Crossing the bridge between similar games[⋆]

Jan-David Quesel, Martin Fränzle, Werner Damm

University of Oldenburg, Department of Computing Science, Germany
{quesel,fraenzle,damm}@informatik.uni-oldenburg.de

Abstract. Specifications and implementations of complex physical systems tend to differ as low level effects such as sampling are often ignored when high level models are created. Thus, the low level models are often not exact refinements of the high level specification. However, they are similar to those. To bridge the gap between those models, we study robust simulation relations for hybrid systems. We identify a family of robust simulation relations that allow for certain bounded deviations in the behavior of a system specification and its implementation in both values of the system variables and timings. We show that for this relaxed version of simulation a broad class of logical properties is preserved. The question whether two systems are in simulation relation can be reduced to a reach avoid problem for hybrid games. We provide a sufficient condition under which a winning strategy for these games exists.

Keywords: formal verification of hybrid systems, robust simulation, logics for hybrid systems, hybrid games

1 Introduction

Hybrid systems provide a mathematical model for systems with interacting discrete and continuous dynamics. Examples include controllers for trains, cars, and airplanes. In many cases, these controllers are critical for the system operation and safety. Thus, verification is a crucial task during the design of those systems. Unfortunately, verification approaches do not scale well enough to directly tackle the whole implementation of such controllers.

Different approaches have been taken to overcome this issue so far. Refinement between a relaxed model and a synthesized discrete implementation is established by Stauner [21]. His method relies on models that obey certain restrictions. For example the invariants and guards must be overlapping in a single point. Stauner then constructs a sampled implementation that is a refinement of a relaxed version of the model where the guards and invariants overlap in a larger region. The gap from timed systems to implementations was bridged by De Wulf et.al. [25] in a similar way. They provide a relaxed semantics for timed automata that ensures implementability and preservation of LTL properties. Girard et. al. [12] presented an approach how to construct approximately bisimilar symbolic models for switched dynamical systems.

[⋆] This work was partially supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS).

Thrane et. al. [22] studied different types of simulations that allow for some deviations of the costs in weighted timed automaton. A notion of simulation in the presence of spatial deviations that allows to related hybrid systems with identical control graphs is presented by Girard et. al. [11].

As effects like sampling that often occur in implementations influence the timing behavior similarity notions have been studied that allow for some deviation in the timing behavior as well. Davoren [5] presented an approach generalizing Skorokhod-metrics and provided conditions under which these initially pseudo-metrics induce topologies which are Hausdorff and are thus indeed metrics. However, she does not present a constructive method for determining the values assigned to two concrete system by the metric.

Inspired by the simulation notion presented by Girard et. al. [11] and the more general similarity notion in Davoren's work [5], (a) we study a simulation relation that allows one system to simulate another in a robust way, where deviations in continuous variable valuations, and in timing behavior are subject to constant bounds, and (b) we then investigate properties preserved under such a simulation relation. Additionally, we drop the requirement inherent to the work in [11] that the discrete behavior must have the same control graph.

We model hybrid systems using a notion of hybrid automata [13]. However, the formalism allows for some specifications that lack real-world realizability. As we want to bridge the gap between specifications and implementations, we do neither consider runs of a system that are Zeno, i.e. an infinite number of transitions is taken within a finite amount of time, nor behaviors that are time blocking. Time blocking means that at a certain point in time, no future evolution of the system is possible due to an invariant preventing any continuous evolution and no transition guard being satisfied. Also, as valuations of transient intermediate variables in internal calculations are not observable, we consider systems similar as long as their outputs are similar, even if transient intermediate values differ.

To determine whether two systems are related by our simulation relation, we build up a two player hybrid game where the existence of a winning strategy for the second player coincides with the fact that the systems are in simulation relation. Hybrid games are a natural extension of timed games [18] where derivatives are no longer restricted to 1 and resets can exhibit a more complex structure. Different types of hybrid games have been studied in the literature so far. Restricted classes of hybrid games have been proven to be decidable (see e.g. [14, 2, 24]). Unfortunately, the games expressive enough for our purpose do not fall into such a class. Tomlin et. al. [23] study hybrid games for controller synthesis. They give an algorithm how to compute controllable predecessors and thus checking if there is a controller that drives the system into a safe state.

To express properties of real-time as well as hybrid systems a variety of different logics have been proposed (see [1] for a survey). We choose to use a variant of the future fragment of MTL [16] to specify properties of our systems. As basic propositions we use expressions that are evaluated on the system variables. This gives us a nice partitioning between temporal and spatial propositions on a syntactical level which we will exploit when proving that certain properties are preserved by our simulation relation. Henzinger et. al. [15] studied the relation between a timed variant of CTL and a notion of simulation that allows for some bounded deviation in the timing behavior of the timed automaton under consideration. They prove that a certain modification function applied to the temporal operators of their logic ensures that properties can be transfered using this simulation relation. We will use a similar transformation but add some modifications to the basic propositions as well to capture the deviations in space.

With Fainekos and Pappas [7] and Donzé and Maler [6] we share the goal of defining robust satisfaction of linear-time metric temporal logic formulas in a form permitting the generalization of findings obtained on one trajectory to "close-by" trajectories, where "close-by" refers to both space—as already addressed by Fainekos and Pappas—and time—a combination also covered by Donzé and Maler. Our work, however, is notably different from theirs in that they deal with instance properties while we deal with ensemble properties: while theirs robustly evaluates a given formula over a given (sampled) trajectory, we are concerned with computing a simulation relating *every* trajectory of a given concrete system to a robustly corresponding abstract counterpart such that the robust semantics of *every* temporal-logic formula is preserved up to a given tolerance. The latter constitutes a natural notion of system refinement which can help bridge the gap between the abstract models used in system verification and their actual implementations, while the former is the adequate setting for assessing observed trajectories against a given set of requirements.

Contributions. We study a similarity of hybrid systems. In contrast to many other approaches, we do not restrict ourselves to classical refinement, but develop a notion of similarity that relates systems based on behavioral distance so that we can transfer properties even if the behavior is slightly different. We allow the system behavior to differ in both valuations of the system variables as well as timings. We give a sufficient criterion under which a system A is simulated by another system B using our notion of simulation and prove that for all formulas that are satisfied by B there is a "similar" formula that is satisfied by A.

