\mathsf{S}(F_0 \cup F_1)} = \overline{\mathsf{S}(F_0)} \uplus \overline{\mathsf{S}(F_1)}$. By taking the inverse image of the two disjoint closed subsets we get a partitioning of $[0,d]$ in two disjoint closed subsets, but $[0,d]$ is connected, so one of them is empty. $\square$

Remark 2. The lemma says that the transition relation of the union of two HS $\mathcal{H}_0$ and $\mathcal{H}_1$ (on the same state space) is the union of their transition relations only if the flow relations of the two HS are *apart*, ie $\overline{\mathsf{S}(F_0)}$ and $\overline{\mathsf{S}(F_1)}$ are disjoint. If $s \xrightarrow[F]{d} s'$, then s and s' belong to the same connected component of $\overline{\mathsf{S}(F)}$. Given a flow relation F and a connected component C of $\overline{\mathsf{S}(F)}$, let F_C be the restriction of F to C. By definition the flow relations F_C are pairwise apart and $F = \bigcup_C F_C$, thus $\xrightarrow[F]{} = \bigcup_C \xrightarrow[F_C]{}$. Therefore, the connected components of $\overline{\mathsf{S}(F)}$ could be viewed as the *control modes* of a hybrid automaton [5].

4 Operational semantics

The operational semantics uses some auxiliary domains and maps, which form an abstract data-type (ADT). To establish correctness of the operational semantics this ADT must satisfy certain properties. Here we give properties of the ADT that do not refer directly to the denotational semantics, in Section 5 we give more properties that make direct reference to the denotational semantics.

Enclosures. D is a countable set of **enclosures** d interpreted as closed subsets $[\![d]\!]\colon \mathsf{C}(\mathbb{S})$. We assume that D is closed wrt binary intersection $d_1 \cap d_2$, contains the empty enclosure $\varnothing$, and the **cover relation** $d \leq_{\mathsf{D}} [d_i|i\colon n] \overset{\Delta}{\iff} [\![d]\!] \subseteq \bigcup_{i:n} [\![d_i]\!]$ is decidable (we drop the subscript when it is clear from the context). The inclusion relation on D is definable as $d' \subseteq d \overset{\Delta}{\iff} d' \leq [d]$ and $\leq$ extends to a pre-order on D^*, namely $D' \leq D \overset{\Delta}{\iff} \forall d'\colon D'.d' \leq D$.
A possible choice for D is the set of *P-boxes* in $\mathbb{R}^n$, ie, cartesian products of n closed intervals $[x, y]$, whose endpoints are in a countable subset P of $\mathbb{R}$, eg the subset of rational numbers or the finite subset of floating point numbers.

Timed enclosures. The operational semantics uses only an ADT Z of **timed enclosures**, representing over-approximations for closed subsets of $\mathbb{T} \times \mathbb{S}$. An *Acumen-like* implementation is $\mathsf{Z} \subset \mathsf{D} \times \mathsf{D} \times \mathsf{D}$ containing *initial* $e(z) = b(z) = \varnothing$ and *proper* $i(z), e(z) \subseteq b(z)$ triples, where $i(z)$, $b(z)$ and $e(z)$ denotes the three components of a $z\colon \mathsf{Z}$. The interpretation $[\![z]\!]_h\colon \mathsf{P}(\mathsf{C}([0, h] \times \mathbb{S}))$ is given by

$$C\colon [\![z]\!]_h \overset{\Delta}{\iff} C(0) \subseteq [\![i(z)]\!] \wedge (\forall t\colon (0, h).C(t) \subseteq [\![b(z)]\!]) \wedge C(h) \subseteq [\![e(z)]\!]$$

where $C(t) \overset{\Delta}{=} \{s|(t, s)\colon C\}$ when $C\colon \mathsf{C}([0, h] \times \mathbb{S})$.
Z inherits from D intersection, defined component-wise, and the **cover relation** $z \leq_{\mathsf{Z}} [z_i|i\colon n] \overset{\Delta}{\iff} \forall C\colon [\![z]\!]_h.\exists C'\colon \prod_{i:n} [\![z_i]\!]_h.C = \bigcup_{i:n} C'_i$ (the derived notions of inclusion and the pre-order $\leq$ on Z^* are defined as in the case of D)