Structure of this Paper. In Sect. 2 we give the formal basis for specifying hybrid systems and properties of those. Section 3 provides a motivating example for the study of system similarity which is examined further in Sect. 4. In Sect. 5 we give sufficient conditions under which we can assert that systems are similar and study what properties are preserved by this relation in Sect. 6.

2 Basics

To specify systems, we use a standard notion called hybrid automata. The syntax is similar to the syntax for hybrid automata originally defined by Henzinger in [13].

Definition 1. *A hybrid automaton is a tuple* $H = (U, X, L, E, F, Inv, Init)$ *where*

- *$V := U \dot{\cup} X$ is a set of real-valued variables where U is the set of external variables and X contains the internal ones.*
- *L is the set of locations.*
  - *Invariants are provided by a mapping Inv of locations to predicates over variables in V.*
  - *Flows are given by F, which is a mapping of locations to predicates of the form $\bigwedge_{x \in \dot{X}} \dot{x} = e_x$ where e_x are expressions over V. Here, $\dot{X}$ denotes the derivatives of the variables in X.*
- *$E \subseteq L \times G \times L$ are discrete transitions, where G denotes predicates over $V \cup X'$. The variables in X' are valuated with the values of the variables in the post-state.*
- *$Init$ is a mapping of locations to predicates over V, which characterizes the initial condition to start in the specific location.*

We use a hybrid time model that is the cross product of the non-negative real numbers (denoted by $\mathbb{R}$ for real numbers and $\mathbb{R}^+$ for the non-negative subset) and the natural numbers (denoted by $\mathbb{N}$). The natural number component allows arbitrarily many discrete jumps at a single real-valued point in time. Further, we use $|\cdot|$ to denote the cardinality of sets or the absolute value of numbers and $\dot{\cup}$ to denote the disjoint union of sets.

The semantics of a hybrid automaton is given by a set of runs. A *run* maps a state to each point in time, which is pair of a non-negative real number and a natural number. A *state* consists of a discrete location and a valuation of the system variables. Every run of the system starts in some location and has a valuation of the variables that satisfies the initial condition. If there is a discrete jump, meaning an increase of the second argument leads to a changed valuation, then there was an edge leading from the previous to the current location, whose guard was satisfied. This also means that the current valuation of variables are subject to certain restrictions based on this guard. Continuous flows range over real-valued time-points. They change the variables based on the solution of the flow predicate. While time is passing, the variable values change continuously, and each of the valuations has to satisfy the location invariant.

Let π denote the projection to specific components of our hybrid automaton. For example $\pi_V(s)$ gives us the valuation of the variables in state s whereas $\pi_L(s)$ returns the location.

Definition 2. *The semantics of a hybrid automaton is given by a set of runs generated by a function $\Xi : ((\mathbb{R}^+ \times \mathbb{N}) \to \mathbb{R}^{|U|}) \to 2^{(\mathbb{R}^+ \times \mathbb{N}) \to L \times \mathbb{R}^{|X|}}$. Here, for some input stream ι, which is a total function $\iota : (\mathbb{R}^+ \times \mathbb{N}) \to \mathbb{R}^{|U|}$, we say the output stream ω is possible for ι, i.e. $\omega \in \Xi(\iota)$, iff there is a run, i.e. total function, $\xi(x, y) := (\iota(x, y), \omega(x, y))$ such that:*

1. *$\xi(0, 0) \models Init$*
2. *If $\xi(t, n + 1) \neq \xi(t, n)$ then there is an edge from $l = \pi_L(\xi(t, n))$ to $l' = \pi_L(\xi(t, n+1))$ and $\pi_V(\xi(t, n))$ satisfies the invariant of l and $\pi_V(\xi(t, n+1))$ satisfies the invariant of l'. Additionally, the guard on this edge is satisfied by using the values of $\xi(t, n)$ for V and $\pi_X(\xi(t, n + 1))$ for X'.*

3. *There is an upper bound on the number of discrete changes at each real-valued point in time:* $\forall t : \exists n : \forall n' > n : \xi(t,n) = \xi(t,n')$. *For each* t, *we call the smallest* n *for which this condition holds* n_t.
4. *During continuous evolutions from some real-valued time point* t *to some* t' *the location stays constant and no discrete computations change any values. They start after the discrete computations stabilized, i.e. at* (t, n_t), *and all intermediate states* t'' *satisfy the invariant of the current location:* $\forall t : \exists t' > t : \forall t < t'' \leq t' : \pi_L(\xi(t'',0)) = l \wedge (t'' < t' \rightarrow \forall n'' : \xi(t'',0) = \xi(t'',n'')) \wedge \xi(t'',0) \models Inv(l) \wedge \pi_V(\xi(t'',0)) = s(t''-t)$ *where* $l = \pi_L(\xi(t,n_t))$ *and* s *is a solution to the initial value problem* $s(0) = \pi_V(\xi(t,n_t))$ *of the flow predicate* $F(\pi_L(\xi(t,n)))$.

Note that this definition of the semantics explicitly excludes runs where there is an infinite number of different states at a single real-valued point in time. It, thus, ensures that every instantaneous discrete calculation comes to a result after a finite number of steps. However, Zeno behavior can still occur, as the delays between calculation steps might go to zero in the limit. As such systems would not be implementable, we assume in the following that those effects do not occur. For example the models could be altered in a way that between every two discrete transitions there is some small constant delay.

To describe properties of hybrid systems we need a logic that is able to express temporal relations of real-valued states. Therefore, we study the following real-valued real-time linear time temporal logic that we call $\mathcal{L}\natural$ in this paper.

First, let us define what a Lipschitz continuous function is.

Definition 3 (Lipschitz continuity). *We say that a function* $f : \mathbb{R}^n \rightarrow \mathbb{R}$ *is Lipschitz continuous, iff there is some constant* M *such that for all* $x_1, \ldots, x_n$, *and all* $y_1, \ldots, y_n$ *holds:*

$$|f(x_1,\ldots,x_n) - f(y_1,\ldots,y_n)| \leq M \cdot ||(x_1,\ldots,x_n),(y_1,\ldots,y_n)|| \ ,$$

where $||\cdot,\cdot||$ *denotes the Euclidean distance. We call the smallest* M *that has this property the Lipschitz constant of* f.