Theorem 1. *The following decision procedure $\leq'$ is sound for the cover relation on Z, ie, $z \leq' Z \implies z \leq_{\mathsf{Z}} Z$, and the converse holds when $\forall d\colon \mathsf{D}.[\![d]\!]$ is convex*

$$\begin{aligned} z \leq' Z \overset{\Delta}{\iff} & \textit{ if } b(z) = \varnothing \textit{ (ie, } z \textit{ is initial) then } i(z) \leq_{\mathsf{D}} [i(z')|z'\colon Z] \\ & \textit{else } b(z) \leq_{\mathsf{D}} [b(z')|z'\colon Z \wedge b(z') \cap i(z) \subseteq i(z')] \textit{ and} \\ & \quad b(z) \leq_{\mathsf{D}} [b(z')|z'\colon Z \wedge b(z') \cap e(z) \subseteq e(z')] \end{aligned}$$

Proof. Soundness means $C\colon [\![z]\!]_h \wedge z \leq' [z_i|i\colon n] \implies \exists C'\colon \prod_{i:n} [\![z_i]\!]_h.C = \bigcup_{i:n} C'_i$.
The case "z initial" is trivial, otherwise fix $0 < 0' < h' < h$ and let $C'_i \overset{\Delta}{=} C \cap \bigcup(\{[0, h'] \times b(z_i)|b(z_i) \cap i(z) \leq i(z_i)\} \cup \{[0', h] \times b(z_i)|b(z_i) \cap e(z) \leq e(z_i)\})$. □

Remark 3. In general $[\![z]\!]_h$ is downward closed and closed wrt finite unions, but may not have a biggest element, except when z is $(d, \varnothing, \varnothing)$ or (d, d, d).

Jumping. $\mathsf{Jump}\colon \mathbf{Set}_p(\mathsf{Q}\times\mathsf{D},\mathsf{D})$ interprets jumps. $\mathsf{Jump}(q,d)\uparrow$ when it cannot compute an enclosure of the states reachable by jumping with q from d. We assume the following properties

O.J Jump is *strict*, ie $\mathsf{Jump}(q,\varnothing)=\varnothing$, and
monotonic in d, ie $d'\subseteq d\wedge \mathsf{Jump}(q,d)\downarrow\implies \mathsf{Jump}(q,d')\subseteq\mathsf{Jump}(q,d)$.

Flowing. $\mathsf{Flow}_h\colon \mathbf{Set}_p(\mathsf{Q}\times\mathsf{D},\mathsf{Z})$ interprets flows for time step h. $\mathsf{Flow}_h(q,d)\uparrow$ when it cannot compute a timed enclosure of the states reachable by flowing with q from d. We assume the following properties

O.F Flow_h is *strict*, *monotonic* in d, and
the flow starts from d, ie $z=\mathsf{Flow}_h(q,d)\implies z=\mathsf{Flow}_h(q,d\cap i(z))$.

Jump and Flow_h are extended from D to Z as follows (and the extensions inherit the properties assumed for the original maps, like strictness and monotonicity)

- $\mathsf{Jump}(q,z)=z'\overset{\Delta}{\iff} z'=(\mathsf{Jump}(q,i(z)),\mathsf{Jump}(q,b(z)),\mathsf{Jump}(q,e(z)))$
- $\mathsf{Flow}_h(q,z)=z'\overset{\Delta}{\iff}$ if $b(z)=\varnothing$ then $z'=z'_i$ else $z'=(i(z'_i),d'_b,d'_b)$ where $z'_i\overset{\Delta}{=}\mathsf{Flow}_h(q,i(z))$ and $d'_b\overset{\Delta}{=}b(\mathsf{Flow}_h(q,b(z)))$.

Operational rules. Fix a finite set $Q\colon\mathsf{P_f}(\mathsf{Q})$ of modes.
A **Q-set** is a sequence $W\colon(\mathsf{Z}\times\mathsf{P_f}(Q)\times\mathsf{P_f}(Q)\times\mathsf{P_f}(Q))^*$ such that

$$\forall(z,Q_a,Q_d,Q_c)\colon W.\varnothing\subset z\wedge Q_a\uplus Q_d\uplus Q_c\subseteq Q$$

- W is **initial** $\overset{\Delta}{\iff}\forall(z,Q_a,Q_d,Q_c)\colon W.Q_a=Q$.
- W is **terminal** $\overset{\Delta}{\iff}\forall(z,Q_a,Q_d,Q_c)\colon W.Q_a=\varnothing$.

For defining the operational semantics we make the following assumptions

O.# $\forall q\colon Q.(\forall z\colon\mathsf{Z}.\mathsf{Jump}(q,z)=\varnothing)\vee(\forall z\colon\mathsf{Z}.\mathsf{Flow}_h(q,z)=\varnothing)$, thus Q is partitioned in flows Q_F (ie, modes that cannot jump) and the rest Q_J (that cannot flow)
O.Q $\forall q,q'\colon Q.\forall z,z'\colon\mathsf{Z}.z'=\mathsf{Flow}_h(q,z)\wedge q\neq q'\implies\mathsf{Flow}_h(q',z')=\varnothing$, this says at the level of timed enclosures that flows in Q are *apart* (see Remark 2).

The binary relation $\xrightarrow[Q]{}$ on Q-sets is defined by the following rules

jump $W,(z,Q_a\uplus q,Q_d,Q_c),W'\xrightarrow[Q]{}W,(z,Q_a,q\uplus Q_d,Q_c),W',(z',Q-q,\varnothing,\varnothing)$
if $\varnothing\subset\mathsf{Jump}(q,z)=z'$
flow $W,(z,Q_a\uplus q,Q_d,Q_c),W'\xrightarrow[Q]{}W,(z,Q_a,q\uplus Q_d,Q_c),W',(z',Q-q,\varnothing,\varnothing)$
if $\varnothing\subset\mathsf{Flow}_h(q,z)=z'$
done $W,(z,Q_a\uplus q,Q_d,Q_c),W'\xrightarrow[Q]{}W,(z,Q_a,q\uplus Q_d,Q_c),W'$
if $\mathsf{Flow}_h(q,z)=\mathsf{Jump}(q,z)=\varnothing$
cover $W,(z,Q_a\uplus q,Q_d,Q_c),W'\xrightarrow[Q]{}W,(z,Q_a,Q_d,q\uplus Q_c),W'$
if $z\leq[z'|(z',Q'_a,Q'_d,Q'_c)\colon W,W'\wedge q\in Q'_d]$.

Remark 4. The side conditions of (jump), (flow) and (done) are mutually exclusive by (O.#). The following rule is derivable by exploiting (O.#) and (O.Q)

flow* $W, (z, Q_a \uplus q, Q_d, Q_c), W' \xrightarrow[Q]{} W, (z, Q_a, q \uplus Q_d, Q_c), W', (z', Q_J, Q_F - q, \varnothing)$
if $\varnothing \subset \mathsf{Flow}_h(q, z) = z'$, where Q_J and Q_F are defined in assumption (O.#).

The assumptions (O.#) and (O.Q) can be recast in terms of D

O.#* $\forall q{:}\, Q.(\forall d{:}\, \mathsf{D}.\mathsf{Jump}(q, d) = \varnothing) \vee (\forall d{:}\, \mathsf{D}.\mathsf{Flow}_h(q, d) = \varnothing)$
O.Q* $\forall q, q'{:}\, Q.\forall d{:}\, \mathsf{D}.\forall z'{:}\, \mathsf{Z}.z' = \mathsf{Flow}_h(q, d) \wedge q \neq q' \implies \mathsf{Flow}_h(q', b(z')) = \varnothing$

but the operational rules (and the proof of correctness) treat Z as an ADT, thus one can adopt a different implementation of Z without invalidating correctness, provided all assumptions are cast in terms of Z.

5 Correctness

The operational semantics is defined on top of the ADT Z for timed enclosures, thus its correctness is an assume-guarantee result of the form "if the ADT Z satisfies certain properties, then the operational semantics is correct".

Assumptions We fix $Q{:}\, \mathsf{P_f}(\mathsf{Q})$, define $(F_q, G_q) = \llbracket q \rrbracket{:}\, \mathbb{H}(\mathbb{S})$ (see Section 3), and make the following assumptions, in addition to (O,#) and (O.Q) of Section 4