Now, based on this definition to restrict the possible basic terms, the syntax of our logic is defined as follows:

Definition 4 (Syntax). *The basic formulas are defined by*

$$\phi ::= x \in \mathcal{I} \mid f(x_1,\ldots,x_n) \leq 0 \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \, \mathbb{U}_{\mathcal{J}} \, \phi_2$$

where $\mathcal{I} \subseteq \mathbb{R}$, $\mathcal{J} \subseteq \mathbb{R}^+$, f *is a Lipschitz continuous function and the* x_i *are variables.*

We interpret this metric variant of LTL on runs of hybrid systems. However, as transient states of the automaton are not observable in the real world, we project the runs on a continuous domain. This gives us a valuation function which maps a value to each variable at each real-valued point in time.

Definition 5 (Valuation). *We define the valuation of a variable x at time t on a run ξ as*

$$\zeta_\xi(t, x) := \lim_{n\to\infty} \xi(t, n)|_x \enspace ,$$

where $y|_x$ denotes the projection of the vector y to its component associated with the variable name x.

If the run is obvious from the context we just write $\zeta(t, x)$.

Note that this definition is well defined as property 3 of our semantics of hybrid automaton demands that the number of discrete changes to the variables is finite at each real-valued point in time. For the original semantics of hybrid automaton by Henzinger [13] the limit might not exist, as a countably infinite number of transitions might be taken at a real-valued time-point and each might assign new values to the variables. However properties of these runs cannot be observed in the real world, and we, thus, drop them from our focus.

In Fig. 1 an example trajectory of a hybrid system is shown. Here, squares mark values that are omitted by our valuation function.

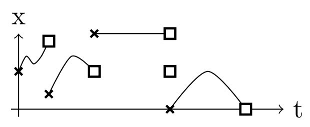


Fig. 1: Example trajectory

We now turn our focus to the semantics of the logical formulas. The semantics of the basic terms imposes bounds on the valuations of the system variables. This can either be done by restricting the values of a variable to be within some real-valued set or by imposing a Lipschitz continuous constraint on multiple variables. The variables are connected to the system state using the valuation function just defined. Boolean connectives are defined as usual, and the until operator provides us with the possibility to express both the temporal order of states as well as postulate time bounds on these orders. The set annotation forces the postcondition to hold at some point in time that lies within this set.

Definition 6 (Semantics). *We define for a run ξ and some $t \in \mathbb{R}^+$ the semantics of a formula ϕ by:*

$$\xi, t \models x \in \mathcal{I} \textit{ iff } \zeta(t, x) \in \mathcal{I} \tag{1}$$

$$\xi, t \models f(x_1, \dots, x_n) \le 0 \textit{ iff } f(\zeta(t, x_1), \dots, \zeta(t, x_n)) \le 0 \tag{2}$$

$$\xi, t \models \neg\phi \textit{ iff not } \xi, t \models \phi \tag{3}$$

$$\xi, t \models \phi \wedge \psi \textit{ iff } \xi, t \models \phi \textit{ and } \xi, t \models \psi \tag{4}$$

$$\xi, t \models \phi \,\mathbb{U}_\mathcal{J}\, \psi \textit{ iff } \exists t' \in \mathcal{J} : \xi, t' + t \models \psi \textit{ and } \forall t \le t'' < t' + t : \xi, t'' \models \phi \tag{5}$$

Additionally we define for a set of runs Ξ:

$$\Xi, t \models \phi \textit{ iff for all runs } \xi \in \Xi \textit{ holds } \xi, t \models \phi \tag{6}$$

A hybrid system Sys satisfies a formula denoted by $Sys \models \phi$ iff $\Xi_{Sys}, 0 \models \phi$.

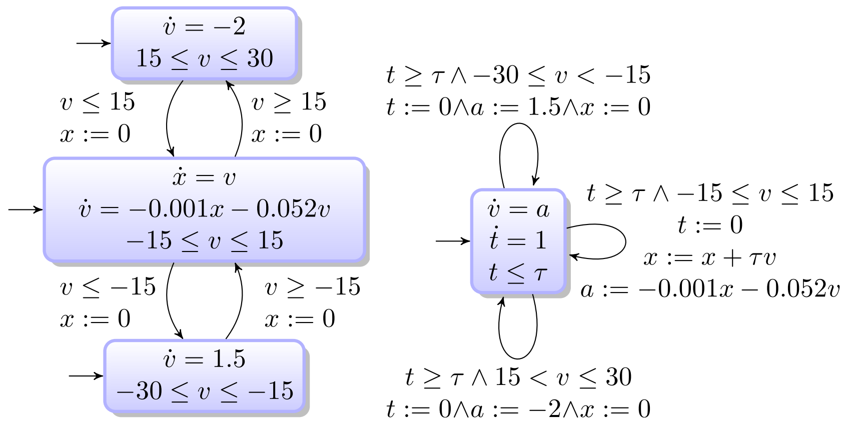
Obviously, *true* and *false* can be expressed in various ways using this logic. Additionally, we define the *eventually* modality by $\Diamond_\mathcal{J}\phi \equiv true\,\mathbb{U}_\mathcal{J}\,\phi$ and the *always* modality as abbreviation $\Box_\mathcal{J}\phi \equiv \neg\Diamond_\mathcal{J}\neg\phi$.

3 Running Example

To further motivate the study of similarity and to illustrate how our results can be applied, we present a specification of a cruise controller taken from [4] and provide a very basic implementation that we use for comparison.

The goal of the cruise controller is to stabilize the system at a velocity difference of $v = 0$ to some target velocity within a certain time bound if it was started with a velocity difference v between -30 and 30. If the velocity difference is below -15 the controller just chooses the maximum acceleration 1.5. In the range of -15 to 15 a proportional-integral (PI) controller takes over. It controls the acceleration proportional to the current and the accumulated velocity difference since this mode was entered. The velocity difference is accumulated by integration over v. We, here, write the integral implicitly as differential equation $\dot{x} = v$. The integral part is used by the PI controller to smoothen the velocity trajectory as it is approaching its target, i.e. a velocity difference of $v = 0$.

If the difference is above 15 the controller enforces braking with maximal deceleration of -2. Figure 2a shows the corresponding automaton. The model consists of three modes. Depending on the initial velocity difference, exactly one of these is enabled. The variable x is used to track the integral over v such that the PI control used in the central mode can access this data.



2a: Original specification   2b: Implementation

Fig. 2: Cruise controller variants

A sampling implementation is provided by the automaton depicted in Fig. 2b. Here, a single mode is sufficient and the acceleration is updated depending on the current velocity difference with a sampling rate of $\tau := 10$ time units. The two systems do not produce identical trajectories and the implementation is not a classical refinement of the specification. Still, they are similar in their behavior.

4 Similarity

To introduce a notion of similarity, we compare the observable values of the system trajectories. Like in the definition of the logic, we restrict our focus to a single valuation of variable at each real-valued point in time. Now, the idea is to say that two runs are similar if both evolve in a similar fashion by comparing variable valuations of one system with valuations of the other that might be shifted in time. This means given a valuation of the variables of one system at some point in time, there is within close distance in time a point where the valuation of the variables of the other system is close. We restrict the temporal distance by a constant ε and the spatial distance by another constant δ.

Definition 7. *For two streams $\sigma_i : \mathbb{R}^+ \times \mathbb{N} \to \mathbb{R}^p$ with $i \in \{1, 2\}$, given two non-negative real numbers ε, δ, we say that σ_1 is ε-δ-simulated by stream σ_2 (denoted by $\sigma_1 \trianglelefteq^{\varepsilon,\delta} \sigma_2$) iff there is some left-total, surjective relation $\mathfrak{r} \subseteq \mathbb{R}^+ \times \mathbb{R}^+$ with*

$$\forall (t, \tilde{t}) \in \mathfrak{r} : |t - \tilde{t}| < \varepsilon \wedge \forall (t', \tilde{t}') \in \mathfrak{r} : (t \leq t' \to \tilde{t} \leq \tilde{t}') \tag{7}$$

and

$$\forall (t, \tilde{t}) \in \mathfrak{r} : ||c(\sigma_1)(t), c(\sigma_2)(\tilde{t})|| < \delta \tag{8}$$

where for $k \in \{1, 2\}$: $c(\sigma_k)$ is defined by $c(\sigma_k)(t) := \lim_{q\to\infty} \sigma_k(t, q)$.

As $\mathfrak{r}$ allows stretching or compressing the time line, we call it a *retiming* relation. An example for such a relation is depicted in Fig. 3. The relation is motivated by the fact that slight variations in the switching points of the system as those variations should not endanger the safety of any robust real-world system.

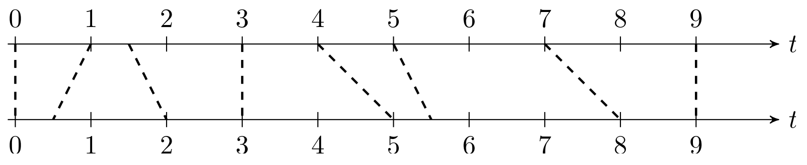


Fig. 3: Example for a retiming relation $\mathfrak{r}$

In some cases it is sufficient to use a slightly weaker notion of similarity. This can be obtained by dropping the bound on the temporal distance. If we do not impose an upper bound on the temporal distance, we can still get useful insights. When comparing systems where the timing behavior is of limited interest, to prove that, for instance, the one system works within certain spatial bounds compared to the other one, it is sufficient to use the following notion of similarity.

Definition 8. *For two streams $\sigma_i : \mathbb{R}^+ \times \mathbb{N} \to \mathbb{R}^p$ with $i \in \{1, 2\}$, given a non-negative real number δ, we say that σ_1 is weakly δ-simulated by stream σ_2*

(denoted by $\sigma_1 \trianglelefteq^{\delta} \sigma_2$) iff there is some left-total, surjective relation $\mathfrak{r} \subseteq \mathbb{R}^+ \times \mathbb{R}^+$ with

$$\forall (t, \tilde{t}) \in \mathfrak{r} : \forall (t', \tilde{t}') \in \mathfrak{r} : (t \leq t' \rightarrow \tilde{t} \leq \tilde{t}') \tag{9}$$

and (8) holds.

We apply the same definitions to output streams as well, by ignoring the location component of those. Now we introduce the main similarity notion on hybrid systems used in this paper using the notion of the stream similarity.

Definition 9. *A hybrid system A is ε-δ-simulated by another system B (denoted by $A \trianglelefteq^{\varepsilon,\delta} B$) iff for all input streams ι_A and for all input streams ι_B for which $\iota_A \trianglelefteq^{\varepsilon,\delta} \iota_B$ and for all output streams $\omega_A \in \Xi(\iota_A)$ of A, there is an output stream $\omega_B \in \Xi(\iota_B)$ of B such that $\omega_A \trianglelefteq^{\varepsilon,\delta} \omega_B$ holds. And similarly, A is weakly δ-simulated by B (denoted by $A \trianglelefteq^{\delta} B$) iff the above conditions hold for weak δ-simulations on the streams.*

Note that if a system has no input variables, then it still has a possible input stream of type $\mathbb{R}^+ \times \mathbb{N} \rightarrow \emptyset$. Therefore, in the absence of input variables, the whole relation is determined by the system outputs.

5 Determining Similarity

In this section we present a sufficient criterion for determining whether two systems are similar. We assume that the system inputs can be described by differential equations. This way we can add these differential equations to each mode and further assume inputless systems.

Now, we give an encoding of the question whether a system A is ε-δ-simulated by a system B into a two player game. The idea is that if there is a winning strategy for the second player then the systems are in simulation relation.

First, we give the general definition of hybrid games.