A.Z $\forall z{:}\, \mathsf{Z}.\llbracket z \rrbracket_h \subseteq \mathsf{C}([0, h] \times \mathbb{S})$ is downward closed and has a top element $C(z)$
A.# $\forall q{:}\, Q.F_q = \varnothing \vee G_q = \varnothing$, ie, $\overline{\llbracket q \rrbracket}$ is either a jump or a flow
A.Q $\forall q, q'{:}\, Q.q \neq q' \implies \overline{\mathsf{S}(F_q)} \cap \overline{\mathsf{S}(F_{q'})} = \varnothing$, ie, $\overline{\llbracket Q \rrbracket}$ is a hybrid automaton (see Remark 2), since $\mathsf{S}(\overline{F}) = \overline{\mathsf{S}(F)}$
A.J $\forall q{:}\, Q.\forall z, z'{:}\, \mathsf{Z}.\mathsf{Jump}(q, z) = z' \wedge C{:}\, \llbracket z \rrbracket_h \implies \mathsf{E}_h(\overline{G_q}, \mathsf{T}_h(\overline{G_q}, C)){:}\, \llbracket z' \rrbracket_h$
A.F $\forall q{:}\, Q.\forall z, z'{:}\, \mathsf{Z}.\mathsf{Flow}_h(q, z) = z' \wedge C{:}\, \llbracket z \rrbracket_h \implies \mathsf{E}_h(\overline{F_q}, \mathsf{T}_h(\overline{F_q}, C)){:}\, \llbracket z' \rrbracket_h$.

In the sequel we write $q(C)$ for $\mathsf{T}_h(\overline{\llbracket q \rrbracket}, C)$ and $q^+(C)$ for $\mathsf{E}_h(\overline{\llbracket q \rrbracket}, q(C))$, where $q{:}\, Q$ and $C{:}\, \mathsf{C}([0, h] \times \mathbb{S})$. By Lemma 1 the assumptions (A.#) and (A.Q) imply $\xrightarrow[\overline{\llbracket Q' \rrbracket}]{} = \bigcup_{q:Q'} \xrightarrow[\overline{\llbracket q \rrbracket}]{}$, or equivalently $\mathsf{T}_h(\overline{\llbracket Q' \rrbracket}, C) = \bigcup_{q:Q'} q(C)$, when $Q' \subseteq Q$.

Remark 5. The assumptions (A.J) and (A.F) refer to the extensions of Jump and Flow_h to Z. Section 4 implements Z using a simpler ADT D, but the operational rules refer only to Z, thus correctness holds, as far as the assumptions on Z hold. **Warning:** the simple implementation of Z in terms of D defined in Section 4 satisfies a weaker property than (A.Z), see Remark 3, thus we cannot claim correctness for an operational semantics using that implementation.

To state correctness we have to define the semantics of Q-sets W, this is done coherently with the semantics of timed enclosures $z{:}\, \mathsf{Z}$.

Definition 4. *The semantics* $\llbracket W \rrbracket_h{:}\, \mathsf{P}(\mathsf{C}([0, h] \times \mathbb{S}))$ *for a Q-set W is*

– $[\![(z, Q_a, Q_d, Q_c)]\!]_h \triangleq [\![z]\!]_h \colon \mathsf{P}(\mathsf{C}([0,h] \times \mathbb{S}))$
– $[\![W]\!]_h \triangleq \{\bigcup_{i:n} C_i | C \colon \prod_{i:n} [\![W(i)]\!]_h\}$, *with* $n = |W|$ *and* $W(i)$ *i-th item in* W.

Correctness says that the operational semantics computes over-approximations of the safe evolution map. However, the computation may fail to terminate, there is no bound on the accuracy of the over-approximations, termination and accuracy may depend on the order in which the operational rules are applied.

Theorem 2 (Correctness). *If* $W \xrightarrow[Q]{*} W'$ *with* W *initial and* W' *terminal, then* $\forall C \colon [\![W]\!]_h . \mathsf{E}_h(\overline{[\![Q]\!]}, C) \colon [\![W']\!]_h$.

Correctness relies on a lemma saying that $\xrightarrow[Q]{}$ preserves *well-formed* Q-sets.