Definition 10 (Hybrid Game). *A* hybrid game *$HG = (S, E_c, U_c, l)$ consists of a hybrid automaton $S = (U, X, L, E, F, Inv, Init)$, a set of controllable transitions $E_c \subseteq E$, a set of controllable variables $U_c \subseteq U$, and a location $l \in L$.*

The game is played on the states of the hybrid system denoted by S. The possible moves of the first player are determined by the uncontrollable transitions $E \setminus E_c$ and the corresponding invariants and guards. The second player plays on the controllable transitions E_c. In addition, the second player always proposes a function that gives the future valuations of the variables in U_c until the next move is determined. At every state of the game each player chooses an action that is either a finite number of discrete transitions (uncontrollable transitions for the first player and controllable ones for the second player) or a time period they want to let pass. If both players choose discrete transitions then the first player gets precedence and all transitions of the first player are executed. Afterwards the transitions proposed by the second player are executed, if they are still enabled. If both players choose to let time pass, the smaller amount of time is taken. In *case one chooses a discrete transition and the other one chooses to let time pass, the discrete transition gets precedence.*

The first player wins, if he can force the game to enter the location l or if the second player does not have any more moves. The second player wins, if he can assert that the location l is avoided.

The case where the second player has no more moves can happen if for example the system is on the edge of an invariant region and thus a discrete transition has to happen, but no controllable transition is enabled.

To translate the question whether two systems are in simulation relation into such a hybrid game, we encode the restrictions of the simulation relation into a hybrid automaton that is able to check whether either the distance between the system states is too large, or whether we are not able to find a suitable retiming at a certain point. The retiming is modeled by speeding up/slowing down the system dynamics by multiplying them with either s for the dynamics of the first system or $2-s$ for those of second one. As s can be altered arbitrarily we can emulate all possible retiming relations. We keep track of the temporal distance of the systems in the variable r that represents the integral over $2s-2$. That r indeed models the temporal distance can be seen if one considers the evolution of local clocks. A clock in the first system evolves with rate s while a clock in a second system evolves with $2-s$. In case $s=1$ both evolve with speed 1. Else we have a clock drift of $s-(2-s)=2s-2$.

The spatial distance can be checked directly. We add invariants that force the automaton to go to the *bad* location if the distance is too large. Most of the controllable transitions are only enabled as long as the system variables and timings are close. Only in cases, where the second player reacts on some action performed by an uncontrollable transition this is not directly enforced. Therefore, all uncontrollable transitions lead to a location (second component is in $\hat{L}$) where the second player might react. However, in these locations no time must pass which is enforced using the fresh clock c.

Formally this gives:

Definition 11 (Simulation Game). *Given two real numbers ε, δ, a hybrid system $A=(U_A,X_A,L_A,E_A,F_A,Inv_A,Init_A)$, and another hybrid system $B=(U_B,X_B,L_B,E_B,F_B,Inv_B,Init_B)$, we define, w.l.o.g. assuming $V_A\cap V_B=\emptyset$, a hybrid game $SG=(A\lessdot B,E_c,\{s\},bad)$ in the following way:*

- $A\lessdot B=(U_\lessdot,X_\lessdot,L_\lessdot,E_\lessdot,F_\lessdot,Inv_\lessdot,Init_\lessdot)$
- *The variables of the resulting system are given by $U_\lessdot=U_A\cup U_B\cup\{s\}$ and $X_\lessdot=X_A\cup X_B\cup\{r,c\}$ where w.l.o.g. $(V_A\cup V_B)\cap\{s,r,c\}=\emptyset$ holds.*
- *The locations are given by $L_\lessdot=L_A\times(L_B\cup\hat{L}_B)\,\dot\cup\,\{bad\}$, where $\hat{L}_B$ are duplicates of the original locations in L_B.*
- *Let χ be the following formula: $||x_A,x_B||<\delta\wedge|r|<\varepsilon$, where x_A and x_B are the state vectors of the systems A and B respectively.*
- *The discrete transitions $E_\lessdot$ are the smallest set such that:*
  - *If $(l_A,\phi,l'_A)\in E_A$ then for all $l_B\in L_B$,*

$$((l_A,l_B),\phi\wedge\chi\wedge c'=0,(l'_A,\hat{l}_B))\in E_\lessdot\ .$$

  - *If* $(l_B, \phi, l'_B) \in E_B$ *then for all* $l_A \in L_A$*,*

$$((l_A, l_B), \phi \wedge \chi, (l_A, l'_B)) \in (E_{\lessdot} \cap E_c)$$

    and

$$((l_A, \hat{l}_B)), \phi, (l_A, l'_B)) \in (E_{\lessdot} \cap E_c) \ .$$

  - *For all* $l_A \in L_A$ *and all* $l_B \in L_B$*,* $((l_A, l_B), \neg\chi, bad) \in E_{\lessdot}$*.*
  - *For all* $l_A \in L_A$ *and all* $l_B \in L_B$*,* $((l_A, \hat{l}_B), true, (l_A, l_B)) \in (E_{\lessdot} \cap E_c)$*.*

– *For all* $l = (l_A, l_B) \in L_{\lessdot}$ *or* $l = (l_A, \hat{l}_B) \in L_{\lessdot}$ *we construct* $F_{\lessdot}(l)$ *as* $\dot{r} = 2s - 2 \wedge \dot{c} = 1 \wedge mod(F_A(l_A), s) \wedge mod(F_B(l_B), 2 - s)$ *where* $mod(F, x)$ *alters* f *by multiplying the right side of each differential equation occurring in* F *with* x*.*

– *The invariants of the locations are given by* $Inv_{\lessdot}$ *which assigns each location* (l_A, l_B) *or* $(l_A, \hat{l}_B)$ *an invariant of the form* $Inv_A(l_A) \wedge Inv_B(l_B) \wedge \chi \wedge 0 \leq s \leq 2$*. If* $l = (l_A, \hat{l}_B)$ *we further add* $c \leq 0$*.*

– $Init_{\lessdot} = \{((l_A, l_B), Init_A(l_A) \wedge Init_B(l_B)) \mid l_A \in L_A \wedge l_B \in L_B\}$

Using this game, we can determine whether two systems stand in simulation relation as defined in Def. 9.

Theorem 1. *Given two hybrid systems* A *and* B*. If there is a winning strategy for the second player in the game* $(A \lessdot B, E_c, \{s\}, bad)$ *then* $A \trianglelefteq^{\varepsilon,\delta} B$ *holds.*

Proof. Assume that there is a winning strategy but $A \trianglelefteq^{\varepsilon,\delta} B$ does not hold. From the latter, we know that there is a run of system A such that, no matter which retiming is applied, system B cannot stay close enough. Let this run be ξ. We now construct a winning strategy for the first player using ξ. The first player chooses his actions in a way that the valuations of the variables of the first system at time t coincide with $\xi(t - r)$. The second player is not able to influence the valuations of the variables at those points, as the discrete transitions that it can choose from are those of B. He is also not able to restrict the movement of the first player, as the intermediate locations only allow him to react on actions performed by the first player and as he looses if there is a time deadlock, he can also not avoid time going to infinity. This, as there was no run of B that stays close enough to ξ, eventually leads to a state where the condition $||x_A, x_B|| < \delta \wedge |r| < \varepsilon$ is violated. In this state the first player can choose to enter the location *bad*. This contradicts the assumption that there is a winning strategy for the second player and thus concludes the proof. □

Note that the reverse implication does not hold, as the game demands a tighter coupling of the system behaviors with regard to branching than our notion of ε-δ-simulation.

Corollary 1. *If we restrict the possible moves of the second player by adding differential equations describing the evolution of* s *and he is still able to win the game, then the systems are in simulation relation.*

This follows directly from the fact that Theorem 1 demands the existence of a winning strategy. Now, if we can find a winning strategy for a system where the control of s is further restricted, we still can assert that there is a winning strategy for the original system.

A good strategy to use for controlling s is optimal control with the goal of minimizing the value of $||x_A, x_B||$. As the Euclidean distance contains a square root we w.l.o.g. take the square of the distance as minimization target. This yields equivalent results as the square root is a monotone transformation. For $x_A = (x_{A,1}, \dots, x_{A,n})$ and $x_B = (x_{B,1}, \dots, x_{B,n})$, the square of the distance evolves as follows:

$$\begin{aligned}\frac{d(||x_A, x_B||)^2}{dt} &= \frac{d(\sqrt{((x_{A,1} - x_{B,1})^2 + \cdots + (x_{A,n} - x_{B,n})^2)}^2)}{dt} \\ &= \frac{d((x_{A,1} - x_{B,1})^2 + \cdots + (x_{A,n} - x_{B,n})^2)}{dt} \\ &= \Sigma_{i=1}^{n}(2(x_{A,i} - x_{B,i}) \cdot (s\frac{dx_{A,i}}{dt} - (2-s)\frac{dx_{B,i}}{dt}))\end{aligned}$$

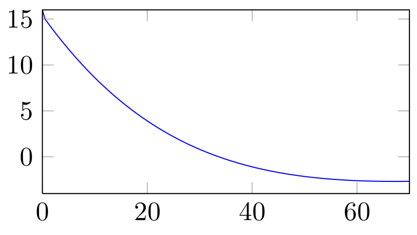
Let s_{min} be the s that minimizes this term. Now choose s in the following way: If $r < \varepsilon \wedge s_{min} > 1$ or $r > -\varepsilon \wedge s_{min} < 1$ choose $s = s_{min}$. Otherwise choose $s = 1$. The resulting strategy, for controlling s can then be encoded into a hybrid automaton and included into the original automaton.

The choice of the strategy is motivated by the fact that the location *bad* can only be entered if the distances between the two systems become too large. As we might be able to trade spatial distance against temporal distance we choose to minimize the spatial distance. However, this strategy is only an heuristic as there are systems, where it is necessary to let the spatial distance increase a bit to for example unify switching timings, as the guard effects might otherwise lead to a violation of the bounds on the spatial distance.

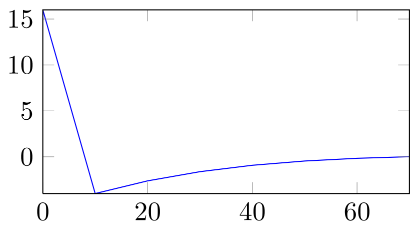
Definition 12. *A hybrid automaton is considered* deterministic *if (1) all of its transitions are urgent, i.e. all the guards of the transitions are overlapping in a singular point with the border of its sources invariant and all trajectories of the mode are pointing outwards of the invariant region at that point, and (2) for each point in time, at most one transition is enabled.*

Let A be a hybrid automaton and B be a deterministic hybrid automaton. If we modify $A \lessdot B$ in a way that the assumptions of Corollary 1, assume that the system values are identical at the initial locations, and are able to show that on all runs of this modified version of $A \lessdot B$ the location *bad* is avoided, then, by Corollary 1, $A \trianglelefteq^{\varepsilon,\delta} B$ holds. The assumptions of Corollary 1 could, e.g., be satisfied by using the optimal control strategy. Thus, we can use a model checker (e.g. FOMC [3], PHAVer [8], SpaceEx [9], or HSolver [20], depending on the system class and complexity) to search for a certificate for the fact that *bad* is unreachable. Provided that suitable inductive invariants can be found, we could also prove that the trajectories of these two uncontrolled systems stay close using a theorem prover for hybrid systems like KeYmaera [19].

On a similar line of thought, if one can prove that the systems stabilize within a certain time, as it is the case for our running example, it also possible to use bounded model checking up until the time of stabilization. This could be performed, e.g., using iSAT [10].



4a: Velocity under continuous control

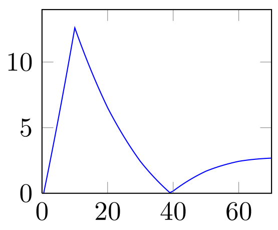


4b: Velocity under sampled control

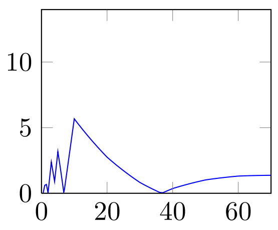
Fig. 4: Simulated velocities over time

Let us now consider our example presented in Sect. 3 with respect to these results. Using MATLAB *Simulink* we can determine that the maximum distance between these two systems if started in the initial state $v = 16$ is 12.7. The velocity trajectories of the two systems are shown in Fig. 4a and Fig. 4b. A plot of the resulting differences is depicted in Fig. 5a.