Definition 5. *Let* $(z(i), Q_a(i), Q_d(i), Q_c(i)) = W(i)$ *and* $Q(i) = Q_a(i) \uplus Q_d(i) \uplus Q_c(i)$ *for* $i \colon n = |W|$, *then* W *is a* ***well-formed*** Q*-set* $\overset{\triangle}{\iff} \exists p \colon n \rightharpoonup n \times Q$ *such that*

1. $p(i) = (j, q) \implies j < i$, *ie,* n *forms a forest with arcs* $j \xrightarrow{q} i$
2. $p(i) = p(j) \implies i = j$
3. $i \colon n \wedge p(i) \uparrow \implies Q(i) = Q$
4. $p(i) = (j, q) \wedge q \colon Q_J \implies q \colon Q_d(j) \wedge \mathsf{Jump}(q, z(j)) = z(i) \wedge Q(i) = Q - q$
5. $p(i) = (j, q) \wedge q \colon Q_F \implies q \colon Q_d(j) \wedge \mathsf{Flow}_h(q, z(j)) = z(i) \wedge Q(i) = Q - q$
6. $q \colon Q_d(j) \implies \mathsf{Jump}(q, z(j)) = \mathsf{Flow}_h(q, z(j)) = \varnothing \vee \exists i \colon n . p(i) = (j, q)$
7. $q \colon Q_c(j) \implies z(j) \leq [z(i) | i \colon n \wedge q \colon Q_d(i)]$.

In particular, an initial W *is well-formed by taking* p *such that* $\forall i \colon n . p(i) \uparrow$.

The partial map $p \colon n \rightharpoonup n \times Q$ records how items were added to W, ie, $p(i) = (j, q)$ means that $W(i)$ was added by applying (flow) or (jump) to remove q from $Q_a(j)$.

Lemma 2. *If* W *is well-formed and* $W \xrightarrow[Q]{} W'$, *then* W' *is well-formed.*

Proof. By case analysis on the operational rule used to derive $W \xrightarrow[Q]{} W'$. The proof relies on the assumption (O.#) and the side-conditions of the operational rules. In particular, the witness p' that W' is well-formed is given by the witness p for W in the cases (done) and (cover), while it is an extension of p in the cases (jump) and (flow). □

Lemma 3. *If* W *is well-formed and terminal, then* $\mathsf{E}_h(\overline{[\![Q]\!]}, D) \colon [\![W]\!]_h$, *where* $D = \bigcup \{C(z(i)) | i \colon n \wedge p(i) \uparrow\}$ *with* $n = |W|$ *and* p *witness that* W *is well-formed.*

Proof. Define $C' \colon [\![W]\!]_h$ such that $D \subseteq C'$ and $\mathsf{T}_h(\overline{[\![Q]\!]}, C') = \bigcup_{q:Q} q(C') \subseteq C'$, therefore $\mathsf{E}_h(\overline{[\![Q]\!]}, D) \subseteq C'$ belongs to $[\![W]\!]_h$, because $[\![W]\!]_h$ is downward closed. For $i \colon n$ let $\varnothing \subset C_i \triangleq C(z(i))$ the top element in $[\![W(i)]\!]_h$ by (A.Z), $C \triangleq \bigcup_{i:n} C_i$ the top element in $[\![W]\!]_h$, $C'_i \triangleq C_i$ when $p(i) \uparrow$, $C'_i \triangleq q^+(C_j)$ when $p(i) = (j, q)$, and $C' \triangleq \bigcup_{i:n} C'_i$, then the following properties hold

1. $D \subseteq C'$, by definition of D and C'
2. $C' \subseteq C$, because W is well-formed and $C'_i = q^+(C_j) \subseteq C_i$ when $p(i) = (j, q)$, by (A.#), (A.J), (A.F) and definition of C_i
3. $\forall j\colon n.\forall q\colon Q_d(j).q^+(C_j) \subseteq C'$, because W is well-formed and $q^+(C_j) = \varnothing$ or $\exists i\colon n.p(i) = (j, q) \wedge q^+(C_j) = C'_i$
4. $\forall i\colon n.\forall q\colon Q_c(i).q^+(C_i) \subseteq C'$, because $C_i \subseteq \bigcup\{C_j | j\colon n \wedge q\colon Q_d(j)\}$ by W well-formed, and $q^+(C_i) = \bigcup\{q^+(C_j) | j\colon n \wedge q\colon Q_d(j)\} \subseteq C'$ by point 3
5. $\forall i\colon n.\forall q\colon Q(i).q(C_i) \subseteq C'$, by the points 3 and 4, because $q(C_i) \subseteq q^+(C_i)$ and $Q(i) = Q_d(i) \uplus Q_c(i)$ by W terminal
6. $\forall q\colon Q.q(C') \subseteq C'$. We prove $\forall i\colon n.\forall q\colon Q.q(C'_i) \subseteq C'$ by case analysis on $i\colon n$:
   - if $p(i) \uparrow$, then $Q(i) = Q$ and $C'_i = C_i$, thus $\forall q\colon Q.q(C'_i) \subseteq C'$ by point 5
   - $p(i) = (j, q)$, then $Q(i) = Q - q$ and $C'_i = q^+(C_j)$, thus $q(C'_i) \subseteq C'_i \subseteq C'$ by definition of C'_i, and $\forall q'\colon Q - q.q'(C'_i) \subseteq C'$ by point 5. □

6 Conclusions and Future Work

The main contribution of the paper is an assume-guarantee proof of correctness (see Sec 5) for the rigorous simulator defined in Sec 4, where the assumptions concern an ADT Z of timed enclosures. The proof may serve as a blueprint for similar results. For instance, one could replace safe evolution with a variant which is *robust* wrt arbitrary small over-approximations of the hybrid system and the set of initial states (see [20]), or strengthen the correctness guarantees by specifying the accuracy of the over-approximation computed by the rigorous simulator (this means that accuracy becomes a parameter for the simulator, the auxiliary maps Jump and Flow_h, and the statement of correctness).

We showed that a simple implementation of Z, defined in terms of an ADT D (see Sec 4), does not satisfy the assumption that the interpretation $[\![z]\!]_h$ has a top element, or more precisely that $[\![z]\!]_h$ is a principal ideal in $\mathsf{C}([0, h] \times \mathbb{S})$ ordered by inclusion. Thus, an important next step will be to determine whether there can be an implementation of Z satisfying all assumptions. It will also be interesting to see if there is an alternative proof of correctness that rely on the weaker assumption that $[\![z]\!]_h$ is only an ideal.

We sketch an implementation of Z satisfying all assumptions required by the proof of correctness, in particular for each $z\colon \mathsf{Z}$ the ideal $[\![z]\!]_h$ has a top element $E(z)$ and $[\![z]\!]_h = \{C\colon \mathsf{C}([0, h] \times \mathbb{S}) | C \subseteq E(z)\}$. The basic idea is that $E(z)$ is a convex bounded polytope P in $\mathbb{T} \times \mathbb{R}^n$ such that $P(t)$ is a box in $\mathbb{R}^n$ for each $t\colon \mathbb{T}$. More formally, we take as $z\colon \mathsf{Z}$ sequences of inequalities of the form $a \leq t \leq b$ (where $0 \leq a, b \leq h$) or $a + a't \leq x_i \leq b + b't$ with rational coefficients and involving $n + 1$ variables, namely t for time and x_i for the i-th state variable.

A sequence z of inequalities defines a closed convex subset $E(z)$ of $\mathsf{C}([0, h] \times \mathbb{S})$ consisting of the points satisfying all inequalities (thus $E(z)$ is a polytope), and it is bounded when z includes an inequality $a \leq t \leq b$ and at least one inequality for each x_i. Finally, the inclusion and cover relation for convex polytopes described by conjunctions of linear inequalities with rational coefficients are decidable, because they are Turing-reducible to linear programming.