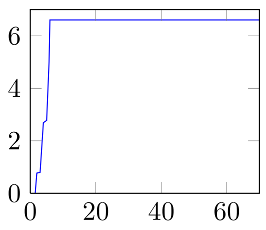
Using optimal control to keep the distance between the two systems minimal as long as the retiming bounds allow and the fact that both systems stabilize at $v = 0$ we can also show that there is a 5-6.61-simulation, if the initial region is restricted to this singular point $v = 16$. This enables us to transfer more knowledge from one model to the other than the previous observation. The resulting trajectory corresponds to applying a retiming relation that results in the temporal differences depicted in Fig. 5c to the original trajectories and we get the smaller differences (compared to Fig. 5a) shown in Fig. 5b.



5a: Without retiming



5b: With retiming



5c: Temporal differences

Fig. 5: Maximal differences of the system velocities over time

6 Preservation of Logical Properties

Now that we have identified sufficient conditions for our simulation relation, we study what properties are preserved by the relation.

The similarity notion defined in Def. 9 can be used to transfer properties from one system to another. As we have upper bounds on the deviations we can use those to weaken the formulas to be sure that they still hold in similar systems.

Theorem 2. *If hybrid systems A and B satisfy $A \trianglelefteq^{\varepsilon,\delta} B$ and $B \models \phi$ then $A \models \phi^{+\delta}_{+\varepsilon}$ where $\phi^{+\delta}_{+\varepsilon} := re_{\varepsilon,\delta}(\phi)$ and $re_{\varepsilon,\delta}$ is defined by:*

- $re_{\varepsilon,\delta}(x \in \mathcal{I}) := x \in \mathcal{I}'$, *where* $\mathcal{I}' = \{a \mid \exists b \in \mathcal{I} : a \in [b-\delta, b+\delta]\}$.
- $re_{\varepsilon,\delta}(f(x_1,\dots,x_n) \leq 0) := f(x_1,\dots,x_n) - \delta \cdot M \leq 0$ *where M is the Lipschitz constant for f.*
- $re_{\varepsilon,\delta}(\neg\phi) := \neg ro_{\varepsilon,\delta}(\phi)$.
- $re_{\varepsilon,\delta}(\phi \wedge \psi) := re_{\varepsilon,\delta}(\phi) \wedge re_{\varepsilon,\delta}(\psi)$.
- $re_{\varepsilon,\delta}(\phi\, \mathbb{U}_{\mathcal{J}}\, \psi) := re_{\varepsilon,\delta}(\phi)\, \mathbb{U}_{\mathcal{J}'}\, re_{\varepsilon,\delta}(\psi)$, *where* $\mathcal{J}' = \{a \mid \exists b \in \mathcal{J} : a \in [b-\varepsilon, b+\varepsilon] \cap [0,\infty)\}$.

The transformation function $ro_{\varepsilon,\delta}$ is given by:

- $ro_{\varepsilon,\delta}(x \in \mathcal{I}) := x \in \mathcal{I}'$, *where* $\mathcal{I}' = \{a \mid \exists b \in \mathcal{I} : a \in [b+\delta, b-\delta]\}$.
- $ro_{\varepsilon,\delta}(f(x_1,\dots,x_n) \leq 0) := f(x_1,\dots,x_n) + \delta \cdot M \leq 0$ *where M is the Lipschitz constant for f.*
- $ro_{\varepsilon,\delta}(\neg\phi) := \neg re_{\varepsilon,\delta}(\phi)$.
- $ro_{\varepsilon,\delta}(\phi \wedge \psi) := ro_{\varepsilon,\delta}(\phi) \wedge ro_{\varepsilon,\delta}(\psi)$.
- $ro_{\varepsilon,\delta}(\phi\, \mathbb{U}_{\mathcal{J}}\, \psi) := ro_{\varepsilon,\delta}(\phi)\, \mathbb{U}_{\mathcal{J}'}\, ro_{\varepsilon,\delta}(\psi)$, *where* $\mathcal{J}' = \{a \mid \exists b \in \mathcal{J} : a \in [b+\varepsilon, b-\varepsilon] \cap [0,\infty)\}$.

As before $\mathcal{I} \subseteq \mathbb{R}$ and $\mathcal{J} \subseteq \mathbb{R}^+$ hold.

The function $re_{\varepsilon,\delta}$ is applied if the current subformula is in a context of an even number of negation, whereas the function $ro_{\varepsilon,\delta}$ is applied to subformulas under an odd number of negations. Note that if the set indexing an until operator becomes empty, the formula is trivially false.

Our notion of similarity can be seen as a decrease of the resolution of the image we have of the system behavior thus blurring the borders. If we originally knew that at some time between t and t' some event would happen, we now have to account for the timing deviations that might occur. Thus, if the event originally happened at time t it might now occur in the worst case already at $t - \varepsilon$. If it originally occurred at time t', the worst case we have to consider is that it now might occur as late as $t' + \varepsilon$. This widens the set of possible time points for the event, thus reducing our knowledge about exact timings. A similar effect happens on the variable valuations.

This theorem can be proven by induction over the formula structure and the time. The induction base is formed by showing that the modified formulas hold at time 0. One important argument for this is the fact that $(0,0) \in \mathfrak{r}$ and of course the bounds on the deviations. Now the only operation that modifies the time at which the formulas are evaluated is the until operator. As stated in the previous paragraph, from the fact that the systems are in simulation relation, we know that the postcondition was originally satisfied during some point in the set annotation, the modified version of the formula might now hold a bit earlier or a bit later but is forced to hold at some point. Up until this point, all the values have to be similar to those originally satisfying the precondition.

The other crucial point is to prove that negated formulas can be transfered. This can be shown by another induction. Again the difficult part is to show that the proposition holds for the until operator. If we assume, the modified formula would not hold, we can use our relation $\mathfrak{r}$ that gives us a point in time where the original pre- and postconditions touches. This, however, is a contradiction to the fact that the until is not satisfied for the original system.

Using the weaker version of similarity we can also transfer some properties. The version is weaker with respect to knowledge about timing. Those timings are only present in the intervals indexing the until-operators in the formulas. Thus weakening these operators by replacing those intervals by $[0, +\infty)$ removes all exact timing informations. Only temporal properties are preserved in that case, e.g. we could still retain knowledge about event orders.