References

1. Acumen. http://acumen-language.org, 2016.
2. Ayman Aljarbouh and Benoît Caillaud. On the regularization of chattering executions in real time simulation of hybrid systems. In *Baltic Young Scientists Conference*, page 49, 2015.
3. Ayman Aljarbouh, Yingfu Zeng, Adam Duracz, Benoît Caillaud, and Walid Taha. Chattering-free simulation for hybrid dynamical systems. In *2016 IEEE International Conference on Computational Science and Engineering, IEEE International Conference on Embedded and Ubiquitous Computing, and International Symposium on Distributed Computing and Applications to Business, Engineering and Science*. IEEE Computer Society, 2016.
4. Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical computer science*, 138:3–34, 1995.
5. Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.
6. Gilles Chabert and Luc Jaulin. Contractor programming. *Artificial Intelligence*, 173(11):1079–1100, 2009.
7. Patrick Cousot and Radhia Cousot. Abstract interpretation frameworks. *Journal of logic and computation*, 2(4):511–547, 1992.
8. Adam Duracz. *Rigorous Simulation: Its Theory and Applications.* PhD thesis, Halmstad University Press, 2016.
9. Adam Duracz, Ferenc A Bartha, and Walid Taha. Accurate rigorous simulation should be possible for good designs. In *Symbolic and Numerical Methods for Reachability Analysis (SNR), 2016 International Workshop on*, pages 1–10. IEEE, 2016.
10. Adam Duracz, Henrik Eriksson, Ferenc A. Bartha, Yingfu Zeng, Fei Xu, and Walid Taha. Using rigorous simulation to support ISO 26262 hazard analysis and risk assessment. In *12th International Conference on Embedded Software and Systems (ICESS)*, pages 1093–1096. IEEE, 2015.
11. Rafal Goebel, Ricardo G Sanfelice, and A Teel. Hybrid dynamical systems. *Control Systems, IEEE*, 29(2):28–93, 2009.
12. T. A. Henzinger. The theory of hybrid automata. In *Logic in Computer Science*, pages 278–292, New Brunswick, NJ, 1996. IEEE Computer Society.
13. Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics.* Springer Verlag, 2001.
14. Karl Henrik Johansson, Magnus Egerstedt, John Lygeros, and Shankar Sastry. On the regularization of zeno hybrid automata. *Systems & control letters*, 38(3):141–150, 1999.
15. Michal Konečnỳ, Walid Taha, Ferenc A. Bartha, Jan Duracz, Adam Duracz, and Aaron D Ames. Enclosing the behavior of a hybrid automaton up to and beyond a Zeno point. *Nonlinear Analysis: Hybrid Systems*, 20:1–20, 2016.
16. M. Konečný, W. Taha, J. Duracz, A. Duracz, and A. Ames. Enclosing the behavior of a hybrid system up to and beyond a zeno point. In *1st IEEE International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, pages 120–125, 2013.

17. Edward A. Lee and Haiyang Zheng. Operational semantics of hybrid systems. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control: 8th International Workshop, Zurich, Switzerland. Proceedings*, pages 25–53, Berlin, Heidelberg, 2005. Springer.
18. Edward Ashford Lee and Sanjit A Seshia. *Introduction to embedded systems: A Cyber-Physical Systems approach.* MIT Press, 2016.
19. John Lygeros. Lecture notes on hybrid systems, section 4.2. In *Notes for an ENSIETA workshop*, 2004.
20. E. Moggi, A. Farjudian, A. Duracz, and W. Taha. Safe & Robust Reachability Analysis of Hybrid Systems. *ArXiv e-prints*, September 2017. https://arxiv.org/abs/1709.05658.
21. Ramon E. Moore. *Interval analysis*, volume 4. Prentice-Hall, 1966.
22. Walid Taha, Paul Brauner, Yingfu Zeng, Robert Cartwright, Veronica Gaspes, Aaron Ames, and Alexandre Chapoutot. A core language for executable models of cyber-physical systems (preliminary report). In *32nd International Conference on Distributed Computing Systems*, pages 303–308. IEEE, 2012.
23. Warwick Tucker. *Validated numerics: a short introduction to rigorous computations.* Princeton University Press, 2011.
24. Yingfu Zeng, Ferenc Bartha, and Walid Taha. Compile-time extensions to hybrid odes. In Erika Ábrahám and Sergiy Bogomolov, editors, Proceedings 3rd International Workshop on *Symbolic and Numerical Methods for Reachability Analysis,* Uppsala, Sweden, volume 247 of *Electronic Proceedings in Theoretical Computer Science*, pages 52–70. Open Publishing Association, 2017.
25. Yingfu Zeng, Rose Chad, Walid Taha, Adam Duracz, Kevin Atkinson, Roland Philippsen, Robert Cartwright, and Marcia O'Malley. Modeling electromechanical aspects of Cyber-Physical Systems. *Journal of Software Engineering for Robotics*, 7(1):100–119, 2016.
26. Jun Zhang, Karl Henrik Johansson, John Lygeros, and Shankar Sastry. Zeno hybrid systems. *International journal of robust and nonlinear control*, 11(5):435–451, 2001.
27. Haiyang Zheng, Edward A. Lee, and Aaron D. Ames. Beyond Zeno: Get on with It! In *Proceedings of the 9th International Conference on Hybrid Systems: Computation and Control*, pages 568–582, Berlin, Heidelberg, 2006. Springer.
28. Yun Zhu, Edwin Westbrook, Jun Inoue, Alexandre Chapoutot, Cherif Salama, Marisa Peralta, Travis Martin, Walid Taha, Marcia O'Malley, Robert Cartwright, et al. Mathematical equations as executable models of mechanical systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 1–11. ACM, 2010.