Theorem 3. *If for hybrid systems A and B holds $A \trianglelefteq^{\delta} B$ and $B \models \phi$, where ϕ does not contain any until-operations in a negative context, then $A \models \phi_{\sim}^{+\delta}$ where $\phi_{\sim}^{+\delta} = w(re_{0,\delta}(\phi))$ and w replaces the index of every until operator by $\mathbb{R}^{+}$.*

The proof follows easily from the proof for Theorem 2 by altering the induction steps for the until operator. Unfortunately, we here loose to much information about the timings to keep knowledge about until operations in a negative context, i.e. under an odd number of negations, thus the restriction to positive contexts in this theorem.

7 Summary

In this paper we presented our notion of similarity for hybrid systems that we call ε-δ-simulation. We have given a translation of the question whether one system simulates another into hybrid games and given sufficient conditions under which a winning strategy for these games exists. Further, we have studied what properties are preserved under this notion of similarity.

Currently, we can only determine similarity of a restricted class of models and the approach itself has a large complexity making it easier in many cases to check the properties one wants to transfer on the implementation directly. For future work, we will study how we can incorporate system decompositions into our approach. For this goal, we need to find a way to determine whether two open-loop systems are in simulation relation.

Acknowledgments. We like to thank Ernst-Rüdiger Olderog and Anders Ravn for the fruitful discussions, and Sven Linker, Johannes Faber, and Mani Swaminathan for reading preliminary versions of this paper.

References

1. Alur, R., Henzinger, T.A.: Logics and models of real time: A survey. In: Proceedings of the Real-Time: Theory in Practice, REX Workshop, London, UK, Springer-Verlag (1992) 74–106
2. Bouyer, P., Brihaye, T., Chevalier, F.: O-minimal hybrid reachability games. Logical Methods in Computer Science **6**(1) (2009)
3. Damm, W., Dierks, H., Disch, S., Hagemann, W., Pigorsch, F., Scholl, C., Waldmann, U., Wirtz, B.: Exact and fully symbolic verification of linear hybrid automata with large discrete state spaces. Science of Computer Programming, Special Issue on Automated Verification of Critical Systems (2011) (to appear).
4. Damm, W., Dierks, H., Oehlerking, J., Pnueli, A.: Towards component based design of hybrid systems: Safety and stability. In Manna, Z., Peled, D., eds.: Essays in Memory of Amir Pnueli. Volume 6200 of Lecture Notes in Computer Science., Springer (2010) 96–143
5. Davoren, J.M.: Epsilon-tubes and generalized skorokhod metrics for hybrid paths spaces. [17] 135–149
6. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In Chatterjee, K., Henzinger, T.A., eds.: FORMATS. Volume 6246 of Lecture Notes in Computer Science., Springer (2010) 92–106
7. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42) (2009) 4262–4291
8. Frehse, G.: Phaver: algorithmic verification of hybrid systems past hytech. STTT **10**(3) (2008) 263–279
9. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: Spaceex: Scalable verification of hybrid systems. In Gopalakrishnan, G., Qadeer, S., eds.: CAV. Volume 6806 of Lecture Notes in Computer Science., Springer (2011) 379–395
10. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. Journal on Satisfiability, Boolean Modeling and Computation **1** (2007) 209–236
11. Girard, A., Julius, A.A., Pappas, G.J.: Approximate simulation relations for hybrid systems. Discrete Event Dynamic Systems **18**(2) (2008) 163–179
12. Girard, A., Pola, G., Tabuada, P.: Approximately bisimilar symbolic models for incrementally stable switched systems. In Egerstedt, M., Mishra, B., eds.: HSCC. Volume 4981 of Lecture Notes in Computer Science., Springer (2008) 201–214
13. Henzinger, T.A.: The theory of hybrid automata. In: LICS, IEEE CS Press (1996) 278–292
14. Henzinger, T.A., Horowitz, B., Majumdar, R.: Rectangular hybrid games. In Baeten, J.C.M., Mauw, S., eds.: CONCUR. Volume 1664 of Lecture Notes in Computer Science., Springer (1999) 320–335
15. Henzinger, T.A., Majumdar, R., Prabhu, V.S.: Quantifying similarities between timed systems. In Pettersson, P., Yi, W., eds.: FORMATS. Volume 3829 of Lecture Notes in Computer Science., Springer (2005) 226–241
16. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems **2**(4) (1990) 255–299
17. Majumdar, R., Tabuada, P., eds.: Hybrid Systems: Computation and Control, 12th International Conference, HSCC 2009, San Francisco, CA, USA, April 13-15, 2009. Proceedings. In Majumdar, R., Tabuada, P., eds.: HSCC. Volume 5469 of Lecture Notes in Computer Science., Springer (2009)

18. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems (an extended abstract). In: STACS. (1995) 229–242
19. Platzer, A., Quesel, J.D.: Keymaera: A hybrid theorem prover for hybrid systems (system description). In Armando, A., Baumgartner, P., Dowek, G., eds.: IJCAR. Volume 5195 of Lecture Notes in Computer Science., Springer (2008) 171–178
20. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation based abstraction refinement. ACM Journal in Embedded Computing Systems **6**(1) (2007)
21. Stauner, T.: Discrete-time refinement of hybrid automata. In Tomlin, C., Greenstreet, M.R., eds.: HSCC. Volume 2289 of Lecture Notes in Computer Science., Springer (2002) 407–420
22. Thrane, C.R., Fahrenberg, U., Larsen, K.G.: Quantitative analysis of weighted transition systems. J. Log. Algebr. Program. **79**(7) (2010) 689–703
23. Tomlin, C., Lygeros, J., Sastry, S.: A Game Theoretic Approach to Controller Design for Hybrid Systems. Proceedings of IEEE **88** (July 2000) 949–969
24. Vladimerou, V., Prabhakar, P., Viswanathan, M., Dullerud, G.E.: Stormed hybrid games. [17] 480–484
25. Wulf, M.D., Doyen, L., Raskin, J.F.: Almost asap semantics: from timed models to timed implementations. Formal Asp. Comput. **17**(3) (2005) 319–341